The utility of mathematical fitness-fatigue models for assisting with the planning of physical training for sport: from in silico experiments employing synthetic data, lower-bound operational conditions and model estimation, to the development of software resources for future research.

STEPHENS HEMINGWAY, B.H.

2021

THE UTILITY OF MATHEMATICAL FITNESS-FATIGUE MODELS FOR ASSISTING WITH THE PLANNING OF PHYSICAL TRAINING FOR SPORT

From in silico experiments employing synthetic data, lower-bound operational conditions and model estimation, to the development of software resources for future research

Benedict H. Stephens Hemingway

Department:

School of Health Sciences &

School of Computing

Submitted in partial fulfilment of the requirements for the degree of:

Doctor of Philosophy

May 2021

The Robert Gordon University, Aberdeen, Scotland, UK

Abstract

The greatest potential application of mathematical models in sport science is to predict future performance of individual athletes in response to training with sufficient accuracy to assist with planning of training programs and short tapering periods. The most widely known and investigated set of mathematical models include the fitness-fatigue models (FFMs). However, despite over 45 years of FFM study, problems remaining within the research base and gaps in existing knowledge have limited interpretation of prior research and prevented progression toward practical implementation. These limitations include: 1) inadequate study of model validity in prior experimental study as a result of unsatisfactory model testing; 2) a disorganised literature body without a connective narrative linking previous research and providing consistent recommendations for the requirements and direction of future study; 3) limitations in the structure of basic FFMs matched by little awareness of extensions that have been proposed to address them; 4) no consideration of experimental factors and methods that may interact with model accuracy (e.g., measurement error, testing frequency, parameter estimation); 5) limited practical resources elucidating key concepts, and no tools available to facilitate processes required to fit and evaluate more promising FFMs. Subsequently the aims of this PhD were to 1) Systematise the FFM literature body, providing sufficient detail and structure to the point where there exists a consistent narrative threading the historical literature, pertinent concepts, and contemporary work to address limitations in basic FFM structure; 2) Conduct original study of key experimental factors (measurement error and testing-frequency), and methods of model estimation, that may affect model accuracy or utility; 3) Identify and raise awareness of alternative FFMs beyond the standard model, and advanced methods, that reflect more promising avenues for future research; 4) Develop flexible code tools that facilitate future study and address the current gap in available resources. This first aim was achieved by comprehensive review balancing mathematical rigor with clarity in the communication of concepts and methods ranging from the standard model to the most advanced FFMs. The second aim was achieved by two novel studies that developed *in silico* experimental designs, and that represented prerequisite work prior to any further study of model validity. Study 1 quantified the effects of key experimental factors (measurement noise and testing-frequency) on lower-bound model accuracy in the standard FFM, demonstrating testing practices comprising high error will provide unsatisfactory results and greater deleterious effects of error exist at lower testing frequencies. Study 2 focussed on suitability of a traditional quasi-Newton algorithm for fitting FFMs, by assessing starting point sensitivity and existence/implications of local extrema. Study 2 demonstrated the model-fitting problem is more challenging than researchers have previously acknowledged, and the presence of many local extrema even in the standard FFM may now necessitate global optimisation approaches. Aims 3 and 4 were achieved by development of extensive code resources in the R programming language for fitting and evaluating FFMs, facilitating future study under the most promising models/methods. The original research and systematised literature body provides clearer direction for

future FFM research, guidance with respect to key experimental factors/methods (e.g., measurement practices, estimation, model testing), and reflects the most up-to-date resource available for researchers interested in FFMs. The code tools developed meet the need for flexible practical resources for researchers, and the novel experimental designs developed for the two studies provide a unique and cost-effective approach to study FFMs and potentially other phenomena in sport science.

Acknowledgements

I would like to express my appreciation and gratitude to many colleagues, friends, and family, all of whom have given me their unwavering support throughout this research journey. On countless occasions I have been moved by their patience and generosity and spurred on by their encouragement.

My sincere thanks and gratitude goes to my superb supervisory team, Dr Paul Swinton, Dr Katherine Burgess, and Dr Eyad Elyan, whose collective knowledge and expertise have helped shape my thinking and perspectives across several areas of research. In particular I would like to express special thanks to my principal supervisor Dr Swinton, whose intellectual brilliance and confidence in the research topic has been a dependable source of constructive influence. His selflessness, persistent optimism, and generosity with his time has had a significant positive impact on me, and I have thoroughly enjoyed his mentorship. Without his efforts, outlook, and feedback at several points in the process, the goals of this project could not have been realised. I look forward with great eagerness to soon having the time to work on many of the other interesting research endeavours we have discussed over the years.

There are numerous others who deserve acknowledgement and my thanks, including those who I have been fortunate enough to collaborate with during this research project. In particular, Dr Ben Ogorek who assisted with the development of the Kalman filter code in chapter 6. As both a colleague and friend, I have greatly enjoyed and benefited from all of our discussions spanning performance modelling, statistics, and data science. Similarly, Christian Rasche deserves my deepest thanks for providing illuminating feedback and expert knowledge at many points. Further thanks must be extended to Leon Greig and Mladen Jovanovic, with whom I have thoroughly enjoyed insightful collaboration, and thanks must also go to Andrea McMillan from the graduate school, whose support and encouragement throughout this research project has consistently been above and beyond.

To my close friends, thank you for your support during the completion of this work. In particular, to my housemates Andrew and Cameron, for putting up with me during some of the most stressful points. Your enthusiasm, encouragement, and interest in the work throughout the process has meant a lot. To Sam, thank you for your encouragement and words of mathematical wisdom. To Fraser, thank you for bringing me into the field of strength and conditioning and developing me as a coach. Coaching together over the years and our regular discussions have greatly informed my understanding of the practicalities of sport science and strength and conditioning.

Finally, to my parents, Patricia and Philip, thank you for your support, love, and relentless belief in my abilities. This work is dedicated to you.

Related publications

Peer reviewed

- Stephens Hemingway, B., Burgess, K., Elyan, E., & Swinton, P. (2019). The effects of measurement error and testing frequency in applying the Fitness Fatigue Model to resistance training: A simulation study. *International Journal of Sports Science and Coaching*, 0(0), 1–12. doi.org/10.13140/RG.2.2.19730.56005
- Greig, L., Stephens Hemingway, B., Aspe, R. R., Cooper, K., Comfort, P., & Swinton, P. A. (2020). Autoregulation in Resistance Training: Addressing the Inconsistencies. *Sports Medicine*. doi.org/10.1007/s40279-020-01330-8
- Swinton, P. A., Stephens Hemingway, B., Saunders, B., Gualano, B., & Dolan, E. (2018). A Statistical Framework to Interpret Individual Response to Intervention: Paving the Way for Personalized Nutrition and Exercise Prescription. *Frontiers in Nutrition (Open Access)*. doi.org/10.3389/fnut.2018.00041

In preprint or currently under peer-review

- Stephens Hemingway, B., Swinton, P. A., & Ogorek, B. (2021). The suitability of a quasi-Newton algorithm for estimating fitness-fatigue models: Sensitivity, troublesome local optima, and implications for future research (An in silico experimental design). *SportRxiv (Preprint)2*. doi.org/10.31236/osf.io/dx7gm
- Stephens Hemingway, B., Greig, L., Jovanovic, M., Ogorek, B., & Swinton, P. (2021). Traditional and contemporary approaches to mathematical fitness-fatigue models in exercise science: A practical guide with resources. Part I. *SportRxiv (Preprint)*. doi.org/10.31236/osf.io/ap75j
- Swinton, P., Stephens Hemingway, B., Rasche, C., Pfeiffer, M., & Ogorek, B. (2021). Traditional and contemporary approaches to mathematical fitness-fatigue models in exercise science: A practical guide with resources. Part II. *SportRxiv (Preprint)*. doi.org/10.31236/osf.io/5qgc2

Conference proceedings

- Stephens Hemingway, B., Burgess, K., Swinton, P. (2017). The effects of measurement error and testing frequency in applying the Fitness Fatigue Model to resistance training: A simulation study. *The UK Strength & Conditioning (UKSCA) Annual Conference.* (Poster & Abstract)

Software

- Stephens Hemingway, B., Ogorek, B. (2020). The fitness-fatigue model project: An open-source codebase and decentralised scientific research project investigating models of physical training and performance in sport. fitnessfatigue.com
- Jovanovic, M., Stephens Hemingway, B., & Swinton, P. (2020). dorem: Dose Response Modelling in R (0.0.9000). doi.org/10.5281/zenodo.3757085; dorem.net

Availability of materials

An online repository has been established to provide access to the supplementary materials, data (e.g., inputs, results), and code files associated with this thesis, particularly relevant to chapter 6. In many cases these files can be - or have been purposely designed to - be adapted and used by researchers for their own investigations in fitness-fatigue modelling. Specific files contained in the repository are also referenced at throughout the work at relevant points.

The main directory of the repository can be found at:

github.com/bsh2/thesis

Files contained within the repository:

Chapter		Associated files
2	**Literature review**	Code associated with plots
4	**Research study 1**	Experimental code and supplementary files
5	**Research study 2**	Experimental code and supplementary files
6	**Software development**	Code files containing the practical resources developed for researcher education and future research

All code presented in this thesis was written in the statistical programming language R (r-project.org)

Repository license: GNU GPL v.3 (gnu.org)

Redundancy: A carbon copy of the files contained at the above repository will be stored by RGU library (library@rgu.ac.uk) and made available to download from their OpenAir system alongside this thesis at point of archive.

Table of Contents

List of tables

List of figures

Figure	Chapter	Section	Description	Page
6.19	6	6.2.5	Similar boxplots to figure 6.18, summarising fitted parameter and train/test prediction error variation across the main train-test split (block 1 train, block 2 test), with the model training also comprising multiple fitting iterations from many starting points	225
6.20	6	6.2.5	Plotting the variation in iterations (lines) from the main train-test split (train on block 1, blue points; test on block 2, red points) over the time-series	226
6.21	6	6.3	Hill-saturation for increasing loads under different values of δ, γ, with κ fixed	227
6.22	6	6.3	**Left**: Sigmoidal curve produced by the hill function for changing γ with δ fixed to 1. **Right**: Demonstration of the flattening of the curve and inertia of the hill function to the threshold value by increasing δ to 2 and reproducing the plot on the left with the same values of γ	230
6.23	6	6.4	Simulated state-space FFM with random noise (listing 6.39, lines 10-12)	224
6.24	6	6.4	Kalman-filtered model under true parameters and simulated data developed in listing 6.39	237
6.25	6	6.4	Fitted Kalman model	241
6.26	6	6.4	R output: A comparison between the truth model underpinning the simulated data, and the fitted Kalman-filtered model with parameters recovered by the optimisation procedure.	241

List of code

All code listed below was written in R (r-project.org.uk)

Listing	Chapter	Section	Description	Page
6.40	6	6.4	Kalman filtering function	236
6.41	6	6.4	Kalman filtering demonstration for the simple case of the simulated data under true parameters in listing 6.38	237
6.42	6	6.4	Defining a function to fit a state-space model with Kalman filter to data	239
6.43	6	6.4	Kalman filtering demonstration for the simple case of the simulated data under true parameters in listing 6.38	240

Abbreviations

Abbreviation	Term
BDF	Backward differentiation formula
CG	Conjugate gradient
CoV	Coefficient of variation
CMAES	Covariance matrix adaptive evolution strategy
CPU	Central processing unit
CRAN	Comprehensive R archive network
CSP	Constraint satisfaction problem
CV	Cross validation
DDE	Delay differential equation
DE	Differential evolution
EnD	Every n Days
FFM	Fitness-fatigue model
GA	Genetic algorithm
GAS	General adaptation syndrome (model)
GPS	Global positioning system
HR	Heart rate
L-BFGS-B (BFGS)	(Limited memory modification) *of the* (Broyden-Flecher-Goldfarb-Shannon) *search algorithm with* (Box constraints)
LDH	Lactate Dehydrogenase
MAD	Median absolute deviation
MAPE	Mean average percentage error
MLE	Maximum likelihood estimation
NLS	Nonlinear least squares
ODE	Ordinary differential equation
POMS	Profile of mood states

Abbreviation	Term
PSO	Particle swarm optimisation
RMSE	Root mean squared error
RSS / SSE	Residual sum of squares / Sum of squared errors
rTSS, TSS	(Running) Training stress score
SCM	Source code management
SHBG	Sex hormone binding globulin
sRPE, RPE	(Session) Rate of perceived exertion
TRIMP	Training impulse (dose)
VDR	Variable dose-response (model)
VO_2max	Maximal oxygen uptake
VTRS	Velocity at ventilatory threshold

Glossary of key terms

Term	Definition
Athlete	A person who participates in organised sport, or competitive physical tasks involving non-trivial levels of physical exertion
Algorithm	A finite set of instructions that can be implemented within a computer to solve problem(s) or perform computation(s)
Closed skill sport	A self-paced sport where the environment is highly consistent and predictable (e.g., running, swimming, weightlifting)
Competitive outcome	The resultant of a competitive event (e.g., win/loss/draw or ranking)
Competitive performance	A term capturing the interaction of multiple dynamic processes and events contributing to a terminal state (competitive outcome)
Computer experiment (*in silico* system)	An experiment used to study a computer simulation
Computer model	An algorithm that captures the behaviour of or replicates the modelled system
Criterion performance	The physical performance measure selected as the modelled output (target) of a fitness-fatigue model
Cross validation	Also called out-of-sample testing, refers to a set of techniques to assess the generalisability of a statistical model to an independent data set
Endurance sports	Sports requiring sustained physical exertion over a long duration, often comprising a large locomotive component (e.g., distance running or cycling)
Mathematical model	An abstraction or representation of a system or phenomena via mathematical structure(s)
Physical capability	A relatively stable latent variable representing the athlete's maximal level of physical performance. Estimated in practice from maximal-effort physical performance testing or predictive approaches.
Physical performance	The measured value of a dependent variable (typically denoted p in this work) in a sporting or biomechanical task or event, that typically reflects dimensions of fitness (e.g., strength, power, stamina, speed).

Term	Definition
Performance modelling	An area of scientific study in sport science concerned with identifying accurate models of human response to training on physical outcomes
Physical training	The purposeful execution of exercise-based tasks with the goal of positively influencing change in or retaining current physical capability
Resistance training	A single or multi-joint exercise task involving muscular contraction under external load or resistance
Simulation	The process of running a computer model
System	Interaction of multiple elements forming a unified whole
Training load	The input variable manipulated to elicit the desired training response (further subdivided into external and internal load variables)
Training program	An ordered schedule or detailed plan of future training activities (physical or skill-based)

Chapter 1: Introduction

1.1 A brief introduction to the work

A key axiom of training theory is that physical response to exercise is systematic and the process is not random (Stone, Stone and Sands, 2007), such that it can be studied and understood in a way that may permit the value of different exercise systems to be distinguished and therefore training better organised (e.g., periodised) (Stone, Stone and Sands, 2007; Verkhoshansky and Siff, 2009; Bompa and Buzzichelli, 2018; Cunanan *et al.*, 2018). Furthermore, if an accurate model of human response to physical training can be identified, outcomes of the training process can be better predicted and possibly influenced more precisely at an individual or group level (Schaefer, Asteroth and Ludwig, 2015). Identifying and operationalising an accurate model of human response to physical training remains a longstanding goal of sport science, and the principal focus of researchers working in the area of 'performance modelling' (Banister *et al.*, 1975; Perl, 2001; Rasche and Pfeiffer, 2019).

Although the fitness-fatigue model (FFM) was one of the earliest and most popular approaches to addressing the need for an accurate model of training response (Banister *et al.*, 1975), it was preceded by, and arguably extends the conceptual model referred to as the general adaptation syndrome (GAS) model, developed in the 1940s and 1950s by Hungarian-Canadian scientist Hans Selye (Selye, 1946, 1950, 1951). The GAS model was not specifically designed to model exercise response, rather Selye's desire was to develop a conceptual model that could explain the core relationship between stress and adaptation in all biological systems, whereby researchers could adopt and develop it further within their own fields (Selye, 1951; Cunanan *et al.*, 2018). The GAS model proposed a three-phased response to stress including an *alarm*, *resistance*, and *exhaustion* phase (Selye, 1951). In the context of training response, the alarm phase describes short-term negative impact of a stimuli (such as exercise), commonly termed acute fatigue, resulting in a slight decline in physical performance (Cunanan *et al.*, 2018). The subsequent resistance phase was then said to occur as the acute negative response resolves and adaptation occurs ultimately leading to an increase in performance, before a downward trend to the original baseline begins in the absence of further training. The final phase, exhaustion, was said to occur if the application of training is too high in magnitude and prolonged that it results in physical exhaustion, detraining, and ultimately a chronic reduction in performance below baseline (chronic fatigue and overtraining) (Chiu and Barnes, 2003; Cunanan *et al.*, 2018). The FFM first introduced by Banister *et al.* (1975) posits that training results in two separate after-effects, fitness and fatigue, the combination of which describes change in performance. The fitness-fatigue and GAS models mirror each other in the basic case, however the FFM splits the physiological response into two channels, that conceptually can then be split further into acute and chronic effects to facilitate development of training

paradigms (Chiu and Barnes, 2003; Jeffries *et al.*, 2020). In research and practice, conceptual models such the GAS and FFM are important tools for informing future investigation and provide a reference point to guide decision making (Cunanan *et al.*, 2018; Jeffries *et al.*, 2020).

Translation of the conceptual FFM into a mathematical model also parameterises this response framework in a manner that permits, crucially, individualisation and therefore addresses differences in adaptive and fatigue responses between athletes (Greig *et al.*, 2020). However, it should be clarified that facilitation of individualisation by model structure does not suggest effectiveness at doing so. In addition, the mathematical structure of the FFM can be generalised to encompass components that account for aspects such as nutrition, psychology, and sleep to be incorporated under their own transfer functions, as was the original intention of Banister *et al.* (1975). The FFM, its history, conceptual basis, mathematical detail, and various extensions are covered extensively in chapter 2 and this introduction is only intended to provide a broader overview of the origins of FFMs and their importance to sport science, to help situate the work of this thesis. Crucially, the FFM represents one of only two conceptual models (Banister *et al.*, 1975; Perl, 2001) with unique mathematical properties that have attempted to generalise the response to training on performance (Jeffries *et al.*, 2020), capturing behaviour as a whole rather than as the sum of individual mechanistic parts. Although this type of systems model can be improved by further understanding of integrated physiological processes involved at a mechanistic level, the grey-box approach of the FFM (Figure 1.1) offers the advantage that it is both operable, testable, and adaptable from an early stage based only on knowledge and assumptions of conceptual relationships between inputs (quantified training load) and observed outputs (measured performance values) arising from the training process (Banister *et al.*, 1975; Calvert *et al.*, 1976). The model is also flexible enough that it can be extended to encompass factors such as psychology, nutrition, and sleep as first intended by the original authors with these further components each possessing their own transfer functions linking inputs to their contribution on performance (Banister *et al.*, 1975; Calvert *et al.*, 1976).

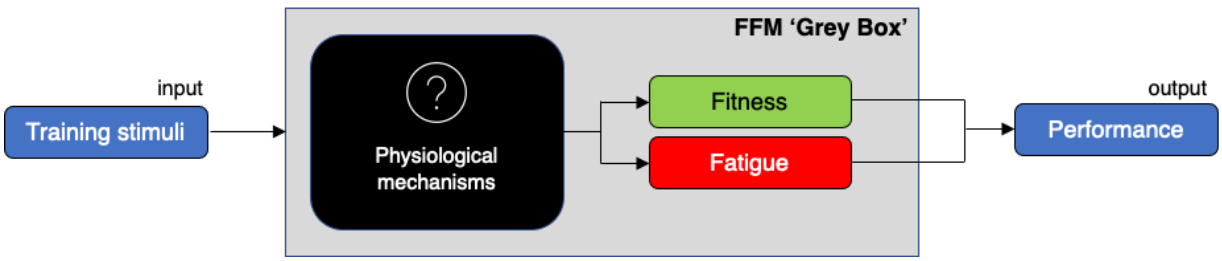


Figure 1.1: An overview of the 'dual channel' FFM represented as a grey box system that combines a partial theoretical structure with inputs and outputs (to be derived from data) to model the training response.

As alluded to in the opening of this chapter, the greatest potential application of mathematical FFMs is to predict the future performance of individual athletes (or small groups) in response to physical training with sufficient accuracy that these models can be used to assist in the planning of future training programs and short tapering and peaking periods prior to competition (Fitz-Clarke, Morton and Banister, 1991; Schaefer, Asteroth and Ludwig, 2015). Despite over 45 years of continued study, problems remain within the research base and there are clear gaps in existing knowledge that fundamentally limit both the interpretation of prior FFM research and experimental practices within future work, collectively precluding sufficient progression in the field to enable implementation in practice. These include: 1) inadequate methods used to study model accuracy in most previous research, and an associated insufficiency in the availability of evidence to support or disregard FFMs as valid models with counterfactual properties; 2) a disorganised literature body without a connective narrative that links previous work, clearly explains mathematical concepts, and that provides consistent recommendations for the requirements and directions of future study; 3) limitations in the structure of basic mathematical FFMs, matched by low awareness within sport science of contemporary extensions and updates that have been proposed in the literature to address them; 4) no consideration of key experimental factors and methods that may interact with model accuracy within real-world study, e.g., measurement error, measurement frequency, method of estimation of model parameters from data; and 5) very few practical resources elucidating key concepts, and no tools available to facilitate processes required to fit and robustly evaluate more promising FFMs and advanced methods.

Despite consistent appearance of the model in the sport science literature and its conceptual place in the discipline (Chiu and Barnes, 2003; Bompa and Buzzichelli, 2018), the issues described above emphasise the high bar to entry in conducting effective study in the area of performance modelling and FFMs. Most previous experimental research appears to be replications of itself, focussing on fitting models to slightly different populations studied under small sample sizes, with only some ingenuity in how the model inputs were quantified and performance of different individuals measured (summary literature tables provided in appendix B). Despite the history of the work and enormous creativity the model framework affords, the issues described highlight how far the field is from assessing the effectiveness of these models, due to a range of factors that this thesis tries to partly address. Fundamentally, to be applied in the future within training program design frameworks, researchers must establish a threshold of evidence to answer the central question of whether a particular FFM can be identified that reflects an accurate conceptualisation of training response and has strong predictive power.

The aims of this thesis are briefly outlined here and examined again in detail in chapter 3 accompanied by the thesis objectives following detailed synthesis of the relevant literature. The aims of the thesis were: 1) to systematise the FFM literature body, providing sufficient detail and structure to where there

exists a consistent narrative threading the historical literature, pertinent concepts, and contemporary work to address limitations in basic FFM structure (Chapter 2); 2) to conduct original research studying key experimental factors (measurement error and testing frequency) and methods of model estimation that may affect model accuracy or utility (Chapters 4 and 5); 3) to identify and present alternative FFMs beyond the basic model, and advanced methods, that reflect more promising avenues of future research (Chapters 2 and 6); and 4) to develop flexible code tools that facilitate future study and address the current gap in available resources (Chapter 6).

The structure of the thesis is split into 7 chapters. The remainder of this chapter is focussed on discussing several background topics introduced to provide context to the work, including management of the training process, definitions and operationalisation of key terms used in the thesis such as 'physical performance' in relation to performance models, and a description of the research field with respect to an increase in available technology for practitioners to collect regular data. In chapter 2, the FFM literature is reviewed critically and in depth, with sections building on one another. By the end of chapter 2, the reader will have a stronger awareness of the timeline of FFM research, pertinent mathematical concepts, existing limitations, practical considerations (e.g., model inputs, estimation, evaluation), and advanced FFMs proposed to address limitations in basic model structure. In chapter 3, a research framework guiding the project is presented and discussed, particularly in relation to challenges arising early during the research project, and the current state of the research field. By the end of chapter 3, the reader will have a good awareness of the overall frame of the project, the thesis' contribution to knowledge, and the impact of experimental and practical work carried out. In chapters 4 and 5, original experimental study is conducted investigating the effects of measurement error, testing frequency, and parameter estimation on model utility. These studies adopted *in silico* (i.e., *in computer*) experimental designs that are largely novel to sport science. At the start of chapter's 4 and 5 these experimental designs are discussed and appraised. By the end of chapters 4 and 5, the reader will have a good understanding of the experimental design underpinning the original research, and an awareness of the practical implications of the results of these studies on prior and future study of model validity. In chapter 6, an extensive set of code tools are developed in the programming language R which is becoming increasingly popular among sport scientists. The code tools can be used to fit and evaluate FFMs from the standard case through to the most advanced models and methods that have been proposed to address limitations in model structure. By the end of chapter 6, the reader will have a good awareness and improved understanding of practical techniques and available tools and resources that can be developed in the R environment to facilitate implementation aspects of FFM research. Chapter 7 provides a summary of the work, including discussion of the main findings, an analysis of impact of the thesis on current knowledge, limitations of the PhD, and directions for future study.

1.2 Research background

1.2.1 Management of the training process – a tractable optimisation problem?

Scientific study and practical application of physical training in athletic populations is primarily concerned with stimulating and optimising adaptation to training interventions (Bompa and Buzzichelli, 2018). More specifically, maximising improvement in dimensions of general and sport-specific fitness (e.g., strength, power, speed, endurance) via planned training programs, robust nutritional strategies, minimising fatigue (careful 'dosing' and planned recovery), and altering biomechanical factors that may improve (or currently impede) physical performance (Stone, Stone and Sands, 2007; Bompa and Buzzichelli, 2018). A central premise of training theory is that a structured training approach can be developed that targets physiological characteristics pertinent to the demands of a particular sport, and that this is tailored to the individual needs of the athlete (Stone, Stone and Sands, 2007; Bompa and Buzzichelli, 2018). Management of the training process in athletic groups involves exposing the athlete to progressive demands structured on a periodic or cyclic basis (Plisk and Stone, 2003). Planning of structured training evolves from associated processes and outcomes of human decision making, informed by the integration of scientific knowledge with individual practitioner experience, general consensus, and constrained by available means (Bompa and Buzzichelli, 2018). Development and individualisation of training programs is far from an exact science (i.e., informed by deterministic laws), and arguably the process of planning and conducting training comprises a significant 'artistic' element whereby coaches incorporate guesswork and heuristics informed by anecdotal evidence and collective coaching experience. It is also typically a multi-objective problem and almost never unconstrained (Schaefer, Asteroth and Ludwig, 2015), and there may be several prospective solutions that yield similar (good) outcomes. Furthermore, specific objectives of a given training period may fall within several distinct categories, each with different planning implications (Bompa and Buzzichelli, 2018). Categories include, general physical fitness, sport specific fitness, technical skills, psychological factors, injury resistance, and improvement in an athletes theoretical and tactic knowledge (Bompa and Buzzichelli, 2018). Logistical constraints may either be known in advance, such as competitive scheduling, available equipment, facilities and budget, administrative factors, and contact time with athletes. Or may appear suddenly within a planned training period, e.g., injury, illness, or personal emergency, routing the athlete on a different course to that previously intended within the plan.

Most coaches working in high level sport are well versed in prioritising objectives, planning training within the bounds of identified constraints, tailoring training programs to both general and specific physical demands of a sport and adjusting to unexpected events or the personal situation of the athlete. High level coaches are also often able to consistently find 'good solutions' which add value to an

athlete's physical performance. Nonetheless, given that the set of constraints is typically quite large and often well-defined, and that the possible set of training plans (under these constraints) is also moderately sized, optimising the planning of training may still reflect a tractable problem that can be at least partially solved, or more aptly, assisted by quantitative approaches (Schaefer, Asteroth and Ludwig, 2015; Stein *et al.*, 2017; Connor, Fagan and O'Neill, 2019). The training program 'optimisation' problem can be broken down into two aspects: 1) logistics and 2) training response. Logistics is the much more straightforward aspect to tackle. Even with little knowledge of training theory, the space of possible training programs can quickly be reduced based on logistical constraints. For example, times and dates the coach or athlete is available to train, competitive scheduling, and/or available equipment or facilities provides a template from which possible training plans could emerge. In certain circumstances a coach may also be able to formulate more subjective constraints surrounding the logistics of training planning based on their understanding of an athlete's psychology or character, particularly with regards to exercises unlikely to be tolerated or that might limit an athlete's engagement. In many cases, the use of compute resources is not typical or necessary (although may still be useful) to solve the logistical aspect in isolation.

The much harder aspect of this optimisation problem is filling in the remaining gaps in the program space (after logistical constraints have been met) with a robust training plan. Particularly one that selects activities based on an understanding of or ability to predict individual response to acute or chronic stressors on the desired training outcome (e.g., performance). At the heart of applied sport science is the pursuit of new knowledge that can be used to predict population response to training stressors and enhance performance, toward the ultimate ambition of individualising training (Haff, 2010). However, the research body is a long way off from this ambition. At present, coaches can only make speculative predictions about individual response, and are often forced to extrapolate from short term intervention studies where estimated average response (to a given stressor) is reported across a small sample of participants often drawn from a non-elite populations (Cissik, Hedrick and Barnes, 2008; Jeffreys, 2015). This knowledge is then amalgamated (consciously or unconsciously) within a mental framework (or painted by coaches as a *training system*) comprising their own experience and intuition, and an athletes available training history. However, large inter-individual variability exists in response to virtually all training interventions (Mann, Lamberts and Lambert, 2014; Swinton *et al.*, 2018). Furthermore, most intervention based and observational research available to coaches has been conducted on non-elite populations (e.g., university students, youth athletes, or recreational trainees) (Haff, 2010), and is frequently weak with regard to aspects of experimental design (e.g., cross-sectional, low statistical power, unrepresentative or non-specific training programs, short familiarisation period and duration) (Cissik, Hedrick and Barnes, 2008). Theoretically, to reach a point within sport and exercise science where coaches can precisely adjust for short term between-athlete variation and long term within-athlete variation within the process of structuring an individual training

plan, a substantial scientific effort is required. If a purely reductionist paradigm was selected as the scientific approach toward achieving a counterfactual model of individual response, researchers would need to identify and deconstruct physiological and biomechanical mechanisms contributing to the between-athlete variation in training response within elite populations, sub-levels of this population (e.g., males, females), and for a wide variety of training modalities and measured outcomes. This would be an extensive undertaking, costs would be exorbitant, and there are no guarantees that such mechanisms could be identified or measured. Evident is that utilising this type of approach alone is likely to scale rapidly beyond economic and logistical resources of most academic institutions, even before difficulties in accessing elite populations are considered. In addition, planning of training interventions in elite populations via standard controlled approaches is at odds with what would be expected in a high-performance environment, and likely to be intractable due to conflicts of interest. Unsurprisingly, elite populations (and their sponsors) are first and foremost, interested in winning in the present moment, and furthering scientific progress without direct renumeration in the form of gaining a non-trivial edge in competition or immediately appliable practices is understandably not a priority (Haff, 2010).

An alternative approach to developing predictions of individual response (and subsequent optimisation of training) involves combining analytical thinking within a complex systems (modelling) approach. This was first introduced mathematically to the literature by Banister *et al.* (1975) in their seminal FFM paper. Under this type of mixed approach, conceptual components representing relationships between training and physical performance are identified, and mathematical transfer functions are ascribed to them forming quantitative relationships between system inputs and output (Calvert *et al.*, 1976). One advantage of this approach is that the model (mathematical abstraction) does not necessarily have to map directly to underlying physiological mechanisms to have a chance at meaningful predictions, but rather considers them indirectly via exchanges between input and output flows. Modelling is by definition a simplification of reality, and it is in our interests to construct the simplest mathematical models that can predict observed phenomena. Despite this approach having existed in sport science since the mid-seventies, when Banister *et al.* (1975) proposed the first systems model for describing athletic progress (in response to training) in terms of two model components (fitness and fatigue), this work has remained largely out of the realms of coaching practice despite the FFMs conceptual significance (Chiu and Barnes, 2003; Taha and Thomas, 2003). In addition, the original mathematical model by Banister *et al.* (1975) provided a scaffold from which several extensions of this model have since been proposed (Busso *et al.*, 1990), each addressing some limitation in model structure or incorporating advancements in training theory (Calvert *et al.*, 1976; Busso, 2003; Hellard *et al.*, 2005; Kolossa *et al.*, 2017; Turner *et al.*, 2017), each employing a systems approach with no direct link to underlying physiological processes (Taha and Thomas, 2003). Fitness-fatigue modelling is the focus of chapter 2 (literature review), but its relevance here is its importance

as one of the few counterfactual models within sport science (Jeffries *et al.*, 2020), and its potential use in predicting training response. The FFM reflects one of the very few proposed avenues to addressing the problem of individualisation of training planning based on quantitative prediction in sport science. If a suitable FFM can be identified and fitted to an individual athlete, the optimisation problem (as a function of both logistics and training response) can be formally specified as a seemingly solvable mathematical problem (Schaefer, Asteroth and Ludwig, 2015; Kumyaito, Yupapin and Tamee, 2018; Proshin and Solodyannikov, 2018; Connor, Fagan and O'Neill, 2019). Subsequently, robust training design solutions may be identifiable within an optimisation framework operationalised by an FFM, to support practitioner decision making in practice. This concept of training program optimisation is also the focus of a growing body of research that has investigated applications of the FFM as a training response framework to study the concept of training program optimisation as a manipulable computer science problem (Schaefer, Asteroth and Ludwig, 2015; Turner *et al.*, 2017; Kumyaito, Yupapin and Tamee, 2018; Proshin and Solodyannikov, 2018; Connor, Fagan and O'Neill, 2019). Although promising progress has been made in this area with regard to quantifying program constraints (e.g., maximum/minimum rates of progress) and identifying adequate solvers, this work has not concerned with or examined the validity of FFMs. Consequently, the training program optimisation research neighbouring the FFM literature reflects the step beyond model development and sets the stage if new or existing FFMs can be proven useful under robust scientific study.

1.2.2 Advances in technology and increased access to field data creating new opportunities

There are two measures considered important in determining the capacity to compute information: the number of operations that can be performed per second, and the amount of information that can be stored (Hilbert and Lopez, 2012). By the start of the millennium and what is commonly referred to as the start of the information age, improvements in the capacity of general-purpose computers had almost matched estimates of human cognitive capacity and quickly went on to surpass these significantly. As early as 2002, several companies recognised the potential for exploiting these increases in computational power and began to develop systems and technologies for researchers and practitioners working in sport and exercise science. Notable examples include first-generation automatic video tracking systems (e.g. Vicon ®, Oxford Metrics; and Quintic, Quintic Consultancy), rudimentary three-dimensional virtual training environments, the use of radar to capture motion as a function of time, and force platforms to measure forces across three orthogonal axes (Liebermann *et al.*, 2002). At that time, novel technologies promised practitioners new means of providing quantifiable and accurate feedback to their athletes during training and competition, and approaches to inform decisions surrounding prescription of future training (Liebermann *et al.*, 2002). Much of the original machinery has evolved significantly since the early 2000s, taking advantage of improvements in areas such as

computation speed, storage capacity, information transmission, microsensor technology, and automation. These advances have been accompanied by the natural increase in the frequency, volume, and variety of time-series data generated, purchased and typically stored by individuals and teams (Morgulev, Azar and Lidor, 2018). Increased access to data in sport science spanning the last twenty years has increased the uptake of measurement technology to assist in training monitoring and the study of physical performance response within elite athletic environments (Jobson *et al.*, 2009; James and Petrone, 2016; Williams *et al.*, 2018). Just some examples of modern technology now in daily use across high-performance sport environments include: 1) 'wearable' kinematic and kinetic devices that can capture inertial variables such as barbell velocity and acceleration during resistance training exercises; 2) devices such as watch and strap based GPS, and video tracking technology, that both provide high-frequency spatial-temporal data; 3) bike computers with paired power-meters that provide and can transmit via mobile networks detailed information regarding torque, angular velocity, and power during training and competition; 4) computer-vision machinery to track and record player actions during team sport events; and 5) heart rate monitors and lactate analysers to assess and adjust relative training intensity (James and Petrone, 2016).

Increased collection of data spanning the last two decades in modern sport science practice presents a unique opportunity for researchers and practitioners to utilise techniques from areas such as informatics, mathematics and statistics to derive new knowledge of the training practice, study predictions of future response within mathematical frameworks, and derive new approaches to inform decision making processes. Although there is a balance to be achieved between research conducted in laboratory and practical environments, advances in technology offer new ways to involve and engage coaches in the scientific process and integrate domain expertise in developing areas such as data analytics and predictive modelling (Mello, Leite and Martins, 2014). In typical academic research (i.e., laboratory) settings, data collection usually only takes place over a finite and often limited period of time. However, in practice the training process is cyclical or periodic and involves many test-adjust feedback loops that are not possible to study in laboratory settings due to prohibitive costs and the level of commitment required from participants. Therefore, despite certain inherent challenges and limitations associated with observational field data and retrospective analyses (Passfield and Hopker, 2017), there has never been a better opportunity in the history of the sport science for researchers and practitioners to collaborate, via sharing and pooling of data that can be used to address issues at the heart of daily practice.

In the broader context of this PhD project, the hypothesis that collection of high volume data will continue to rise and be more effectively applied to characterise the physical stress endured by athletes in training and competition will be considered when identifying practical applications and future recommendations in fitness-fatigue modelling. It is argued that FFM development depends crucially

on an integrated approach between practitioners and researchers in the future due to the limitations of data collection in typical research settings.

1.2.3 Performance and training: Clarifying terms and relationships

Prior to the literature review it is beneficial to examine definitions of terms that will be used in describing the practices involved in fitness-fatigue modelling. These include physical training, training load, and performance. The former terms, *physical training* and *training load* have received sufficient attention in the literature yielding precise definitions used fairly consistently (Coutts, Crowcroft and Kempton, 2017; Impellizzeri, Marcora and Coutts, 2019). The latter, *performance*, appears to have a wider scope in its usage within the field of sport science, and has been used broadly to refer to both the processes involved in carrying out a sporting task and its outcome. For example, Haff (2010) conducted a roundtable discussion of several prominent sport scientists and sought to identify the central scientific concerns of sport science. In Haff (2010) the word 'performance' and associated terms 'athletic performance' and 'sporting performance' feature numerous times but are never clearly defined by any of the authors. In one such instance, in answer to a central question posed, "what is sport science", several sport scientists stated that the aim of sport science was to improve sports performance (Haff, 2010). For example, Bishop states "*sport science can be thought of as using the scientific process to guide the practice of sport with the ultimate aim of improving sport performance*", and Stone says "*sport science deals with sport performance enhancement*" (Stone, Sands and Stone, 2004; Haff, 2010). These statements, and the phrases "improving sports performance" and "sport performance enhancement" imply that the respondents view performance as a measurable or quantifiable phenomena, possibly via a principal variable or indirectly through multiple related variables. In the context of performance modelling, the term performance must be defined more precisely to be operationally useful and refers to a dependent variable (outcome) the model is trying to predict. Nevertheless, on its own this definition is still too vague, and its use benefits from being partitioned further into two auxiliary terms: 1) physical performance; and 2) competitive performance; each with specific definitions and different constructs. This is the main purpose of the following subsections to 1) briefly define these auxiliary terms (physical and competitive performance) in a manner that permits them to be understood in relation to the fitness-fatigue modelling process; 2) discuss measurement of physical performance and implications of noise on performance modelling; 3) discuss the terms physical training and training load.

1.2.3.1 Defining physical and competitive performance

To begin with, *physical performance* is defined for the purposes of this thesis as a measured value of a dependent variable (typically denoted p in this thesis) captured from a sporting or biomechanical

task, that typically reflects dimensions of fitness (e.g., strength, power, stamina, speed). In closed sports, such as weightlifting, physical performance (e.g., total weight lifted in the clean and jerk) may also be directly related to the competitive outcome of the event. In contrast, due to the complex nature of sport and in particular team sports, competitive performance is arguably more useful when conceptualised as the interaction of multiple dynamic processes and tasks that terminate with the system state reaching a final outcome (termed here as the competitive outcome). Viewed in isolation, the competitive outcome of a sporting event does not provide much information, and any discernible patterns over time in this type of data cannot be explained without additional context. Therefore, observation and analysis of the interaction of multiple processes comprising competitive performance as defined above has developed into a field of its own within sport science, often called 'performance analysis', concerned with the 'why' of a sporting result in the hope of informing training practices and improving future outcomes (McGarry, 2009). The associated role of the performance analyst is well established as a key component of the backroom staff in professional sport, particularly in team sports such as soccer, basketball, rugby, and American football (Pappalardo and Cintia, 2017; Morgulev, Azar and Lidor, 2018).

1.2.3.2 The measurement of physical performance and implications for performance modelling

A common concept for most but potentially not all physical performance variables in sport science is that there exists an upper limit across all athletes, typically dependent on three interrelated categories of limiting factors (task, environmental, human) (Beneke and Boning, 2008). In most instances, this limit is never accurately known, and can only ever be estimated based on observed data and hypothetical modelling of multiple factors influencing the task (Nevill and Whyte, 2005; Kuper and Sterken, 2007). For example, between 1 second and the current record of 9.58s it is reasonable to assume that there exists an absolute limit on time to run a 100m sprint (Nevill and Whyte, 2005). It is not an individual construct, but rather a level thought to exist globally across all athletes for a specific sporting task. Its main use here is providing a theoretical reference point that says physical performance measured for an individual athlete can range from some relative zero point to this maximum level, and therefore all observations for an athlete will lie between these two points. More important to a researcher or practitioner is the interpretation of observations of physical performance for an individual athlete. In most instances, practitioners measure physical performance with an intention to infer whether change in the athlete's physical capability has occurred because of training. Coaches will typically apply standardised maximal-effort testing protocols to try and identify if change has occurred, that may otherwise be masked if using submaximal effort tasks. However, all maximal effort tests comprise noise made up of instrumentation error and/or the effects of biological variability (Mann, Lamberts and Lambert, 2014; Swinton *et al.*, 2018). Even with standardised conditions and reliable measurement instruments, the influence of normal biological variability makes it unlikely that

a given measurement will reflect a completely an accurate estimate of the latent state of physical capability (Swinton *et al.*, 2018). Therefore, observed measurements from maximal physical performance testing are most likely reflect a Gaussian or non-symmetrical estimate distribution. Skew negative is seemingly more likely as there is a capacity for many reasons such as injury and psychological disengagement to substantively reduce performance, and these may occur semi-regularly. In contrast, levels of physical performance closer to the latent state of maximal physical capability are likely to be a much rarer phenomenon. The implication and importance of this consideration in the context of FFMs and performance modelling is that if models are overfitted to decidedly noisy data, then resultant predictions may be unlikely to generalise the athlete's response to training leading to difficulties in reliably predicting future observations. In other words, the quality of the data is likely to be an important factor in interpreting the validity of FFM research employing a model fitting approach. It follows that researchers interested in performance modelling must carefully plan the type of data they collect (measure) and have a good awareness of what it tells them about the athlete. For example, if objective data (e.g., time, speed) is recorded during sub-maximal tasks, this alone informs very little about what an athlete is typically capable of had they performed a best effort. However, if more information is added such as average velocity during competition, subjective RPE, and/or heart rate monitoring then the chance of accurately inferring physical capability may be improved via comparison with previous data, improving its potential to be applied within a performance modelling framework. In a strength and conditioning environment, the equivalent scenario is a coach recording routine weight training data (e.g., sets, reps, weight lifted) without any additional information that provides insight into the relative difficulty of the task (e.g., reps-in-reserve, barbell velocity, percentage of previous 1RM). These topics are covered in more detail in chapter 2, when measures and suitable data for performance modelling are discussed further. The identification of the effects of measurement error on model predictive accuracy is also investigated in the computational study comprising chapter 4. This subsection has also introduced the term physical capability to the thesis and defined its relationship with physical performance testing. That is, physical performance testing, when applied under maximal effort protocols, can be thought of as the approach to estimating physical capability (thought of as a latent variable).

1.2.3.3 Physical training and training load

Following the definitions of physical performance and capability developed so far, physical training is defined in this thesis as the purposeful execution of exercise-based tasks with the goal of positively influencing or retaining current physical capability. In some cases, particularly within team sport environments where levels of physical capability are already high among athletes, improvement in individual skill and team tactics/dynamics may be the primary concern of training, such that the aspect

of physical training is integrated within simulated game-based tasks and drills primarily targeting technical and tactical factors (Gabbett, Jenkins and Abernethy, 2009).

To conclude this chapter, the term *training load* is reviewed. This term, along with two ancillary terms, *internal* and *external* load with specific definitions, have been used extensively in practice and research (Impellizzeri, Marcora and Coutts, 2019). In relation to physical training, training load has been defined as the "*input variable that is manipulated to elicit the desired training response*" (Coutts, Crowcroft and Kempton, 2017; Impellizzeri, Marcora and Coutts, 2019). Training load has been described as being either external, and/or internal (Impellizzeri, Marcora and Coutts, 2019). In broad terms, external load reflects the physical work completed in the prescribed training session, and is normally specific to the type of training conducted (Impellizzeri, Marcora and Coutts, 2019). For example, in resistance training a measure of external load typically recorded is the load lifted, and in endurance sports coaches may record variables such as distance covered (Impellizzeri, Marcora and Coutts, 2019). Contrastingly, measures of internal load are "*indicators reflecting the physiological or psychological response that the body initiates to cope with the requirements elicited by the external load*" (Impellizzeri, Marcora and Coutts, 2019). One example of internal load for endurance training is heart rate, which responds to the external load as a function of volume (distance) and intensity (speed). Both internal and external measures of training load may be considered in the context of performance modelling, and quantification of training load for input into FFMs and other performance models is discussed in Chapter 2.

1.3 Summary

The purpose of this chapter has been to situate the very specific work of this thesis within the broader contexts of training theory, performance modelling, advances in technology, and the need to appropriately define terms to enhance precision and enable systematic research to follow. The structure of the thesis has also been outlined highlighting the contribution of the work. In addition to the experimental chapters, substantive study and novelty exists in situating and critiquing the most relevant aspects of the performance modelling and FFM literature. Combined with extensive code resources presented in a range of didactic and illustrative formats, this work represents the most comprehensive and self-contained document to date on FFM and it is intended that an individual can follow the work and by the end be well versed in the area with the requisite skills to build and appropriately evaluate models.

Chapter 2: Literature review

2.1 Introduction

FFMs underpin a principal concept guiding the theory and practice of physical training in exercise science (Chiu and Barnes, 2003). The standard FFM posits that a single bout of training creates two antagonistic after-effects including a positive long-lasting and low magnitude fitness effect, and a negative short-lasting and high-magnitude fatigue effect (Banister *et al.*, 1975). It is the combination of these antagonistic components that is said to describe an individual's current level of physical performance over time. In conceptual form, the interaction of fitness and fatigue is frequently used to guide decisions around exercise prescription, recovery, and tapering strategies (Chiu and Barnes, 2003). However, FFMs can also be expressed as parameterised models, with five free parameters in the standard formulation (Banister *et al.*, 1975; Calvert *et al.*, 1976), and upward of eight in more complex representations (Busso *et al.*, 1992; Busso, 2003; Turner *et al.*, 2017; Philippe *et al.*, 2018; Matabuena and Rodríguez-López, 2019). The initial mathematical framework was devised by Banister and colleagues over four decades ago (Banister *et al.*, 1975). Study of the underlying mathematical structure reveals no single model, but instead a collection of related models with similar properties (Banister *et al.*, 1975; Calvert *et al.*, 1976; Busso *et al.*, 1990, 1992; Morton, Fitz-clarke and Banister, 1990; Busso, 2003, 2017; Gouba *et al.*, 2013; Turner *et al.*, 2017; Kolossa *et al.*, 2017; Matabuena and Rodríguez-López, 2019). All FFMs take as input a series of quantified daily training doses and frequently measured performance values and attempt to model - and ultimately predict - performance across time (Banister *et al.*, 1975). Individual tailoring is achieved by setting parameters in the models to match elements such as the magnitude and decay rate of the positive and negative after-effects experienced by the individual. This is achieved by conducting a period of model fitting, where frequent performance measurements are collected, and parameters fitted retrospectively to best match input and output data. If a suitable model can be fit, FFM predictions may assist practitioners with training program design and athlete monitoring (Banister *et al.*, 1975; Schaefer, Asteroth and Ludwig, 2015; Turner *et al.*, 2017; Kumyaito, Yupapin and Tamee, 2018; Connor, Fagan and O'Neill, 2019). Despite the central position of FFMs in traditional training theory and their long history of study by researchers, there has been little uptake across contemporary research and in real world practical environments. This is likely explained by a combination of factors including: 1) inaccessibility of research that assumes readers are proficient with requisite mathematical concepts and have good awareness of historical conventions, nomenclature and terminology; 2) a lack of accessible resources featuring clear methods, and simple tools for model implementation and robust evaluation; and 3) limited empirical research evaluating key aspects of data modelling such as prediction accuracy, model stability, and appropriate methods for parameter estimation. Previous research has summarised various aspects of FFMs and provided a useful entry point for those interested (Chiu and Barnes, 2003; Taha and Thomas,

2003; Hellard *et al.*, 2006; Pfeiffer, 2008; Clarke and Skiba, 2013; Proshin and Solodyannikov, 2018; Rasche and Pfeiffer, 2019). However, prior to a recent review series (see citations in footnote [1,2]) forming part of this work, no up to date resource existed that comprehensively summarised historical developments, methods, and limitations of FFMs. Furthermore, no simple and flexible tools existed to assist researchers fit, evaluate, and systematically investigate FFMs and associated methods using their own data. Researchers have typically developed their own implementations exclusive to their specific needs, limiting uptake and visibility of several potentially useful resources (Pfeiffer, 2008; Schaefer, Asteroth and Ludwig, 2015; Turner *et al.*, 2017). To address this resource gap which represents a substantive barrier to progress, a codebase was developed in the increasingly popular statistical programming language R (R Core Team, 2020). The codebase provides a novel educational resource and scientific toolbox for fitting and evaluating FFMs. This work forms the focus of attention in chapter 6.

The aim of this literature review was to synthesise existing knowledge about FFMs and their potential limitations while balancing technical material with practical considerations relating to the prospective use of the models in sport and exercise science. In this chapter, key mathematical content is blended with applied insights, considerations, and recommendations to present the subject history in a rounded manner. Given the vast quantities of data that practitioners and researchers generate daily, increased usage is required to better develop FFMs that incorporate training design features employed by practitioners and to assess the validity of models by their ability to make useful future predictions under high quality input distributions (data). Training theory and design has developed considerably since Banister *et al*. (1975) developed the first FFM. This increased knowledge, combined with a rapid increase in the volume of data collected by practitioners over the last two decades provides a unique opportunity to accelerate the development of data-driven modelling approaches to assist with individualised prescription and decision making.

2.1.2 Organisation of the literature review

This literature review follows a structure that builds progressively from each prior section in terms of the concepts, research history and thoughts introduced. It is therefore best read the first time in its natural order. There are three primary sections: (2.2) Historical development of the model; (2.3) Application of fitness-fatigue models in research and practice; and (2.4) Model developments and

[1] Stephens Hemingway, B., Greig, L., Jovanovic, M., Ogorek, B., & Swinton, P. (2021). Traditional and contemporary approaches to mathematical fitness-fatigue models in exercise science: A practical guide with resources. Part I. *SportRxiv (Preprint)*. https://doi.org/10.31236/osf.io/ap75j

[2] Swinton, P., Stephens Hemingway, B., Rasche, C., Pfeiffer, M., & Ogorek, B. (2021). Traditional and contemporary approaches to mathematical fitness-fatigue models in exercise science: A practical guide with resources. Part II. *SportRxiv (Preprint)*. https://doi.org/10.31236/osf.io/5qgc2

extensions (state-of-the-art). Section 2.2 starts with a detailed synopsis of the seminal fitness-fatigue model and follows its development into what is referred to in this thesis as the *standard model* and its more generalised form the *general model*. In section 2.3, several basic concepts and methods as they relate to the application of an FFM in research and practice are introduced, including training load quantification (model input), performance measurement (model objective), model parameters (model fitting), and model evaluation (testing). Finally, section 2.4 examines the contemporary period of the historical research body and examines key attempts to address the central limitations of early FFMs using various model extensions. Section 2.4 also informs work conducted in chapter 6 examining the implementation of prospective future methods. A summary is provided at the end of this chapter that reflects on the timeline of FFM research and outlines the scope of key future study of fitness-fatigue models. This literature review, whilst it does not contain any primary research, represents the most complete synthesis of the FFM literature to date whilst incorporating central concepts and practices from the wider fields of performance modelling, statistics and data analysis, much of which is lacking in current FFM research. As such, the literature review and synthesis represent a substantive contribution to understanding of the area of fitness-fatigue modelling and its future direction within sport and exercise science.

2.2 Historical development

2.2.1 The standard fitness-fatigue model

In 1975, Banister and colleagues proposed the first systems model describing athletic performance in terms of training undertaken and fatigue accumulated (Banister *et al.*, 1975; Calvert *et al.*, 1976). The authors' goal was to ultimately develop a predictive model of training that could improve understanding of relevant physiological processes, and in turn support selection of appropriate training programs (Banister *et al.*, 1975). Banister *et al*. (1975) envisioned a complex systems model that could account for several basic determinants of performance such as psychology, skill, and physical capacity. As a first step toward a more complete systems model, the authors isolated training inputs, and then attempted to mathematically quantify their relationship with athletic performance. The product of this initial approach is commonly referred to as the fitness-fatigue model. As noted in the introduction, there is no single model and Banister and colleagues' seminal model - and its common closed form - is referred to herein as the standard FFM. Initially, the authors suggested that a general relationship between physical training and performance could be described by a few general principles including: 1) a moderate increase in physical training followed by a plateau in load will cause performance to rise to a limit approximately 30-50 days later; 2) further rises in training cause further improvements in performance that are proportional to the presently attained performance level; 3) after the continuous cessation of training, performance exponentially decays back to a lower level. It was from these general

principles the authors (Banister *et al.*, 1975) believed the physical response $p(t)$ to training ω_t could be described grossly by the following simple first-order linear differential equation:

$$p'(t) = \omega(t) - \frac{1}{\tau}p(t) \tag{2.1}$$

Although the authors acknowledged that this first-order system (eq. 2.1) would be unable to capture all the known complexity between physical training and performance, they believed a simple first-order system represented a reasonable starting point for further study (Banister *et al.*, 1975). Banister *et al*. (1975) reasoned that individual training sessions could be viewed as impulses, due to their short time duration (i.e. hours) compared to the decay time constants (τ) of the system (i.e. 10s of days) (Banister *et al.*, 1975; Calvert *et al.*, 1976). The abbreviation TRIMP was subsequently coined by the authors to denote these training impulses. Expanding on their original system, it was then suggested that these impulses were responsible for exerting both an immediate positive and an immediate negative training effect that decay at different rates (Banister *et al.*, 1975). As such, performance on a given day was considered to be the linear sum of two antagonistic responses, a (positive) *fitness* response, and (negative) *fatigue* response. Under the model structure, each training session can be thought of as 'dumping' a quantity of positive fitness and fatigue proportional to the training impulse ω; and it is generally assumed that the initial inflow of fatigue is greater than fitness. Immediately, these quantities begin to decay and as outlined below the decay is commonly described by an exponential function with a slower rate of decay for fitness than fatigue (Figure 2.1). The limitation of an impulse-response structure and conflict with conceptual understanding is discussed shortly. At any time t the total amount of fitness, denoted by the function $g(t)$, and fatigue, denoted by the function $h(t)$, is equal to the sum of the remaining quantities for each component. The difference between these positive functions represents model predicted performance $\hat{p}(t)$ which is hoped to reflect actual performance $p(t)$ throughout the training period (eq. 2.2) (Figure 2). Note, hat notation is used to refer to a conditional expectation given past training impulses and the true values of parameters, differing somewhat from the statistical notation of hat to indicate that unknown parameters have been estimated.

$$\hat{p}(t) = p^* + k_g \cdot g(t) - k_h \cdot h(t), \qquad g(t) \geq 0, \qquad h(t) \geq 0, \quad k_h \geq k_g > 0 \tag{2.2}$$

In equation 2.2, parameter p^* acts as an additive term (i.e., a *y*-intercept) of the model and can be interpreted as an estimation of an individual's baseline performance level. Parameters k_g and k_h are the scaling factors that determine the magnitude of the fitness and fatigue realised after a single training session.

The dynamical system at the core of the standard model is specified by the following set of first-order differential equations:

$$g'(t) = \omega(t) - \frac{1}{\tau_g} g(t) \tag{2.3}$$

$$h'(t) = \omega(t) - \frac{1}{\tau_h} h(t) \tag{2.4}$$

Where $\omega(t) \geq 0$, and $\tau_g, \tau_h \geq 0$ are the parameters that describe the rate of fitness and fatigue decay, respectively. Initial conditions of the system are often set at $g(0) = h(0) = 0$ for convenience, and this approximation may be reasonable if the athlete has not engaged in strenuous training for a considerable amount of time. Additionally, if performance measurements are abundant, then a "burn in" period within the fitting process may be used where the cost function is not evaluated for a small initial time-step. Otherwise, these initial conditions may be set and treated as known (preferably based upon previous data), or treated as unknown parameters to be estimated (Busso *et al.*, 1992; Ludwig and Asteroth, 2016). Under the model, larger values of $\omega(t)$ represent greater training doses and therefore a greater stimulus for adaptation. An explicit solution to this linear first-order independent system of differential equations (eq. 2.3, 2.4) is obtained by convolving the training impulse series $\omega(t)$ with exponential functions as follows:

$$g(t) = \omega(t) * e^{\frac{-t}{\tau_g}} \tag{2.5}$$

$$= \int_0^t e^{\left(\frac{-(t-u)}{\tau_g}\right)} \omega(u) du$$

$$h(t) = \omega(t) * e^{\frac{-t}{\tau_h}} \tag{2.6}$$

$$= \int_0^t e^{\left(\frac{-(t-u)}{\tau_h}\right)} \omega(u) du$$

Where in equations 2.5 and 2.6, the symbol $*$ denotes the convolution product. Readers are directed to the appendix A for a complete mathematical derivation. In practice, equations 2.5 and 2.6 are generally approximated by the following Riemann sums:

$$g(t) = \sum_{i=0}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_g}}, \ \Delta_i = 1 \tag{2.7}$$

$$h(t) = \sum_{i=0}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_h}}, \ \Delta_i = 1 \tag{2.8}$$

Where (in equations 2.7 and 2.8) Δ_i is the discrete time step-size, typically set to one day.

Recall that parameters $\tau_g, \tau_h > 0$ are constants setting the rate at which the exponential function decays toward zero. Therefore, their interpretation is related to step size Δ_i in that τ_g, τ_h time-steps bring the fitness or fatigue to roughly $\frac{1}{e}$ of the initial level sans additional training (illustrated in Figure 2). Fitness or fatigue at any given time point can be considered the sum of prior training convolved with the relevant decay function, and therefore the effect of a once-daily training impulse $\omega_i \geq 0$ on performance is described by the following series approximation (Busso *et al.*, 1990, 1992; Morton, Fitz-clarke and Banister, 1990):

$$\hat{p}(t) = p^* + \underbrace{k_g \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_g}}}_{\text{fitness component}} - \underbrace{k_h \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_h}}}_{\text{fatigue component}} \tag{2.9}$$

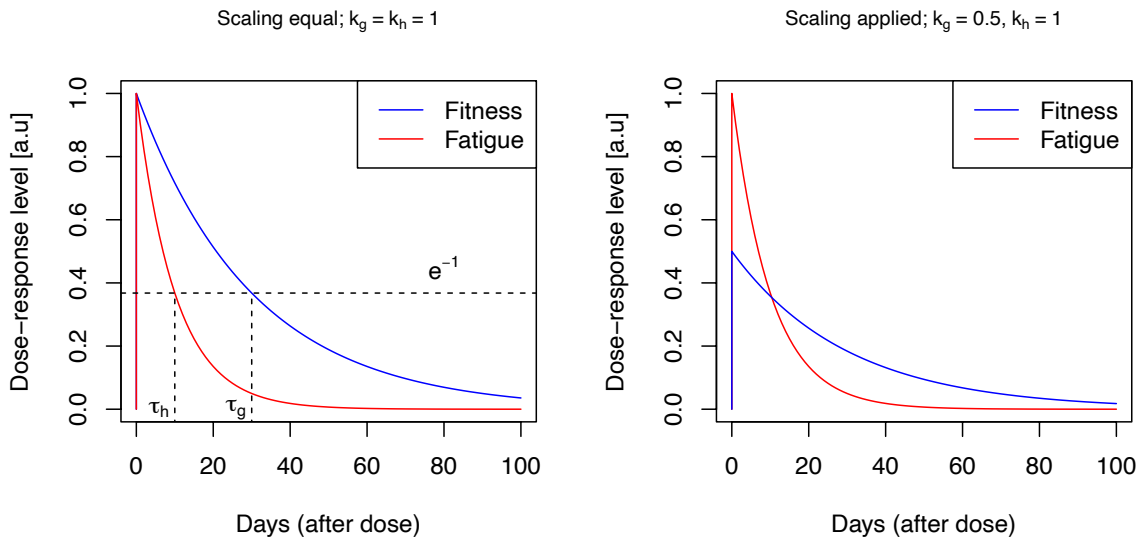


Figure 2.1: Dose-response dynamics of the standard model [3] for a single arbitrary training dose ($\omega = 1$), without scaling (left) and with a scaling ratio ½ acute fitness to fatigue (right). Decay constants $\tau_g = 30, \tau_h = 10$.

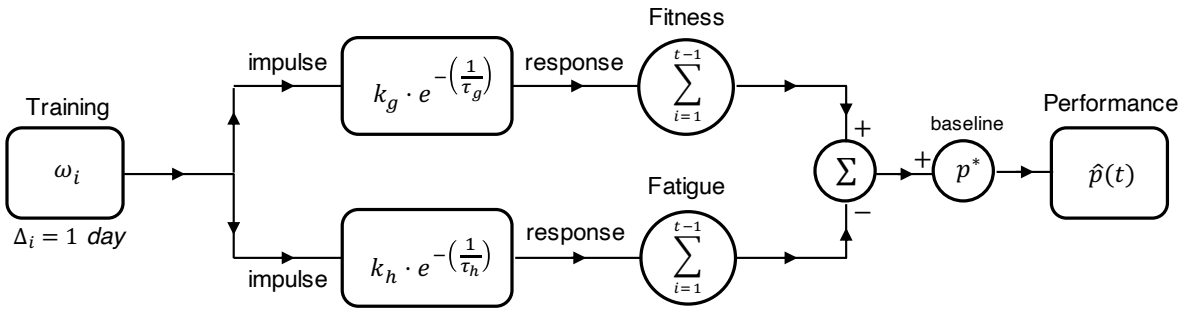


Figure 2.2: Computational flow of the standard model function

[3] **R code for reproducing several of the plots presented in this chapter is available from:** github.com/bsh2/thesis/c2/plots.R

The solution to the model system (eq.'s 2.2 – 2.4), stated in equation 2.9, may also be specified in recursive form as follows:

$$\hat{p}(t) = p^* + \Big(k_g \cdot g(t)\Big) - \big(k_h \cdot h(t)\big) \tag{2.10}$$

$$g(t) = g(t-1) \cdot e^{\frac{-1}{\tau_g}} + \omega_t \qquad g(0) \geq 0. \tag{2.11}$$

$$h(t) = h(t-1) \cdot e^{\frac{-1}{\tau_h}} + \omega_t \qquad h(0) \geq 0 \tag{2.12}$$

The relative magnitudes of the scaling factors $\big(k_g, k_h\big)$ and the decay time constants $\big(\tau_g, \tau_h\big)$ determine the primary dynamics of the model. If a relationship $\tau_g \geq \tau_h \geq 0$ holds, a net positive (or zero) increase in performance is expected over a training cycle provided there is sufficient recovery time. The values of the scaling factors depend on the units used to measure the training load and performance, and some authors have suggested the relationship $k_h > k_g > 0$, or more aptly a ratio of $\frac{k_g}{k_h} < 1$ could indicate a more fatigue-dominated individual (according to model structure), and visa-versa (Morton, Fitz-clarke and Banister, 1990). However, as a single quantity these parameters are harder to interpret physiologically in the same way as the decay constants (i.e., as values representing days), as this requires linking the training units to the performance measure. If both the relationships stated above between the scaling factors and decay time constants are applied as constraints, modelled performance is supressed during and immediately following periods of high training stress until fatigue dissipates sufficiently and fitness effects are revealed (Banister *et al.*, 1975). This model behaviour (commonly referred to as supercompensation) is theoretically consistent with observed effects of high demand training on performance within a normal training cycle, or during short term periods of overreaching. However, these dynamics may not be representative for some groups of elite athletes where training is heavily focussed on maintenance of existing physical performance levels rather than seeking or expecting large increases. Finally, the parameter p^* in equations 2.9 and 2.10 remains the common additive term used across almost all FFMs that can be thought of conceptually as a base level of performance that the individual is not expected to fall chronically below, even after a prolonged absence of training.

The impulse-response dynamics of the standard model in response to a single training dose (as shown in figure 2.1) have been suggested to be an inaccurate conceptualisation of the fitness response to training (Calvert *et al.*, 1976; Busso, 2017; Philippe *et al.*, 2018). Specifically, an unrealistic response of the fitness component obtaining a maximum contribution immediately following training. Noting this limitation, a year after their seminal model (Banister *et al.*, 1975), the same group of authors lead by Calvert presented a simple extension (Calvert *et al.*, 1976) that introduces a delay term within the

fitness component (eq.'s 2.13, 2.14), in an attempt to better match real world understanding of training response. The inclusion of this term creates a period of non-linear growth during the fitness impulse-response, that reaches a peak before undergoing exponential decay (figure 2.3). The fatigue term still retains the standard model dynamics in the authors modified model. The additional term with the fitness component also introduces a scaling effect which is treated shortly. The terms impulse-response, dose-response and load-response are often used interchangeably. Despite being conceptually attractive (Calvert *et al.*, 1976; Morton, Fitz-clarke and Banister, 1990; Rozendaal, 2017), this extension has received limited attention in comparison to the standard model across empirical research. The *fitness-delay* model from Calvert *et al*. (1976) is stated as follows:

$$\hat{p}(t) = p^* + k_g \sum_{i=1}^{t-1} \omega_i \cdot \left(\underbrace{e^{\frac{-(t-i)}{\tau_{g_1}}}}_{\text{effect}} - \underbrace{e^{\frac{-(t-i)}{\tau_{g_2}}}}_{\text{delay}} \right) - k_h \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_h}} \tag{2.13}$$

In Calvert *et al.*'s (1976) model, the load-response function of the fitness component is represented by $g_2(t)$:

$$g_2(t) = \underbrace{e^{-\frac{t}{\tau_{g_1}}}}_{\text{effect}} - \underbrace{e^{-\frac{t}{\tau_{g_2}}}}_{\text{delay}}, \qquad \tau_{g_1} > \tau_{g_2} > 1 \tag{2.14}$$

Although the underlying system of ODE's is not clearly stated in the authors paper (Calvert *et al.*, 1976), it can be determined from the transfer functions (eq. 2.15, 2.16) provided in the original work (Calvert *et al.*, 1976).

Fitness (transfer function)

$$G(s) = \frac{\left(\frac{1}{\tau_{g_2}} - \frac{1}{\tau_{g_1}}\right)}{\left(\left(s + \frac{1}{\tau_{g_1}}\right)\left(s + \frac{1}{\tau_{g_2}}\right)\right)} \tag{2.15}$$

Fatigue (transfer function)

$$G(s) = \frac{K}{s + \frac{1}{\tau_h}} \tag{2.16}$$

From these transfer functions, it is clear that the fitness-delay model (eq. 2.13) arises from a notably different underlying system to that of the standard model. In particular, one involving a second-order

differential equation for fitness and a first-order differential equation (the same as the original model system) for fatigue.

The second-order system from Calvert *et al.* (1976) can be stated as follows:

$$\left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)^{-1} g''(t) = \omega(t) - \left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)^{-1}\left(\frac{1}{\tau_{g_1}}+\frac{1}{\tau_{g_2}}\right) g'(t) - \left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)^{-1} \frac{1}{\tau_{g_1}\tau_{g_2}} g(t) \quad (2.17)$$

$$h'(t) = \omega(t) - \frac{1}{\tau_h} h(t) \quad (2.18)$$

A derivation of the fitness-delay model from the ODE system above is provided in appendix A.

In equation 2.14 (the new load-response for the fitness component), nonlinear growth occurs from $t > 0$ toward a maximum level $\max g_2(t)$ at which point the function begins to decay exponentially back toward zero (as seen in figure 2.3). Time to reach maximum response to a single load under $g_2(t)$ is denoted by $t_{g_2 max}$. The time point $t_{g_2 max}$ can be computed for any values of τ_{g_1}, τ_{g_2} $(\tau_{g_1} > \tau_{g_2} > 1)$ by process of solving $g_2{}'(t) = 0$ for t as follows:

$$g_2{}'(t) = \frac{e^{\left(-\frac{t}{\tau_{g_2}}\right)}}{\tau_{g_2}} - \frac{e^{\left(-\frac{t}{\tau_{g_1}}\right)}}{\tau_{g_1}} \quad (2.19)$$

$$t_{g_2 max} = \frac{\tau_{g_1}\tau_{g_2} \ln\frac{\tau_{g_1}}{\tau_{g_2}}}{\tau_{g_1} - \tau_{g_2}} \quad (2.20)$$

Subsequently, this value can then be used to find $\max g_2(t)\,, t > 0$:

$$\max g_2(t) = e^{-\frac{t_{g_2 max}}{\tau_{g_1}}} e^{-\frac{t_{g_2 max}}{\tau_{g_2}}} \quad (2.21)$$

$$\max g_2(t) = \left(\frac{\tau_{g_1}}{\tau_{g_2}}\right)^{-\left(\frac{\tau_{g_2}}{\tau_{g_1}-\tau_{g_2}}\right)} - \left(\frac{\tau_{g_1}}{\tau_{g_2}}\right)^{-\left(\frac{\tau_{g_1}}{\tau_{g_1}-\tau_{g_2}}\right)} \quad (2.22)$$

As seen in figure 2.3-A on the following page, and from equation 2.14 itself, the addition of a delay term also introduces a natural scaling effect. To see this arithmetically, note (in eq. 2.14) the two terms in the fitness component are of the same form, and differ only by their parameter value, and so as the difference between choice of τ_{g_1} and τ_{g_2} decreases, the magnitude of the peak (unscaled) load response will decrease. This effect can be reversed by scaling the fatigue component response function by $\max g_2(t)$ and then scaling the load value for both components by $\omega / \max g_2(t)$ (shown in Figure

2.3-B). In doing so, this will restore the relationship between the scaling constants k_g, k_h as per the original model.

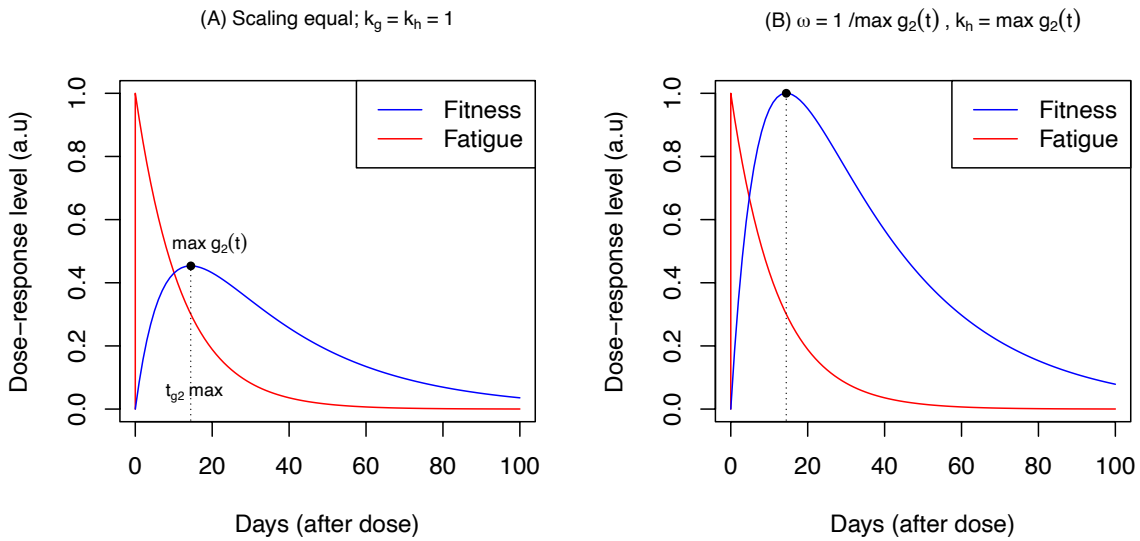


Figure 2.3: (A) Dose-response dynamics of the fitness-delay model for a single arbitrary training dose; **(B)** demonstration of the artificial scaling approach described in the text for reversing the natural scaling effect of the fitness-delay model. Time decay constants $\tau_{g_1} = 30, \tau_{g_2} = 8, \tau_h = 12$. Training dose $\omega = 1$ (left), $\omega = 1/\max g_2(t)$ (right).

Several studies have attempted, with limited success, to link the concepts of fitness and fatigue within FFMs to measurable human processes (i.e., match variation in component traces to variation in observed measurements of known physiological systems). Toward this end, several studies have observed the relationships between variation in model fitness and fatigue traces to variation observed in measured physiological variables over a training period (Taha and Thomas, 2003; Hellard *et al.*, 2006). These variables have included heart rate (Busso, 2003), maximal oxygen uptake (VO_2max) (Corlett, Calvert and Banister, 1978; Busso, 2003; Wood *et al.*, 2005), various respiratory gas parameters (Wood *et al.*, 2005), iron markers (Banister and Hamilton, 1985; Candau, Busso and Lacour, 1992), hormonal fluctuations (Busso *et al.*, 1990, 1992), and enzyme markers (Banister, Morton and Fitz-Clarke, 1992). In some cases, moderate to strong correlations have been observed (a summary table is presented on the following page in Table 2.1). However, there is noticeable inconsistency in these relationships. Although interesting, these results do not necessarily indicate that FFMs are ineffective. Fitness and fatigue are abstract concepts used to model the resultant effect of a range of physiological processes positively and negatively influencing the expression of physical performance. Therefore, perfect correlations with single observed physiological measures should not be expected. Additionally results may indicate the existence of additional factors which is typical of the modelling process (Taha and Thomas, 2003; Hellard *et al.*, 2006; Pfeiffer, 2008).

Table 2.1: Physiological correlates of model fitness and fatigue traces across prior research

Ref	n	Domain	Duration	Physiological Measures	Fitness Component	Fatigue Component
(Corlett, Calvert and Banister, 1978)	1	Lab-based general exercise	4 months	Serum Bilirubin & Maximal Oxygen Uptake	Maximal oxygen uptake demonstrated moderate correlations with traces of the fitness component (visual plot only, no quantitative data provided)	Bilirubin demonstrated moderate correlations with traces of the fatigue component (visual plot only, no quantitative data provided)
(Banister and Hamilton, 1985)	5	Distance Running (Amateur)	300 Days	Serum Iron	No data provided	All markers of iron status (Ferritin, RBC and Haemoglobin) appear to demonstrate low-moderate correlations with fatigue trace. Inter-individual variation evident, with correlations likely stronger for some individuals than others
(Busso et al., 1990)	6	Weightlifting (Elite)	1 year	Serum testosterone (T), cortisol (C), SHBG as well as ratios between T:C and T:SHBG	[Testosterone, $r = 0.61 - 0.92$] (significant for n=2); [Cortisol, $r = -0.75 - 0.76$]; [SHBG, $r = -0.64 - 0.75$]; [Testosterone : Cortisol, $r = 0.11 - 0.94$] (significant for n=2); [Testosterone : SHBG, $r = -0.27 - 0.89$] (significant for n=1)	[Testosterone, $r = 0.34 - 0.88$] (significant for n=1); [Cortisol, $r = -0.75 - 0.75$]; [SHBG, $r = -0.75 - 0.76$]; [Testosterone : Cortisol, $r = 0.0 - 0.97$] (significant for n=1); [Testosterone : SHBG, $r = -0.80 - 0.92$] (significant for n=1)

Ref	n	Domain	Duration	Physiological Measures	Fitness Component	Fatigue Component
(Banister, Morton and Fitz-Clarke, 1992)	2	Cycling (Amateur)	28 days	Serum biomarkers and enzymes including: Lactate dehydrogenase, creatine kinase, and aspartate transaminase	No graphs or data reported for correlations between any measured physiological marker and fitness parameter	Fatigue and LDH showed a degree of correlation across all subjects, however these would likely be poor-moderate Temporal trends out of phase with LDH generally lagging behind observed changes in fatigue. Similar trend apparent from fatigue and creatine kinase. Greatest correlation appears to be between fatigue and aspartate transaminase with much less variation between subjects and data points fitting much closer. Likely moderate-strong.
(Busso et al., 1992)	6	Weightlifting (Elite)	6 weeks	Serum testosterone, luteinising hormone, SHBG and T:SHBG ratio	Mean Serum Luteinising Hormone ($r = 0.97$) during training period 1 [appears to be for all subjects]	Nothing reported for fatigue parameter in either table, graph or written format. Same data set utilised in Busso 1990, however serum LH results not reported in earlier work.
(Wood et al., 2005)	1	Running (Amateur)	12 weeks	Velocity at ventilatory threshold (VTRS), Profile-of-mood-states (POMS) and VO_2Max, Running economy	[VTRS, $r = 0.94$]; [VO_2Max, $r = 0.42$]; [Running Economy, $r = -0.61$]	[POMS, $r = 0.75$]

2.2.2 The general model

Previous authors have attempted to examine the potential of FFMs to model complex physiological phenomena via a limited number of resultant components, investigating the statistical adequacy of different levels of FFM complexity within their experimental studies (Busso, Carasso and Lacour, 1991; Millet *et al.*, 2002, 2005; Busso, 2003, 2017; le Bris *et al.*, 2006b, 2006a; Agostinho *et al.*, 2015; Philippe *et al.*, 2015). Toward this end, the standard FFM can be specified in a general (closed) form such that the working of the system is reflected by N additive first-order components (Busso, Carasso and Lacour, 1991). In line with the process for solving the standard system (eq. 2.5-2.8, appendix A), the discrete solution of the general model with N components is obtained by convolving the variable representing the time-series of training loads ω individually with N exponential functions of the form $e^{-(t/\tau_r)}$ where $(r = 1, 2, \dots, (N-1), N)$, and then taking their sum. The resultant discrete function representing the solution to the general model is then written:

$$\hat{p}(t) = p^* + \sum_{r=1}^{N} \left(k_r \cdot \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_r}} \right) \tag{2.23}$$

The general scaling constant k_r in (eq. 2.23) is taken as positive or negative dependent upon whether the underlying system response represented by the associated component results in a performance increase or decrease. Several studies have applied the general model with experimental data to assess different levels of model complexity as a function of additional components, using laboratory study and/or retrospective analyses of field data (Busso, Carasso and Lacour, 1991; Candau, Busso and Lacour, 1992; Millet *et al.*, 2002, 2005; Busso, 2003; le Bris *et al.*, 2004, 2006b; Agostinho *et al.*, 2015; Philippe *et al.*, 2015). Several of these studies have found that additional first-order components did not significantly increase model fit in the majority of subjects (Busso, Carasso and Lacour, 1991; Millet *et al.*, 2002, 2005; Busso, 2003; le Bris *et al.*, 2006b, 2006a; Philippe *et al.*, 2015). However, a large proportion of these studies have been limited by sample size, the population (i.e., limited study of elite performers) and an inadequate number of data points, advantaging models with a low number of components. In contrast, none of the studies listed above assessed model prediction accuracy using out-of-sample data or in the context of external training interventions that may change the input distribution (Pearl, 2010). Therefore, it remains difficult to identify the extent of overfitting and whether increasing model complexity provides a predictive advantage. A lack of out-of-sample assessment is not exclusive to studies assessing different levels of model complexity but is a consistent limitation throughout research investigating FFMs and is discussed further in section 2.3.4. It is recommended that future research applying the standard model attempts to assess further levels of model complexity where possible, to identify prediction differences (particularly out of sample) across diverse scenarios. Nevertheless, some authors have suggested that a simple two-component model (i.e.

the standard model, or the fitness-delay model) is sufficient on the basis that it is likely to provide better conceptual and observed agreement with supercompensation phenomena that occurs during the recovery period in athletes that take part in regular training (Busso, Carasso and Lacour, 1991).

Researchers have also previously considered the case where an FFM is fit to an individual who has recently trained, or is currently undergoing training and therefore residual fitness and fatigue effects are expected to carry over into the next modelling period; i.e. $g(0) \neq 0, h(0) \neq 0$ (Busso *et al.*, 1992; Ludwig and Asteroth, 2016). To adjust for this, the addition of P exponentially decaying initial components into the general model structure was suggested as a suitable approach by Busso and colleagues (Busso *et al.*, 1992), as follows:

$$\hat{p}(t) = p^* + \sum_{r=1}^{N}\left(k_r \cdot \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_r}}\right) + \sum_{p=1}^{P} q_p^0 \cdot e^{\frac{-t}{\tau_p}} \tag{2.24}$$

Where the additional parameters (associated with q_p^0) denote the initial level of each component, and τ_p is the decay time constant on the corresponding initial level. Conceptually, each additional parameter can be thought of as an additional 'extra-large' load-response at time $t = 0$, the sum of which will act to translate the function up or down on the vertical axis for a short initial period depending on the magnitudes with each associated component (i.e., fitness or fatigue). Note that not every component may require an initial decaying state (Busso *et al.*, 1992). And, as outlined earlier, the use of a "burn-in period" (alternately named a "grace" period) where early observations are discarded is a technique used in Bayesian computation and may be suitable (Hamra, MacLehose and Richardson, 2013). This function is provided alongside methods such as Maximum Likelihood Estimation within libraries for popular modelling languages such as Python (`statsmodels` module) and R (Seabold and Perktold, 2010).

Only a subset of these general FFM variations have been implemented across the literature. In one experimental study, Busso and colleagues found that the number of statistically significant training components for each athlete ranged from 1-component ($n = 1$) to 2-component's ($n = 5$) (Busso *et al.*, 1992). The number of initial components (IC) in the final model equation were 0-IC ($n = 2$), 1-IC ($n = 2$), and 2-IC ($n = 3$). However, even in-sample R^2 values in the study ranged substantially from 0.29 to 0.85 and statistical analyses were limited by low numbers of data points making it difficult to draw meaningful conclusions. The approach was further explored with regard to improvement in prediction accuracy by Ludwig and colleagues in 2016, and was referred to as an a posteriori 'pre-load' function (Ludwig and Asteroth, 2016). However, *p*-values may not be the most appropriate approach to assessing the benefit (or not) of additional components for in-sample data, and out-of-

sample assessment must be considered in future. An increase in model complexity (i.e., the addition of further filters increasing the number of free parameters) may be used to capture more known phenomena of the training response. However, it is worth noting that models with large numbers of free parameters are at a higher risk of overfitting, in particular without a suitable method to assess out-of-sample prediction accuracy, and the requirements on the training data (i.e., the number of performance measures) may quickly become impractical. These issues are treated in more depth in sections 2.3.2 and 2.3.3.

2.2.3 A variable baseline performance model

FFMs generally include an intercept parameter p^* (baseline performance) which is conceptualised as the performance value obtained after a substantive period of physical recovery such that the residual effects of all previous training sessions have dissipated. However, this perspective is limited as in many cases baseline performance may change between training periods. The implications of this are not reflected in the standard model structure, but how problematic this is will depend on the model's use. For example, if a coach or practitioner wishes to use the standard model to predict short term change prior to a tapering and peaking phase, change in the baseline constant may not have occurred and prior change will have already been factored within the model fitting (estimation) process. However, if a researcher carries over the parameters during a long-term monitoring period, baseline performance may require re-estimation, unless expected change in the parameter itself can be predicted. Gouba *et al.* (2013) suggested an alternative approach to address this limitation by replacing the additive term p^* with a recursive term (eq. 2.25). The basis of this extension is that performance on day t is influenced by both the previous value of measured performance, and the number of training doses between them (Gouba *et al.*, 2013). Their model can be stated as follows:

$$\hat{p}_{t_j} = p_{t_{j-1}} + k_g \sum_{i=t_{j-1}}^{t_j-1} \left(e^{-\frac{(t_j-i)}{\tau_g}} \cdot \omega_i \right) - k_h \sum_{i=t_{j-1}}^{t_j-1} \left(e^{-\frac{(t_j-i)}{\tau_h}} \cdot \omega_i \right), \qquad j \geq 1, \Delta_i = 1 \quad (2.25)$$

Where j is the number of measured performances, t_j is the day of the j-th measured performance, and $p_{t_{j-1}}$ is the observed performance value on t_{j-1}.

Under experimental intervention data, the authors observed poor model fit for both the standard FFM (eq.2.9) ($R^2 = 0.28$) and their own novel extension (eq.2.25) ($R^2 = 0.38$). The authors had attempted to model the data of an elite level monofin swimmer. Little can be drawn from these results due to the confounding effects of low measurement frequency ($n = 6$ over a 24-week period), and a lack of out-of-sample data to test the fitted models for any practical differences in future predictions. Therefore, more research applying this model extension is still needed to assess whether there exists any substantive improvement in accuracy and usefulness in practice.

2.2.4 Influence Curves

Influence curves were first introduced to the literature by Morton and colleagues in 1990 (Morton, Fitz-clarke and Banister, 1990), and then were further examined by Fitz-Clarke *et al*. (1991) a year later as a helpful method to demonstrate how a unit of training impulse at a general time t will effect performance at a specific future time t_p under certain conditions. Earlier, when the fitness-delay modification to the standard model (eq. 2.13) was introduced, these concepts were loosely examined in terms of the formulae for time to maximum performance and maximum performance (peak) following a single training dose (see eq.'s 2.19, 2.22, respectively). This subsection will briefly survey the small volume of literature on influence curves here as their theory is insightful for understanding underlying dynamics of FFMs and they are often mentioned in other related works. As has previously been discussed, model performance for a given day t_p is the summation of the previous day's training impulses up to $t_p - 1$ each decayed over the time between the impulse and t_p. Recall also, that the magnitude of the training impulse determines its contribution to performance through the scaling factors (k_g, k_h) for fitness and fatigue, respectively.

The original derivation of an influence curve from Fitz Clarke et al. (1991) is given as:

$$\begin{aligned} p(t_p) &= k_g \cdot g(t_p) - k_h \cdot h(t_p) \\ &= \int_0^{t_p} \left[k_g \cdot e^{\frac{-(t_p - t)}{\tau_g}} - k_h \cdot e^{\frac{-(t_p - t)}{\tau_h}} \right] \omega(t) dt \\ &= \int_0^{t_p} L(\mu)\omega(t) dt \end{aligned} \tag{2.26}$$

Where the influence curve is then defined by:

$$L(\mu) = k_g \cdot e^{\frac{-\mu}{\tau_g}} - k_h \cdot e^{\frac{-\mu}{\tau_h}} \tag{2.27}$$

Where μ is given by $t_p - t$, and therefore represents the time prior to the performance date t_p for a given dose. The multiplication of the training impulse series by the influence curve $L(\mu)$ gives you a product function, which when integrated represents the performance at t_p. In other words, the area of this product function represents the performance at time t_p (Fitz-Clarke, Morton and Banister, 1991). As the function $\omega(t)$ is typically a discrete series of daily impulses (with time step $\Delta = 1$) the integral becomes a summation via principles of convolution, and is written:

$$\int_0^{t_p} L(\mu)\omega(t)dt = \sum_{i=1}^{t}\left[k_g \cdot e^{-\frac{(t-i)}{\tau_g}} - k_h \cdot e^{-\frac{(t-i)}{\tau_h}}\right] \cdot \omega_i \cdot \Delta_i$$

$$= \sum_{i=1}^{t} L(\mu_i) \cdot \omega_i \cdot \Delta_i \qquad (2.28)$$

The function $L(\mu)$ is independent to the training load series and only uses model parameters. Influence curves may allow identification of a time point (denoted t_r) at which training load should be reduced prior to competition to maximise performance (Morton, Fitz-clarke and Banister, 1990; Fitz-Clarke, Morton and Banister, 1991; Borresen and Lambert, 2009). This point t_r is defined as the *critical time* point (Fitz-Clarke, Morton and Banister, 1991) at which each training impulse will contribute more to the model fatigue component than fitness. In other words, if training load is sustained or increased beyond this time point, performance is likely to decrease significantly by time t_p. Another time point, t_g, was defined as the *period of maximum opportunity* and is described as the period in which previous training will be most influential or positive for performance on a specific day t_p under the model structure. It follows, if used correctly influence curves may provide a unique method to assist in informing a peaking and tapering strategy for a specific future performance (Morton, Fitz-clarke and Banister, 1990; Fitz-Clarke, Morton and Banister, 1991; Borresen and Lambert, 2009). However, these methods rely heavily on the assumption of an FFM that provides characteristic and strong deterministic response profiles for each athlete, which may be unreasonable over long time periods and with the standard formulations. Nevertheless, the methods of influence curves are still insightful as they assist in exploring the complexities of planning training via quantitative methods with respect to single or multiple key future performances. In particular, influence curves help to highlight in simplistic cases the challenges of planning for optimal performance in the presence of multiple tightly spaced competitive events, as the optimal training and tapering period for a single performance is likely to influence subsequent ones (Fitz-Clarke, Morton and Banister, 1991; Borresen and Lambert, 2009). This is another example of the lack of independence between real world events (training and performance) common in sport, that contributes to making modelling human processes such a challenging endeavour. We briefly explore these time points t_r and t_g mathematically and graphically below.

The critical time point t_r is the point at which $k_g \cdot g(t_r) = k_h \cdot h(t_r)$, or $L(\mu) = 0$ (Fitz-Clarke, Morton and Banister, 1991). Thus, at this point a reduction or rest from training is warranted prior to competition, to avoid overwhelming fatigue effects within the performance response to training. Critical time (t_r) is calculated directly from the individualised model parameters, as follows:

$$t_r = t_p - \mu = t_p - \left(\frac{\tau_g \tau_h}{\tau_g - \tau_h} \ln\left(\frac{k_h}{k_g}\right)\right) \tag{2.29}$$

The second of point of interest, the period of maximum opportunity t_g is defined as the point when $L(\mu)$ is the greatest, or $\frac{dL}{d\mu} = 0$. This point is then calculated by

$$t_g = t_p - \frac{\tau_g \tau_h}{\tau_g - \tau_h} \ln \frac{k_h \cdot \tau_g}{k_g \cdot \tau_h} \tag{2.30}$$

For a detailed derivation of these formulae, and underlying algebraic arguments, readers are directed to the appendices of Fitz-Clarke *et al.* (1991) which contains the step-by-step presentation. Figure 2.4 is a plot demonstrating the time points t_g (period of maximum opportunity) and t_r (critical time) according to influence curves for a dummy set of parameter estimates from the original study (Fitz-Clarke, Morton and Banister, 1991) ($\tau_g = 45, \tau_h = 15, k_g = 1, k_h = 2$) and a day of interest $t_p = 60$. According to model dynamics, Fitz-Clarke et al. (1991) demonstrated that maximal performance (with respect to the criterion performance value) may be achievable only once in a season according to model dynamics, due to the rest required to adequately taper and peak, which also represents lost time for subsequent training events (Fitz-Clarke, Morton and Banister, 1991). Therefore, optimisation of performance for multiple events across a season with respect to a FFM requires compromise, and the FFM and simulating influence curves may be a useful approach to ascertain how a compromise might be met within the structure of a training program. Furthermore, techniques of influence curves could be used more creatively within model-fitting processes to penalise parameter sets that generate future predictions that violate basic assumptions or expectations (in terms of rate of change, time to peak growth, etc), regardless of their in-sample model fit.

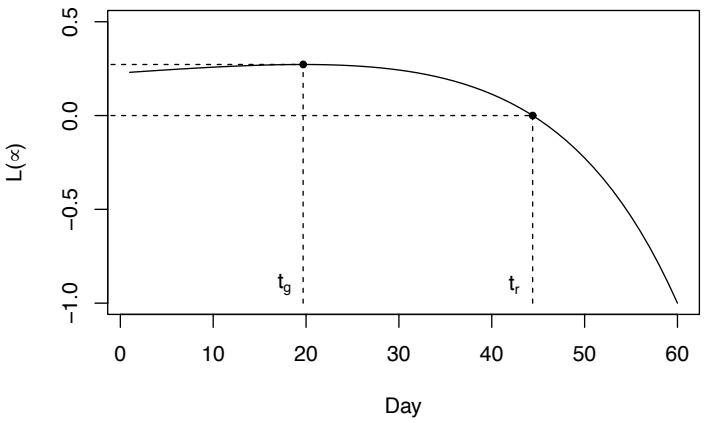


Figure 2.4: Plotting the influence curve to identify the period of maximum opportunity (t_g), and the critical time point (t_r) at which training should be reduced to avoid an overwhelming influence of fatigue effects under the standard model dynamics.

2.2.5 Further developments to model structure

Three key limitations have been identified with the basic FFMs presented in previous sections including: 1) irrespective of training dose presented in any single session, performance is maximised a fixed number (η) of days later; 2) performance can be arbitrarily increased by simply increasing the training dose (Hellard *et al.*, 2006); and 3) there is no interaction between training sessions, and therefore training performed on a given day does not influence the response generated from another session. Collectively, these limitations indicate that the best training plan (under the standard model specification) would be to complete all the planned training in one session, η days prior to competition. The general approaches proposed to address these limitations include either constraining or saturating the model input (i.e., the training dose) (Hellard *et al.*, 2005) or directly manipulating the model formulation (Busso, 2003; Kolossa *et al.*, 2017; Philippe *et al.*, 2018; Matabuena and Rodríguez-López, 2019). For example, the use of an external threshold saturation function to restrict the effective load without changing the relationship between the input and output specified has been proposed (Hellard *et al.*, 2005), along with the introduction of specific structural modifications to account for non-linearities (Turner *et al.*, 2017) and interactions between training sessions (Busso, 2003; Kolossa *et al.*, 2017; Matabuena and Rodríguez-López, 2019). The theory behind these approaches including the use of a Kalman filter is presented in section 2.4 of this literature review. Furthermore, proposed extensions within the literature include exponential growth kinetics (Philippe *et al.*, 2018) and secondary-signal models (Busso, 2017); and these are also examined. In chapter 6, several resources are presented that deal with practical implementation of these advanced approaches in the R programming environment (R Core Team, 2020), for applications in future research beyond the scope of this thesis.

2.3 Application of fitness-fatigue models in research and practice

Fitness-fatigue models depend on constant, or in special cases time-varying (Busso *et al.*, 1997; Kolossa *et al.*, 2017), parameter values that cannot be inferred through observation and must instead be estimated from training and performance data, for example, by the method of nonlinear least squares (Hellard *et al.*, 2006; Soetaert and Petzoldt, 2010; Clarke and Skiba, 2013; Transtrum *et al.*, 2015). FFMs are nonlinear in their parameters and therefore the model fitting process constitutes a nonlinear optimisation problem. The process takes as input a time-series of measured performance and training load values, and provides as output, model parameter estimates that give good (preferably the best possible in some sense) agreement between iteratively computed model performance values and the measured performance data. In summary, to fit an FFM a researcher or practitioner requires a set of suitable training quantification data, performance measurements, and a method to alter the model

parameters to best match these through an optimisation process. Addressing these three requirements is discussed in this respective order in the following sections.

2.3.1 Training load quantification

Existing FFMs require a time-stepped input in the form of a discrete time-series of training load values $\omega_i \geq 0, \Delta_i \in \mathbb{N}$, where a single number represents the training load for the time-step chosen. Typically, the time-step will be set to one and each numeric value intended to represent the resultant effect of the daily training session(s) on the physical response being modelled. Reducing athletic training sessions to numeric values is a highly complex and unresolved problem within fitness-fatigue modelling, particularly with respect to the broad spectrum of training modalities and performance measures. Within laboratory settings, training load quantification may theoretically be easier, due to the tightly controlled nature of a training response study and the use of single training modalities and performance measures that are expected to map directly to the intervention strategy (Busso and Thomas, 2006). However, this is rarely practical in the real world, where the necessary training often comprises different physical capabilities trained concurrently, and adaptive potential may suffer from a number of interacting factors (e.g., environmental, psychosomatic, social). Proposed solutions to address the problem of training load quantification thus far can be described as predominantly exploratory, typically resulting from attempts to identify training variables that are believed to account for the largest variation of training response in a population. The result of these attempts are usually an amalgamation of assumptions, coaching experience, and empirical knowledge to construct and implement one or more methods that seem reasonable but are not extensively validated. The full scope of the training load quantification problem is too extensive to address in this review and requires substantial development as both a standalone area and within the context of fitness-fatigue modelling. However, interested readers are directed toward previous reviews of this topic (Banister, Carter and Zarkadas, 1999; Borresen and Lambert, 2009; Hayes and Quinn, 2009; Jobson *et al.*, 2009). In the following sections an overview of the area in the context of fitness-fatigue modelling is provided.

A basic and general form for quantifying session training load (ω) using a volume-by-intensity approach may be written mathematically as:

$$\omega_i = \mathcal{D}_i \cdot u(I_i) \tag{2.31}$$

Where $\mathcal{D}$ represents the density of the exercise (e.g., volume, duration, distance), I is the intensity (e.g., percent-of-maximum, perceived intensity) and u is a function - potentially non-linear - that can create relatively large changes in training dose with relatively small changes in intensity (this schema is illustrated in Figure 2.5).

As described in Moxnes and Hausken (2008), different performance measures are likely to require different weighting functions based on the understood principles of specificity vs. generality for the effect of exercise on performance. The interaction between density and intensity to provide appropriate training load values may vary between individuals and the type of training undergone relative to the underlying performance measure (Moxnes and Hausken, 2008). However, whether the implications of any differences are significant enough to consider relevant in practice still represents an open problem in the context of fitness-fatigue modelling. Considering the principle of training specificity further, any training load quantification function needs to also scale for differences in the magnitude of response due to the type of training undertaken. For example, an endurance athlete is likely to include both high-volume low-intensity and high-intensity low-volume training in varying proportions within any given training program. At a basic level, these two types of training sessions will be structured to take advantage of improvements in different underlying energy systems and physiological processes. The effect of these adaptations on performance as expressed in a measured setting will therefore will vary depending on the properties of the performance measure selected (e.g., VO_2max or 5k Time-trial (TT) vs. an aerobic threshold test (AeT)), reflected in different magnitudes ascribed to the quantified dose. Athlete level and prior training history are also likely to be required factors that adjust the magnitude of response in any training load quantification method. Therefore, it is very reasonable to suggest that further work is required in this area to develop training load quantification methods that are sensitive to (at minimum) the differences described. In study of alternative performance models some prior attempts have been made to address this issue, for example the PerPot DoMo model which isolates intensity and volume into two concurrent load flows (Perl and Pfeiffer, 2011). In fitness-fatigue modelling, different input for each component (i.e., $\omega_{fit}, \omega_{fat}$) may be required to take into account the effects of certain exercises on systematic fatigue but not fitness in relation to the criterion performance measure. For example, as a basic example considering resistance training, an exercise such as the squat may generate a systemic fatigue response affecting physical performance in the bench press, but not evoke a useful fitness response in relation to measurement of a one-repetition maximum.

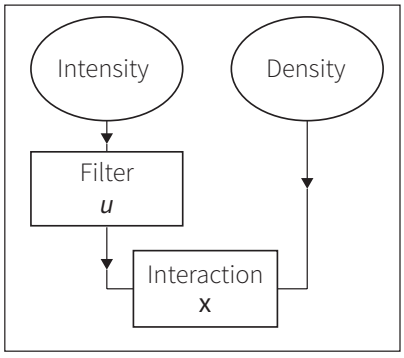


Figure 2.5: A generalisable framework for quantifying training load for use in modelling physical performance change

Variations of the density-intensity approach have featured in the majority of previous fitness-fatigue model applications, in swimming (Banister *et al.*, 1975; Calvert *et al.*, 1976; Mujika *et al.*, 1996; Hellard *et al.*, 2005, 2006; Thomas, Mujika and Busso, 2008; Ishii *et al.*, 2008; Chalencon *et al.*, 2012; Gouba *et al.*, 2013; Chalencon, 2015; Mitchell *et al.*, 2020), cycling (Busso, Carasso and Lacour, 1991; Busso *et al.*, 1997, 2002; Busso, 2003; Clarke and Skiba, 2013), weightlifting (Busso *et al.*, 1990, 1992), judo (Agostinho *et al.*, 2015), gymnastics (Sanchez *et al.*, 2013), hammer throwing (Busso, Candau and Lacour, 1994), and running (Wood *et al.*, 2005; Suzuki *et al.*, 2006; Wallace, Slattery and Coutts, 2014). In 1985, Banister and colleagues introduced Banister's TRIMPs, which included a ratio to quantify intensity via elevation in heart-rate relative to basal and maximum (Banister and Hamilton, 1985):

$$\mathcal{D}_i \times u\left(\frac{[HR_{exercise} - HR_{basal}]}{[HR_{maximum} - HR_{basal}]}\right) \tag{2.32}$$

Where in equation 2.32, $\mathcal{D}_i$ is the duration (in minutes) on day i, u is a non-linear function of the exercise intensity $0 \leq I \leq 1$ which is the fractional elevation of the maximal HR, and $u(I) = 0.86 \cdot e^{1.67 \cdot I}$ (coefficients derived for female athletes in the original study). Therefore, u weights brief efforts at higher heart rates positively, relative to sustained steady state effort at lower heart rates.

Banister's TRIMPs has been used in several applications of fitness-fatigue modelling in running (Banister and Hamilton, 1985; Morton, Fitz-clarke and Banister, 1990; Banister, Morton and Fitz-Clarke, 1992; Wallace, Slattery and Coutts, 2014), skiing (Candau, Busso and Lacour, 1992), swimming (Gouba *et al.*, 2013), triathlon (Zarkadas, Carter and Banister, 1995; Banister, Carter and Zarkadas, 1999; Millet *et al.*, 2002), and cardiac rehabilitation (le Bris *et al.*, 2004, 2006b, 2006a). Although Banister's TRIMPs and similar derived approaches to training load quantification (e.g. session RPE, summated heart rate zone score, and Lucia's TRIMPS) (Lucía *et al.*, 2003; Borresen and Lambert, 2009) comprise the majority of prior research applications, other less known methods may provide unique avenues for further study. These include adapting models to incorporate multiple inputs as outlined earlier (e.g. an input specific to each model component, which may also reduce observed inter dependency between parameters k_g, k_h), and feature selection within a statistical-learning framework (Rozendaal, 2017). While there is no universal method to quantify training load series across the multiple possible applications of fitness-fatigue modelling, and work remains to be completed in this area, reasonable attempts to quantify training load can and should still be made.

From the perspective of fitness-fatigue modelling, care should be taken to ensure that the process of quantifying a full training session to a single value incorporates elements of the training that are most likely to map to the performance outcome measure. Furthermore, as a basic pre-requisite to deriving a new training load quantification method, or utilising existing approaches, researchers and practitioners are recommended to inspect the quality of their data with respect to measurement frequency,

measurement error, recording accuracy, and the breadth of training data recorded (Rasche and Pfeiffer, 2019). Furthermore, researchers and practitioners should seek to record data at the highest possible frequency within-session, reduce instrumentation error and biological variability where possible (Swinton *et al.*, 2018), and consider recording data from associated training or mechanical variables that may reveal further information or provide higher resolution. For example, subjective variables such as repetitions-in-reserve and RPE (rating of perceived exertion) may be used to provide high-frequency mapping of intensity across a training-session. Similarly, objective measures such as heart rate (Williams *et al.*, 2018; Mitchell *et al.*, 2020), barbell velocity, and accelerometer/positional data that can be sampled with little obstruction at high frequencies may also provide insight into session intensity over time, and in particular the relative difficulty of the task compared to previous efforts. As discussed in the introduction, this these measures reflect common variables collected by practitioners on a daily basis, particularly as technology gets more affordable and thus widespread across different levels of practice.

2.3.2 Criterion performance selection

The second major consideration for researchers and practitioners wishing to apply FFMs with their own data is the selection of an appropriate outcome measure (i.e., model target or objective). This outcome measure is commonly referred to within the literature as the *criterion performance* measure. The use of the term *performance* in sport is broad and for the purposes of fitness-fatigue modelling benefits from being conceptualised as either physical or competitive performance. Physical performance was defined in the introduction as the state of any dependent variable that describes physical capability in a sporting or biomechanical task, and typically reflects dimensions of fitness (e.g., strength, stamina, speed). In contrast, competitive performance is composed of the interaction of multiple dynamic, stochastic, and chaotic processes (e.g., physical performance, psychological state, team and opposition behaviour, environment). These processes eventually converge to a conclusive outcome variable (win, loss, draw), which we separated in chapter 1 as the *competitive outcome*. Viewed in isolation, competitive outcomes may fail to provide useful information, and any discernible patterns over time are very unlikely to be explained by concepts of fitness and fatigue alone. Therefore, future use of performance models such as FFMs, particularly within team sport environments, may require a shift in emphasis where training data is used to predict response in terms of dimensions of physical capability measured through common exercises (e.g. one-repetition maximum squat, vertical jump height, or peak power / impulse produced during an explosive movement) rather than sporting outcomes which are likely to demonstrate complex and often confounded relationships with training loads. Exceptions to this may exist in closed skill sports, such as track and field or power/strength-based sports such as weightlifting where outcomes in competitive events relate closely to physical capability (e.g., 100m swim sprint, 5k time-trial run or cycle, maximum weight lifted by a powerlifter

in the squat) (Wang *et al.*, 2013). However, this data is only likely to supplement a modelling approach to increase the number of data points, rather than be the focus of it, due to introduction of other factors such as those highlighted above. Previous examples of criterion performance measures used in FFM applications include performance in mock or actual sporting events particularly within endurance sports (Zarkadas, Carter and Banister, 1995; Hellard *et al.*, 2006; Mitchell *et al.*, 2020), maximum strength or peak power output produced during specific exercises (Busso, Candau and Lacour, 1994; Busso, 2003), and outcomes obtained from clinical exercise capacity tests (le Bris *et al.*, 2004, 2006b, 2006a).

Measurement frequency and underlying measurement error should also be considered when selecting an appropriate performance measure and how these may influence the ability to obtain stable and accurate parameter estimates (Rasche and Pfeiffer, 2019; Stephens Hemingway *et al.*, 2019). It has been highlighted that missing data may lead to non-interpretable model behaviour in iteratively updated (i.e. continuous) modelling applications (Rasche and Pfeiffer, 2019). For multiple linear regression, a heuristic of a minimum of 15 observations per-parameter is recommended (Stevens, 1986). However, the FFM comprises a non-linear function (in its parameters) with subsequent statistical analyses based on asymptotic theory (Bates and Watts, 1988; Sen and Srivastava, 1990; Davidian and Giltinan, 2003). Therefore, more data points are required per model parameter (Hellard *et al.*, 2006) with previous recommendations of 60-200 performance tests over a period of model fitting. As a result, the ability to accurately model and predict performance is expected to require performance tests that can be completed at a high weekly frequency (e.g. at least every 1-3 days) (Stephens Hemingway *et al.*, 2019). Consequently, it is likely that only non-fatiguing performance tests (for example, the counter-movement jump) or those without persistent learning effects may be able to meet this requirement. An alternative strategy that has clear potential but has received limited investigation in the context of fitness-fatigue modelling is the extrapolation of performance measures from standard training session data (Al-Otaibi, 2017). Such procedures could incorporate either objective (e.g., barbell velocity or measurements of power) or subjective variables (e.g., repetitions in reserve) to predict maximum performance with high frequencies. Combined with the previous recommendation of a shift to emphasise physical performance in simple physical tasks or exercise movements, effective use of training monitoring data within fitness-fatigue modelling may generate increased collaboration between researchers and practitioners. The use of extrapolation methods to generate high frequency measures may reveal a novel approach to studying FFMs in certain sports, such as powerlifting, where performance can be considered simultaneously a largely isolated dimension of physical capability and competitive outcome.

When using FFMs in practice, it is important to acknowledge that all measurement of physiological systems comprises error (instrumentation error and biological variability) which often makes it

difficult to accurately quantify underlying physical capability (Shrahili, 2014; Swinton *et al.*, 2018). Thus, true performance can be defined as a random variable where measured (observed) performance is only ever an estimate (Shrahili, 2014; Scarf *et al.*, 2019). FFMs acknowledge measurement error through introduction of a model error term, often assumed to be independent and identically distributed as Gaussian. Even where these assumptions are reasonable, higher error variances generally lead to lower precision in estimation. Several basic steps can be taken to reduce measurement error and its influence on any model's accuracy. These steps include selection of tests that comprise low measurement error (preferably less than 3-5% of coefficient of variation), standardisation of testing procedures and where possible taking the average of repeated independent tests (Swinton *et al.*, 2018; Stephens Hemingway *et al.*, 2019).

2.3.3 Parameter estimation approaches and limitations

Fitting an FFM (i.e., estimating the unknown parameters in the model) constitutes a nonlinear optimisation problem. The objective is to determine values for the parameters that maximise a goodness-of-fit function known as a loss, cost, or objective function (Soetaert and Petzoldt, 2010). Estimation of model parameters in FFMs has been historically approached from a least-squares (Hellard *et al.*, 2006; Pfeiffer, 2008; Proshin and Solodyannikov, 2018) or maximum likelihood perspective (Shrahili, 2014; Busso, 2017; Proshin and Solodyannikov, 2018; Scarf *et al.*, 2019). The nonlinear least-squares approach involves minimising the sum of squared deviations (errors) between modelled and measured performance via the following cost function:

$$\min \sum_{i=1}^{m} (\hat{p}_i - p_i)^2 \qquad (eq. 2.33)$$

Where i is an index over a set of m of data points $\{(p_1, \hat{p}_1), (p_2, \hat{p}_2), \dots, (p_m, \hat{p}_m)\}$ that represent measured (p) and modelled $(\hat{p})$ performance values at specific integer time points $t_i \in \mathbb{N}$. The term $\hat{p}_i$ describes the FFM function $f\left(t_i, \theta, \{\omega_1, \dots, \omega_{t_i}\}\right)$ that not only depends on the time-step input (i.e., $\Delta_t = 1$) up to time t_i (i.e. training load series) $\{\omega_1, \omega_2, \dots, \omega_{t_i}\}$ but also on n model parameters (θ) with $m \geq n$. For example, for the standard model the general parameters θ are the set $\{p^*, k_g, \tau_g, k_h, \tau_h\}$, where p^* is an additive term representing baseline performance, τ_g, τ_h are the decay time constants on fitness and fatigue, respectively, and k_g, k_h are the associated scaling factors. In general, an optimisation algorithm wraps around the objective function (eq. 2.33) and uses specific update and stopping criteria to methodically travel over the available parameter space in search of the best possible set (either the absolute maximum or minimum). NLS regression problems are typically solved by general minimisation algorithms. First and second order algorithms (including those that approximate the Hessian by the outer product of the gradient) have predominantly been used to direct the iterative search to minimise least-squares or maximise likelihood.

Finding suitable parameter values in this manner has been described as "notoriously difficult" in the study of similar modelling problems (Transtrum *et al.*, 2015). Fitting an FFM assumes that an optimum solution exists, is unique, and can be easily found within the multidimensional search space. The FFM even in its most basic form is a model in five dimensions, and therefore solutions cannot be visualised using standard plotting techniques. Furthermore, if there are truly different parameter sets that have the same global minimum under standard nonlinear least squares, there exists a situation where parameters aren't uniquely identified without additional constraints or regularisation terms. However, it is more probable that the presence of many local minima and saddle points present the primary challenge in a typical FFM fitting process. First and second order algorithms can often converge to local optima, or become stuck on saddles, provoking sensitivities in the fitting process and to the starting point in the search space (i.e., the jumping-off point for the algorithm). Identification of suitable model parameters from training and performance data is therefore an outstanding concern in fitness-fatigue modelling (Hellard *et al.*, 2006; Pfeiffer, 2008), and is an area that has received minimal attention in experimental or theoretical research.

In most complex models of real-world systems, parameters have compensatory (dependent) effects relative to the systems collective behaviour (Hellard *et al.*, 2006; Transtrum *et al.*, 2015). For example, with the FFM it is possible to modify a single decay constant without changing the resultant behaviour of the system, provided values in other parameters adjust to compensate (Transtrum *et al.*, 2015). Additionally, FFMs have been shown to be ill-conditioned, such that a small change in parameter values can reflect large changes in model behaviour (Hellard *et al.*, 2006; Pfeiffer, 2008; Stephens Hemingway *et al.*, 2019). This ill conditioning, also termed 'sloppiness' in systems modelling (Transtrum *et al.*, 2015) has been said to affect the accuracy and precision of parameter estimates (Bates and Watts, 1988). It has also been suggested that the fundamental value of FFMs is that they seek to link underlying physiological concepts to observed data via counterfactual structure, and therefore interpretability of parameter estimates in the real-world is viewed as a valuable and necessary property for the models to be useful in practice. However, model sloppiness has not precluded good predictive power in the application of biological, physical and chemical systems models (Transtrum *et al.*, 2015). Prediction accuracy, specifically out-of-sample is an area where FFMs have never been rigorously assessed, particularly under well-designed experimental conditions or using large retrospective datasets from practice. In studies where poor in-sample model fit has been reported, there have typically been issues such as low sample size, selection of an inappropriate measure or limited volumes of fitting data, or little information reported with respect to the fitting process. It is posited that accurate prediction of future response may still be possible under an FFM, even in the absence of stable, characteristic model parameters (i.e., those which can be carried over between training programs). Historically, researchers have fitted FFMs under one set of 'best guess' initial candidates

for parameters under first and second order algorithms (Clarke and Skiba, 2013). Given the concerns outlined above, research reporting a single model solution derived from a 'one shot' run of gradient-based optimisation does not reflect the possibility for uncertainty in parameter estimates. As a basic starting point for future use, practitioners and researchers fitting FFMs via least-squares using traditional derivative-based algorithms should re-fit the model under multiple starting points. In chapter 5, in-silico experimental work under synthetic data is carried out to investigate the magnitude of starting point sensitivity in the model fitting process for the standard and fitness-delay models, and then examines implications of these findings for previous and future research. In-sample model stability can also be assessed through repeated refitting of subsets of the data with points removed (Hellard *et al.*, 2006; Turner *et al.*, 2017). Out-of-sample prediction accuracy, via some form of cross-validation should also be implemented in future experimentation. The following section of this literature review discusses model evaluation, and in chapter 6 practical approaches are demonstrated in R (6.2.5), followed by reflection on suitable guidelines for further model experimentation (6.7).

Researchers and practitioners interested in fitting FFMs may also look to investigate whether derivative-free algorithms can achieve better results (Méline *et al.*, 2018; Philippe *et al.*, 2018; Connor and O'Neill, 2020). These algorithms often perform well with non-convex functions comprising multiple critical points. Although linear least-squares is guaranteed to be convex, there are no such guarantees for nonlinear least-squares problems. The class of evolutionary algorithms offers one promising method and may be effective in cases where constraints or penalty functions are applied via regularisation terms or custom objective functions. However, evolutionary algorithms including differential evolution, genetic algorithms, and particle swarm can require significant understanding and testing of control parameters to tune the algorithm to obtain appropriate results. Evolutionary algorithms are also slow compared to gradient-based methods that can exploit a smooth surface. Hybrid approaches may go some way toward offering a resolution to both the problems of finding suitable initial candidates under first and second order methods, and the lack of guarantee on optimality when applying evolutionary algorithms. A hybrid algorithm typically comprises a stochastic tunnelling method to identify convex basins of attraction surrounding optima, and then subsequently passes these as initial values for gradient-based local search. More advanced optimisation approaches such as the use of evolutionary algorithms have rarely been used or assessed for fitting FFMs (Turner *et al.*, 2017; Méline *et al.*, 2018; Philippe *et al.*, 2018; Connor and O'Neill, 2020), however, previous research has employed evolutionary algorithms to solve training program design problems using pre-defined FFM parameter estimates (Schaefer, Asteroth and Ludwig, 2015; Kumyaito, Yupapin and Tamee, 2018; Connor, Fagan and O'Neill, 2019).

Lastly, the problem of over-fitting can occur when the model is fit too closely to a limited set of data points, reflecting noise in the data (Everitt and Skrondal, 2002). If a model comprises more free

parameters than required to explain most of the variance in observed performance, this may permit explanation of idiosyncrasies within the data. In summary, resolving these issues often requires a larger number of data points, lower relative error within the measured data, re-formulation of the optimisation problem to reduce fitting sensitivity to noise in the data via introduction of distributional error terms (Shrahili, 2014; Busso, 2017; Scarf *et al.*, 2019), or in some cases re-specification of the model to reduce complexity (Hellard *et al.*, 2006; Pfeiffer, 2008). Researchers and practitioners in future collaborations may consider setting relatively tight a priori constraints on likely parameter values (or use of informative priors using a Bayesian approach). Alternatively, ensemble or hierarchical modelling approaches may be considered, aggregating parameters over multiple training and performance sets, or across groups of similar athletes. The introduction of regularisation terms and associated penalty functions that enforce constraints on the cost function may also assist in obtaining parameter values that provide stable predictions or limit overfitting.

2.3.4 Model evaluation: Integrating practitioner data into future research

The fundamental stages of mathematical modelling comprise: 1) problem formulation; 2) determination and examination of solutions; and 3) model evaluation (validity) across diverse cases applicable in the real world (Domotor, 2011). Of the three phases, FFM research is most limited regarding model evaluation to establish validity. As an appropriate starting point, in silico approaches under artificially established 'true conditions' can be used to provide an efficient and inexpensive method to study potential limitations of a model and early indications of predictive performance both in and out of sample (Morton, 1991; Busso and Thomas, 2006; Thomas, Mujika and Busso, 2009). For example, fitting-based in-silico approaches under realistic mock data generated via simulation can be used to identify lower bounds on properties such as prediction error via systematic and iterative deviation of controlled 'small world' conditions (e.g., various levels of measurement error and different testing frequencies). In these instances, the model can only be assumed to perform worse in the real world (Stephens Hemingway *et al.*, 2019). Computer-based experiments can also be used to explore elements such as fitting sensitivity and explore the effectiveness of different optimisation approaches. If lower bounds on model properties identified via these types of experiments appear unreasonable or indicate poor performance in real world applications, then further investment in studying the model under empirical/experimental conditions should be carefully considered, or simply avoided. If model utility is to be assessed under real-world data, then upmost consideration must be given to model selection, methods of fitting, and the framework of model evaluation used. Model selection is given detailed consideration with respect to future research in chapter's 6 and 7, with the existence of more advanced models covered shortly in section 2.4 (state-of-the-art). In section 2.3.3 parameter estimation was discussed specifically with regards to the common least-squares method and prospective algorithmic approaches. In chapter 6, section 6.2.4, practicalities of the optimisation

process in the language R are also discussed. In this subsection, model evaluation is considered, specifically with respect to prediction of future training response. Similarly, the sister section to this subsection is section 6.2.5 that presents a full implementation framework in R for model evaluation (based on theory described shortly).

It is helpful to first clarify definitions and usage of the terms “in-sample” and “out-of-sample”, with regards to available data and methods of model evaluation. When fitting an FFM to data, we hope to match a set of observed measures of physical performance with a set of model values (with the same associated days), that have been computed under a set of parameter estimates and series of training load values. The model parameters are manipulated by the algorithm with respect to the objective function, to iteratively seek out parameter values that provide the best correspondence between these computed model values and the set of observed physical performances. In this process, the training load values used to compute the model and observed data used to check correspondence with the predictions are known as *in-sample data*. If a metric such as the root-mean-square deviation (RMSD) - also referred to as root-mean-square error (RMSE) - is calculated from the differences between the modelled and in-sample observed values, this is a form of *in-sample testing* and reflects *in-sample model fit*. These three concepts (in-sample data, in-sample testing, in-sample model fit) are illustrated in figure 2.6-A. Consider the simple case where a few of the observed performance values are held back, and not used to train the model. These are then defined as *out-of-sample* data. If the resultant model predictions are then compared to these hold-back (or hold-out) datapoints in a similar manner as described, this represents a form of *out-of-sample testing* and reflects what is commonly referred to in the literature as the *model performance* (not to be confused with physical performance). We will call it something like *model error (out-of-sample)* to reduce confusion. These concepts are reflected in figure 2.6-B. However, this basic form of out-of-sample testing alone still does not give the entire picture of model predictive validity, and there is another necessary method that involves testing model predictions against out-of-sample data that is derived from a future training intervention. In this way, the inputs used to compute the model (from fitted model parameters) arise from a relatively non-homogeneous training program to the one used to fit the model (e.g., a different interventional distribution). This concept is illustrated in figure 2.6-C. It is important that both types of out-of-sample testing are implemented, because they tell us slightly different things. The first, tells us that the model can predict well (or not) for inputs that are similar to the one used to train the model. This is most important when considering model prediction for tapering and peaking strategies, where questions usually surround how best to reduce the magnitude of training over a short period of time whilst the type of training stays roughly similar. The latter, gives us insight into whether the model can predict well (or not) for inputs that are not similar (in say, shape, pattern, frequency, magnitude) to the one used to train it. This type of testing is crucial to understanding the utility of the model for assisting in training program design, as its very unlikely that coaches will implement the same type of program

block after block, and so the model needs to be able to still predict well for these differences in training inputs for it to be helpful in this type of use-case. Beyond figure 2.6 on the following page, the topic and terms surrounding *cross-validation* are introduced as they pertain to creating a more robust FFM evaluation approach for future experimental work. A distinct lack of out-of-sample testing represents the biggest limitation of FFM research to date. Fitness-fatigue models are flexible and authors who simply fit an FFM to all available data (e.g., Figure 2.6-A) without assessing forecasting accuracy (out-of-sample model error) limit the potential to assess both the extent of overfitting in the model, and the accuracy of predictions under future training programs. Historically, this has unfortunately been the practice in the bulk of prior FFM experimental research.

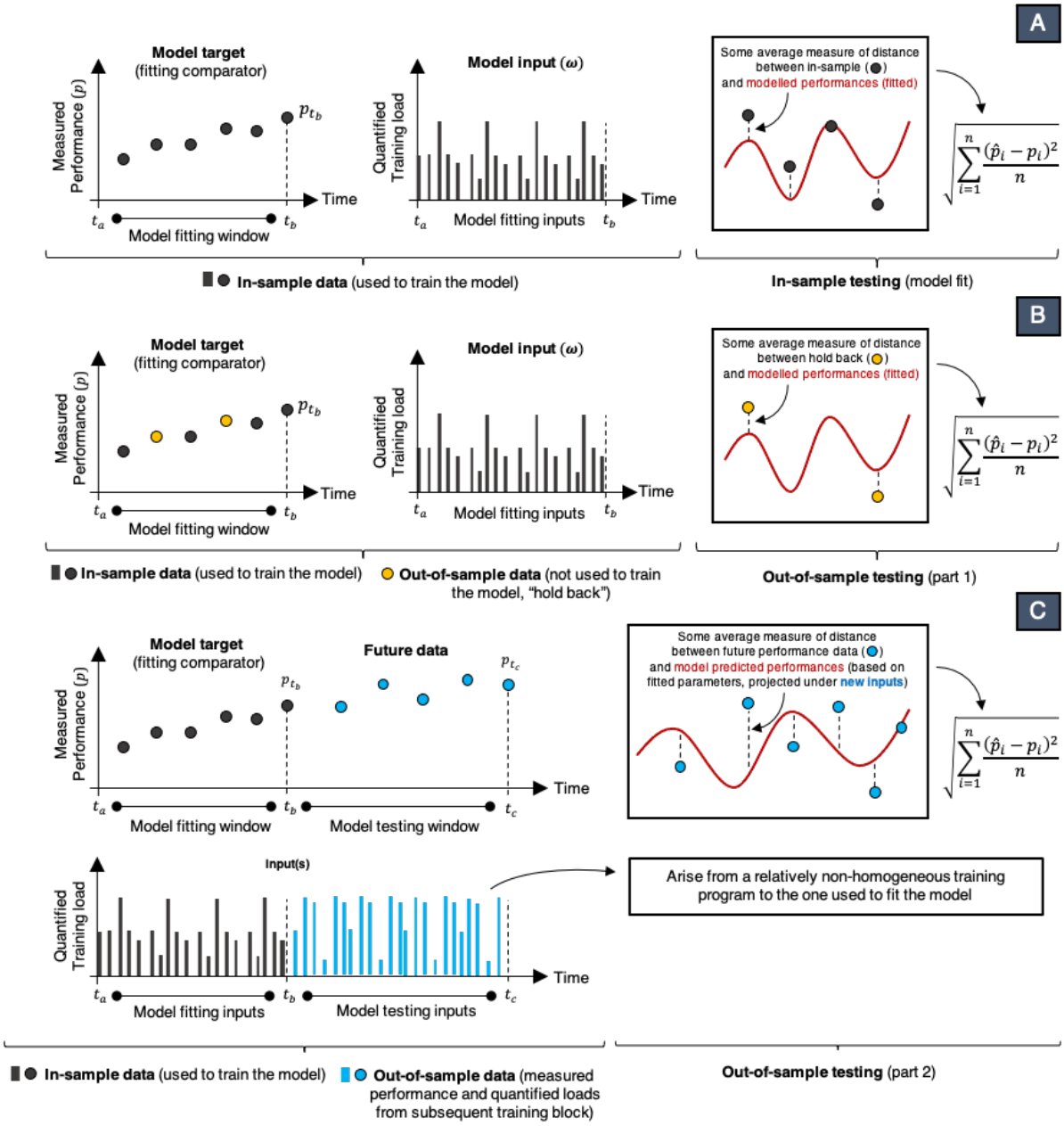


Figure 2.6: (A) In-sample data and testing; (B) Basic out-of-sample testing; (C) Out-of-sample testing of future performance predictions under different a different training program (model inputs).

Cross-validation (in its purest form) is a broad term to describe a particular collection of approaches used in modelling for estimating out-of-sample error (i.e., model accuracy). In the context of FFM evaluation, the basic principle of cross-validation involves partitioning the available data into two complementary subsets: 1) A training (or fitting) set; and 2) a testing set. The data within the training set is used to fit the model (estimate the unknown model parameters) and is therefore referred to as "seen" by the model. The testing set comprises performance data that has been held back from the fitting process and is therefore used to assess the accuracy (error) in model predictions. As shown in figure 2.6, it is the comparison between unseen data and fitted model predictions that provides a measure of model predictive ability and is of most interest when evaluating FFMs. In-sample prediction errors will be optimistically high in almost all cases due to the high number of available parameters for fitting the nonlinear model to data. In contrast, testing against unseen data provides insight into model misspecification and whether the FFMs counterfactual properties are robust (particularly when tested against different input distributions). Under most cross-validation methods, the training and testing process is repeated several times for different partitions of the data. Results are averaged over the rounds/partitions/splits (multiple terms to refer to the same thing) to reduce variability. There are many possible approaches to how the data can be partitioned. The simplest is the *hold out* method, that does not typically involve more than one round (partition) of fitting and evaluation (unless a set of future 'block 2' dataset is also included, in which case it is reasonable to return to the full 'block 1' dataset) to derive final point estimates of the FFM parameters. As the FFM. Is a time-series model, cross-validation techniques that mix previous and future data require special care as they do not respect the temporal order between time-series data nor probable autocorrelation and therefore are likely to introduce look-ahead bias leading to overconfidence in predictive capability. Therefore, methods such as *random hold-out*, and *k-fold* are not recommended. *Combinatorial purged cross validation* offers a more principled approach to mixing past and future, and readers are referred to Prado (2018) for a detailed explanation. A potential walk-forward approach suitable for FFM applications is also the *expanding window* method, where the proportion of data used to train the model progressively increases. Although it may be tempting to consider tuning the model parameters based on the cross-validated loss, tuning using a testing set will result in overoptimistic loss estimates. In contrast, averaging the estimates from the expanding windows may reduce estimator variability in the same way as "bagging" in machine learning (Breiman, 1996). How to best use estimates from cross-validation windows to tune models, and advantages and disadvantages are more complicated topics and possible avenue for future research, although are out with the scope of this work. Figure 2.7 below illustrates the hold-out method and expanding-window method. In chapter 6, practical resources are provided in R along with further discussion for implementing of the expanding window method in the context of fitting and evaluating a fitness-fatigue model.

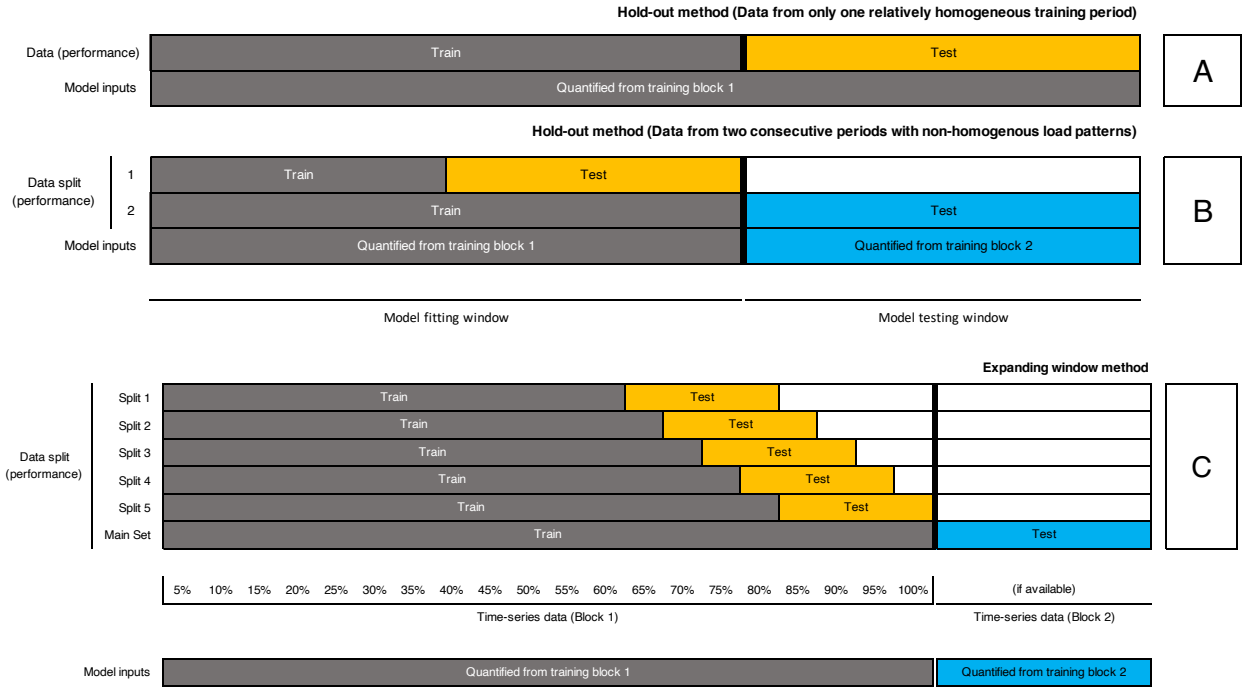


Figure 2.7: Two approaches to cross-validation of fitness-fatigue models (hold-out and expanding-window)

With five parameters in the standard model (eq. 2.9) and upwards of eight in more complex formulations, FFMs would be expected to fit most patterns of data, even those generated from a process very different to that underlying the FFM. If an athlete's training loads are consistent and average performance is trending upwards linearly for example, then both the FFM and a straight line are sufficiently expressive. Provided this pattern continues from training into the hold-out periods, the predictions, even those out of sample, say little about the FFM's ability to meaningfully predict outcomes. In contrast, if the athlete is presented with a new training plan with substantial variation in the training loads, a more effective framework is available to determine if the model can make nontrivial, testable predictions. In statistical Design of Experiments (DoE), variation in the explanatory factors is purposefully created to establish uncertainty in parameter estimation. Optimal designs for evaluating FFMs, including configurations of the training impulses best suited for learning about the model's predictive utility represent another potential direction for future research. In the meantime, researchers can draw from essential DoE design concepts. Since the external "factors" in FFMs are convolutions of training inputs, training loads are required that vary for enough time to modulate these convolutional quantities. For fatigue, this may mean prescribed rest periods of days to weeks. For fitness, creating variation is easiest when the individual has low prior levels, as it is a feature of the model that fitness is relatively stable. For maximal learning, longer rest periods or prolonged lighter

training after the athlete has attained high levels of fitness may best facilitate this variation but may potentially create issues for participation and adherence to studies in elite populations.

The use of covariates, i.e., additional variables that help explain performance beyond the FFM's fitness and fatigue constructs, have not been explored in the literature. To be a true covariate, the variable must *not* be a downstream effect of performance, as this complicates interpretation and leads to "collider bias" (Cole *et al.*, 2010) if it is also a downstream effect of training. Pre-study, stable variables like age and gender are thus the safest candidates, though by lagging time-varying covariates by one period, that risk is diminished. Even so, conditioning on covariates will alter the interpretation of k_g and k_h, as these are now the effects of fitness and fatigue conditional on values of the covariates. Finally, note that the use of covariates makes prediction at longer horizons more difficult, since values of the covariates are needed into the future.

Lastly, consideration must be given to possible sources of suitable data for assessing model utility in the sport sciences. Laboratory studies (controlled training interventions on recruited participants) to collect experimental data are expensive, require significant planning, logistical consideration, and are often of limited ecological validity. It makes little sense to go straight to this type of experimental data-collection approach when it is both possible and likely (given the increase in means and desire to collect data in practice) that suitable data may already exist and which could be used to backtest models to assess model utility. This is exactly what is meant here by integrating practitioners (data) into future research efforts. More attempts must be made by researchers to reach out into the various bubbles of sport science practice (e.g., strength and conditioning, performance monitoring, athletics) to assess whether coaches have both the data and interest to collaborate and further the area of FFM research via backtesting approaches.

2.3.5 The utility of an FFM: Informing training program design

Currently, exercise prescription in research and practice relies heavily on qualitative reasoning, coach and athlete experience, and what is known about underpinning physiological mechanisms from the scientific literature (Scarf *et al.*, 2019). However, individualisation of physical training via quantitative approaches is an area where many open problems still remain (Scarf *et al.*, 2019), such as appropriate training load quantification and outcome selection for existing models such as the FFM. It is recognised that Banister and colleagues were pioneers in advocating for sizeable tapering periods prior to competition to achieve optimal performance (Morton and Fukuba, 2011). For coaches and practitioners, the potential value in applying the quantitative representation of the FFM is partial control of performance potential at specific times or periods during a training cycle. This particular problem can be defined as how best to determine a training program to produce a desired level of

performance at a specific time (Fitz-Clarke, Morton and Banister, 1991), and is also in the domain of model predictive control (Kolossa *et al.*, 2017). Although the desired utility of existing FFMs (i.e. accurate out-of-sample future predictions) has not been rigorously tested, prospective methods for FFMs to generate future training programs under constraints are available and have been previously assessed (Schaefer, Asteroth and Ludwig, 2015; Turner *et al.*, 2017; Kumyaito, Yupapin and Tamee, 2018; Connor, Fagan and O'Neill, 2019). If FFMs or alternative models are determined to be sufficiently accurate, they may be applied within constrained optimisation frameworks to assist in identifying particular programs that are likely to provide the highest rates of physical improvement or performance over a planned training period (Busso and Thomas, 2006; Schaefer, Asteroth and Ludwig, 2015). To summarise briefly some of the previous work conducted so far, Schaefer *et al.* (2015) treated plan generation as a constraint-satisfaction problem (CSP) and assessed several optimisation frameworks for solving them to derive training plans. Model parameter estimates were pre-defined from a previous study, and their work focused solely on assessing whether reasonable training plans could be under the problem of plan-generation using non-linear solvers (Schaefer, Asteroth and Ludwig, 2015). Similarly, Connor *et al.* (2019) used mathematical programming to solve the training plan generation problem under an FFM framework using grammatical evolution, obtaining reasonable training programs for an elite team of Gaelic football players (Connor, Fagan and O'Neill, 2019). Kumyaito *et al.* (2019) attempted to create practical cycling plans that satisfied physiological constraints (with regard to monotony, chronic training load ramp rate, and daily training impulse), but which were reasonable for improving athletic performance. The authors applied adaptive particle swarm optimisation using ϵ constraint methods to formulate the plans and simulate likely performance outcomes (Kumyaito, Yupapin and Tamee, 2018). In this instance, the epsilon (ϵ) constraint method is an algorithm transformation approach that facilitates the handling of constraints using an ϵ level comparison that compares search points based on the pairing of their objective function value and any violation of constraints (Yang, Cai and Fan, 2014). Collectively, these studies represent viable approaches to deriving future training programs with constraints, following identification of useful performance models (perhaps FFMs) that attempt to describe the relationship between training and performance. The importance of this area of research is arguably undervalued at present due to the lack of strongly predictive models in the field of sport science. However, it is reasonable to suggest that this growing body of work can be viewed as paving the way for possible future integration of mathematical models into usable practitioner-focussed software if a useful model is identified or developed.

2.3.6 Current applications: Availability of software and data

Historically, researchers have developed FFM applications tailored to their own requirements, leaving several potentially useful resources difficult or impractical to extend; and therefore unlikely to be used in practice. The two linked tables in this section highlight the current gaps in practical resources for researchers and practitioners to fit and evaluate FFMs in experimental investigations. Prior to this project, the area of available resources for research applications was virtually untouched and required significant investment in time and skill in programming to develop tools that would encourage the uptake of FFM study in contemporary research. Toward this end, a substantial open-source code repository was developed in the statistical programming language R to enable FFM implementation and model testing. These resources are highlighted in the table below for completeness but are discussed in more depth in chapter 6, when the relevant programming concepts and methods of implementation are examined in full. Corresponding URLs to table 2.2 are provided in table 2.3.

Table 2.2: Current software resources available for fitness-fatigue modelling research and education. Highlighted (boxed) row refers to resources developed as part of this research project (further examined in chapter 6).

No	Author(s)	Resource	Development	Type	Skill required
1	Golden Cheetah	Software that includes functionality to compute the basic FFM	Active (not specific to FFM functionality)	GUI (Desktop software)	Low (point and click)
2	(Turner et al., 2017)	Optimiser for non-linear variant of Banister's model using genetic algorithm. Includes plan generation optimiser.	Inactive as of first upload	Source code only	High (compile and interface with code, no documentation)
3	Cooke, Andrew	Interactive training diary similar to a code notebook. Has limited FFM fitting functionality	Active (not specific to FFM functions)	Source code only	High (compile and interface with code)
4	Training peaks	Subscription based software to track endurance training data, that has limited FFM fitting functionality	Active (not specific to FFM functionality)	GUI (multi-platform)	Low (point and click)
5	(Ludwig, Schaefer and Asteroth, 2016)	A custom training portal to fit FFMs and generate optimised training plans under constraints	Inactive as of first upload	Source code only	High (compile and source programmatic interface)
6	Jovanovic, Stephens Hemingway, Swinton	dorem: A dose-response modelling package in R	Active (in early release)	R package	Moderate (command line interface in R but easy commands)

No	Author(s)	Resource	Development	Type	Skill required
7	(Stephens Hemingway *et al.*, 2021)	Code repository of flexible FFM functions and didactic code resources	Active	R functions	Moderate (extensive documentation and code examples provided)
8	(Kolossa et al., 2017)	MATLAB Simulink files from associated Kalman filter experiment	Inactive but well-annotated code	Source code files only	Moderate-high, must have familiarity with commercial language Matlab.
9	(Clarke and Skiba, 2013)	Basic spreadsheet demonstrating the standard fitness-fatigue model and critical power model. Provides some limited fitting functionality.	Inactive	Excel spreadsheet	Low

Table 2.3: URLs for current FFM software

No	Resource	URL	License	Open-source	Platform
1	Golden Cheetah software	goldencheetah.org	Free (GPL 2.0)	Yes	C++
2	(Turner et al., 2017)	github.com/jturner314/nl_perf_model_opt	Free (unspecified)	Yes	C++
3	Cooke, Andrew	github.com/andrewcooke/choochoo	Free (GPL 2.0)	Yes	Python (primary)
4	Training peaks	trainingpeaks.com	Paid	No	Multi
5	(Ludwig, Schaefer and Asteroth, 2016)	github.com/dawedawe/traipor	Free (ISC)	Yes	Python
6	Jovanovic, Stephens Hemingway, Swinton	dorem.net	Free (MIT)	Yes	R
7	(Stephens Hemingway *et al.*, 2021)	fitnessfatigue.com github.com/bsh2/fitness-fatigue-models	Free (GPL 2.0)	Yes	R
8	(Kolossa et al., 2017)	github.com/rub-ksv/Kalman-Fitness-Fatigue	Free (unspecified)	Yes	MATLAB
9	(Clarke and Skiba, 2013)	pubmed.ncbi.nlm.nih.gov/23728131/	Free (unspecified)	Yes	Excel

2.4 Model developments and extensions: A state-of-the-art

Individualised exercise prescription in sport research and practice relies heavily on coach and athlete experience, prior knowledge of the physiological mechanisms behind average response, and can in some cases be supplemented with quantitative monitoring approaches (Scarf *et al.*, 2019). The process of individualised exercise prescription and the subsequent training regimes developed can be further enhanced through increased quantitative monitoring, modelling and prediction (Scarf *et al.*, 2019). Each component of this process represents an active area of research where many open problems remain. Ideally, any performance model developed would possess specific qualities relating training to the adaptive response with respect to causality, predictability, interdependence, non-linearity, relevance to specific training variables, and ease of identifying underlying physiological structures (Rasche and Pfeiffer, 2019). However, development of any performance model is an iterative process of refinement, and it is unlikely an ideal model will be derived in a first attempt. It is worth noting that in areas such as modelling, progress can be made even in the absence of what is traditionally defined as success in other areas of science or industry (e.g., a model that is ready for use *in pleno*). It is argued here that as a field we must be careful to be balanced in our criticisms of prior work, and also appreciative of the historical efforts of researchers and practitioners who have devoted their time and in some cases a careers worth of work to develop these models to their current iterations. As with any challenging pursuit, getting it right the first time is an unrealistic ideal and, in most cases, the prior efforts of others still contribute greatly toward advancing our knowledge and moving us closer to the end goal (in this case a useful model). Moreover, with the advances in technology and modern developments in computing, sport science has never before found itself in a position of having so many opportunities to progress the areas of mathematical and statistical modelling in research and practice, if collaborative efforts are maximised.

Given the consideration that this literature review is unlikely to be read in its entirety in one sitting, and in the interest of maximising reader understanding, we briefly refresh some practical interpretation of model fitting prior to the remainder of this section. Recall that from a technical point of view, the initial step of the modelling process involves training the model to generate fitted model parameters. This procedure may be understood as learning the athlete-specific dynamics in terms of an individual fingerprint, which could help to understand adaptational characteristics. In practice the data usually consists of measured load and performance data in combination with an initial set of model parameters. Through an iterative parameter optimisation process aimed at minimising a loss function, a fitted set of parameters is generated, which results in performance estimates in conjunction with a measure of in-sample model fit (i.e., coefficient of determination R^2). In a second step, the performance model is tested to validate the optimised parameter set using either a fraction of the measured load, which was not included in the fitting process, or future training load of interest. The resulting performance

estimates estimate the out-of-sample model fit, and these are compared with known performance measurements not included in the model training process. In that way, besides predicting future performance, existing knowledge of researchers and practitioners regarding the individual effects of a given load on performance may be extended.

It was over forty years ago that Banister *et al.* (1975) described the first dynamical systems model of training response and referred to in this thesis as the standard FFM. Despite consistent scientific attention over a long history, their seminal model and associated extensions remain in the domain of exploratory research. The first half of this literature review has examined the limitations of the standard model and identified the existing issue of low uptake of FFMs across contemporary research and in real world practical environments. Novel approaches were identified that might enhance the use of basic FFMs including a focus on predicting physical performance via simple tasks or exercises, the generation of high frequency performance data through routinely collected training metrics, and development of unique training load weightings to account for multiple modalities (e.g., hypertrophy, strength or power training) and their specificity according to the performance outcome. Although these recommendations may provide productive approaches and avenues for future research, it is important to note that they fail to address some of the limitations inherent to the structure of basic FFMs. It has been known for an extended period that the standard FFM and subsequent first-order filter extensions suffer from several limitations that are at odds with the conceptual understanding of training response (Banister *et al.*, 1975; Calvert *et al.*, 1976; Morton, Fitz-clarke and Banister, 1990; Busso, Carasso and Lacour, 1991; Busso, Candau and Lacour, 1994; Rasche and Pfeiffer, 2019). Therefore, the second half of this literature review focusses on these limitations and novel attempts within the existing literature that have been made to remedy them. Three key limitations include: 1) the *linearity* assumption, where performance can be arbitrarily increased by simply increasing the training dose and for example doubling the training dose leads to double the improvement (Hellard *et al.*, 2006; Rasche and Pfeiffer, 2019); 2) the *independence assumption*, where there is no interaction between training sessions and therefore training performed on a given day does not influence the response generated from another session; and 3) the *deterministic assumption*, where uncertainty in model parameters and observed performance are not modelled directly and are not updated based on incoming information (Kolossa *et al.*, 2017).

Two general approaches have been proposed to address limitations inherent to basic FFMs: 1) altering the model input through constraining or saturating training loads (Hellard *et al.*, 2005); and 2) altering the model formulation (Busso, 2003; Kolossa *et al.*, 2017; Turner *et al.*, 2017). Under the first of these approaches, Hellard *et al*. (2005) proposed the use of a threshold saturation function, termed the Hill function (Krzyzanski, Perez-Ruixo and Vermeulen, 1999) to address the limitations of the linearity assumption and the result that indefinite increases in training loads result in continual increases in

performance. Their use of a threshold saturation function to transform training loads to a non-linear input with asymptote worked outside of the model formulation and therefore avoided changing the structural relationship between the input and output specified (Rasche and Pfeiffer, 2019). In contrast, under the second approach the structural relationship between model components is changed with previous research focussing on modifications to account for each of the three key limitations previously identified. This remaining section of the literature review will therefore introduce and detail both approaches. In particular: 1) changes to the input via the Hill function (Hellard *et al.*, 2005); 2) changes to model specifications under the variable dose-response (VDR) model (Busso, 2003) and a recursive delay-differential model (Matabuena and Rodríguez-López, 2016, 2019), which were both designed to create a dependence between subsequent training sessions and thus address the issue with the independence assumption; 3) the Kalman filter to explicitly model uncertainty in model parameters and update model results based on incoming information to address the deterministic assumption; and 4) inclusion of non-linear power terms in the underlying model system (Turner *et al.*, 2017) to try and reflect diminishing returns and overtraining effects. Further key model developments that feature in the literature are also examined, such as an FFM with time-varying parameters (Busso *et al.*, 1997), an exponential growth model (Philippe *et al.*, 2018), and more recently a novel secondary-signal model proposed by Busso (2017).

2.4.1 Modification to model input and inclusion of non-linearity

Prior to modifying the formulation of the standard fitness-fatigue model, an approach to address the limitation of arbitrarily large increases in performance is to apply restrictions to unconstrained training inputs (ω). An initial method to consider is to simply adopt an interpretable scale with a maximum value (e.g., 0 to 100). If, for example, a maximum value of 100 is set, then from the standard model (eq. 2.9) the maximum amount of fitness (or fatigue) that can be achieved from a single training session is $100 \times k_g$ (or $100 \times k_h$) where the units of the multiplicative factors match that of the performance being modelled. Additionally, given the nature of general FFMs and their solutions involving exponential functions, these fitness and fatigue effects are at their maximum immediately after a training session. It follows that researchers and practitioners can identify physiological constraints such as maximum improvements that could reasonably occur over a short period (e.g. 1-week) and use this value to set upper bounds when estimating model parameters. Whilst this approach assists with creating more interpretable parameters and training inputs, it does not address limitations of linearity in training response. A potential solution to this limitation was proposed by Hellard *et al*. (2005). The authors proposed the use of a Hill function (Krzyzanski, Perez-Ruixo and Vermeulen, 1999) that mapped training inputs in a non-linear fashion to a threshold (κ), such that higher inputs have no additional effect on performance. Figure 2.8 demonstrates the behaviour of the Hill saturation function,

as recreated from parameter values provided in the authors original paper (Hellard *et al.*, 2005). The Hill function is described by the following equation

$$Hill(\omega) = \kappa\left(\frac{\omega^{\gamma}}{\delta^{\gamma} + \omega^{\gamma}}\right) \tag{2.34}$$

Where the parameter γ expresses the sensitivity of the training load and controls the time to reach the threshold κ (larger γ leads to shorter times to reach κ), and δ is the inertia of the function to the threshold value (low δ expresses a strong effect of dose on performance). In applying the Hill function to the standard FFM there are multiple options available. One option includes setting κ to an interpretable maximum value (e.g., 100) and then setting γ, δ to control the non-linearity desired. The scaling parameters k_g and k_h would then be used to transform training loads to change values in the performance measures. Alternatively, there is potential to develop more complex transforms and for a given input (ω), map this to different non-linear fitness $\{\kappa_g, \gamma_g, \delta_g\}$ and fatigue $\{\kappa_h, \gamma_h, \delta_h\}$ forms. Where κ_g, κ_h could replace the scaling coefficients k_g, k_h and represent the maximum amount of fitness and fatigue expected from a single training session. The parameters γ_g, δ_g and γ_h, δ_h tune the degree of non-linearity for fitness and fatigue, with performance estimated via the following expression:

$$\hat{p}(t) = p^{*} + \sum_{i=1}^{t-1} \omega_g(i) \cdot e^{-\frac{(t-i)}{\tau_g}} - \sum_{i=1}^{t-1} \omega_h(i) \cdot e^{-\frac{(t-i)}{\tau_g}} \tag{2.35}$$

Where the fitness component training load function is defined as:

$$\omega_g(i) = \kappa_g\left(\frac{\omega_i^{\gamma_g}}{\delta_g^{\gamma_g} + \omega_i^{\gamma_g}}\right) \tag{2.36}$$

And similarly, the fatigue component training load function is defined as:

$$\omega_h(i) = \kappa_g\left(\frac{\omega_i^{\gamma_h}}{\delta_h^{\gamma_h} + \omega_i^{\gamma_h}}\right) \tag{2.37}$$

Hellard *et al.* (2005) investigated whether modifying training input with a Hill function resulted in better model fit compared with standard linear input across 7 elite swimmers engaging in a long study period (100-200 weeks). After statistically controlling for the additional parameters, Hellard *et al.* (2005) identified slightly improved model fit with the addition of the Hill function (R^2_{adj} = 0.43 $\pm$ 0.1 vs. 0.36 $\pm$ 0.1). However, goodness-of-fit was still poor across all athletes and both models. The

authors included no out-of-sample assessment of model predictions, and so it is also difficult to evaluate the extent of model overfit considering the additional flexibility the Hill function affords. It is possible that fitting the model over a long duration (100-200 weeks) contributed to the generally poor fit across both models, as parameters obtained may not remain stable for such long periods, either due to estimation issues or underlying change (discussed in section 2.3.3 of this review). Chapter 6 discusses optimisation routines that could be developed for estimating the Hill function parameters for fitness and fatigue, and the rate decay constants.

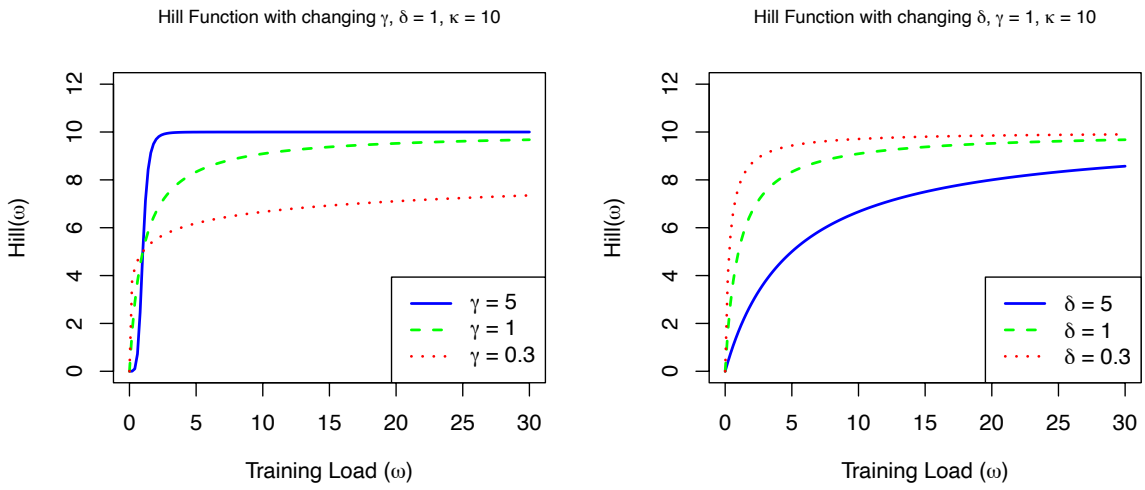


Figure 2.8: Arbitrary behaviour of Hill training load saturation function (eq. 2.29), when $\kappa = 10$, and γ, δ vary as shown. Plots recreated from parameter values provided in Hellard et al. (2005).

2.4.2 Interaction between training sessions: The variable dose-response (VDR) model

As identified earlier, one of the key limitations of the standard FFM is the issue taken with the independence assumption, in particular the disagreement that the response to the current training session is not influenced by previous sessions. Busso (2003) was the first to propose a model to address this limitation that included the addition of a first-order filter on the fatigue component and is commonly referred to as the variable dose-response (VDR) model. To account for the interaction between training sessions Busso focused on the perspective that fatigue was most likely to be influenced by previous training (with higher previous training loads generating greater fatigue) and so introduced a further 'gain' term for fatigue with associated time constant τ_{h_2} (Busso, 2003). One specific implementation of the model can be presented as:

$$\hat{p}(t) = p^* + k_g \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_g}} - k_h \sum_{i=1}^{t-1} k_{h_2}(i) \cdot e^{\frac{-(t-i)}{\tau_h}} \tag{2.38}$$

Where the first-order filter $k_{h_2}(i)$ is calculated by a series of decaying exponentials with scaling factor $k_h \geq 0$ and time-constant $\tau_{h_2} \geq 0$, such that:

$$k_{h_2}(i) = \sum_{j=1}^{i} \omega_j \cdot e^{\frac{-(i-j)}{\tau_{h_2}}} \tag{2.39}$$

Where the term j is the summation index within the recursive function.

These dynamics are intuitive and appear to be a more reasonable description of the change in fatigue response with accumulation of previous training (Chiu and Barnes, 2003) compared to the standard FFM. It can be seen from (eq. 2.38) that both fitness and fatigue components are represented by a sum of exponentially decaying training loads. For the fitness component each decaying training load is set by the individual training input ω_i. In contrast, it can be seen from (eq. 2.39) that each decaying training load for fatigue is set by a weighted average comprising ω_i and previous training sessions $(\omega_1, \dots, \omega_i)$. If τ_{h_2} is set to a low value, there is limited interaction between training sessions and each decaying training load is set primarily by ω_i. However, for larger values of τ_{h_2} there is greater interaction, and the weighted average is influenced to a greater degree by previous training sessions. Rasche and Pfeiffer (2019) also developed and presented a recursive form of this model in their chapter *Training* from the book *Modelling and Simulation in Sport and Exercise* (Baca and Perl, 2018). As follows,

$$g(t) = g(t-1) \cdot e^{-\frac{1}{\tau_g}} + \omega_t \tag{2.40}$$

$$h(t) = \Big(h(t-1) + \omega_{(t-1)} \cdot h_2(t-1)\Big) \cdot e^{-\frac{1}{\tau_h}} \tag{2.41}$$

$$h_2(t) = h_2(t-1) \cdot e^{-\frac{1}{\tau_{h_2}}} + \omega_t \tag{2.42}$$

Where the performance model is described by:

$$\hat{p}(t) = p^* + \Big(k_g(g(t) - \omega_t)\Big) - \big(k_h \cdot h(t)\big) \tag{2.43}$$

Where equations 2.40-2.43 (along with the standard model and most other FFM formulations) assume a 'pre-loaded' model computation. This implies that modelled performance on a given day t is computed prior to that day's training influence rather than after (Rasche and Pfeiffer, 2019). This assumption is reasonable, as in the real world athletes are highly unlikely to train prior to a competition or performance (Rasche and Pfeiffer, 2019). In the case where performance values are derived from

training data, these are also likely to be extracted from the first few exercises in a training session when an athlete is most capable, and also seems to be a reasonable assumption from a modelling perspective.

To assess whether the VDR model could better describe the response to training compared with the standard FFM, Busso (2003) fit both models to six healthy males undergoing a prospectively designed training schedule comprising progressive overload followed by training cessation. Whilst the findings demonstrated the VDR model produced significantly improved fit ($R^2_{adj} = 0.931 - 0.958)(P < 0.001$) compared to the standard FFM $\left(R^2_{adj} = 0.917 - 0.943\right)$, the magnitude of the improvement was small (Busso, 2003). Importantly, Busso (2003) did not include out-of-sample testing and so an assessment of predictive validity is not provided. As a result, it is possible that the additional parameter could improve initial model fit but create less accurate predictions to future training programs due to overfitting. Busso (2003) also demonstrated that the VDR model created an n-shape relationship between constant daily training thereby partly addressing issues of linearity with the original fitness-fatigue model. In chapter 6 (section 6.2.1 and 6.2.2), the VDR model is re-visited, with the behaviour explored and fitting process implemented in the R programming language.

2.4.3 Inclusion of uncertainty and feedback: The Kalman filter

The VDR model and standard FFM do not attempt to model uncertainty in the processes that generate fitness, fatigue and performance estimates. Additionally, estimates typically remain fixed irrespective of new incoming data. There have been some limited examples where estimates are updated by applying constraints that link for example successive sets of parameter estimates via a least squares algorithm (Busso *et al.*, 2002). However, FFMs could be expressed as linear state-space models thereby integrating uncertainty. A state-space representation is a mathematical model of a physical system as a set of input (e.g., training load), output (e.g., performance) and state variables (e.g., fitness and fatigue), where the unobserved state evolves over time depending on current values (e.g., in a recursive format as described in equations 2.11 and 2.41) and the system's input. The linear descriptor refers to the ability to specify the models in terms of matrices and linear algebra calculations.

The recursive form of the standard FFM (eq.'s 2.10 – 2.12) can be specified as a linear state-space model using the following matrix calculations.

$$\mathbf{x}_{n+1} = \mathbf{A}_n\mathbf{x}_n + \mathbf{B}_{\boldsymbol{n}}\omega_n + \mathbf{v}_n \tag{2.44}$$

Where in equation 2.44 $\mathbf{x}_{n+1}$ is a vector comprising fitness and fatigue on day $n+1$, $\mathbf{A}_n$ is a diagonal 2×2 "transition matrix" of coefficients that multiply the current fitness and fatigue values, $\mathbf{B}_n$ is a 2×1 matrix of coefficients that multiplies the scalar training input, and $\mathbf{v}_n$ is the state noise quantified by a 2×2 covariance matrix that is generally denoted by $\boldsymbol{Q}$. The state noise describes the

random changes in state (e.g. fitness and fatigue) above and beyond the deterministic component involving training and past fitness or fatigue. Kolossa *et al*. (2017) noted that these random changes can be considered unmodeled exertions of the athlete and additional features not captured by the fitness and fatigue relationship with training load.

To match the standard FFM, the matrices in equation 2.44 are set to the following:

$$\mathbf{x}_n = \begin{pmatrix} g(n) \\ h(n) \end{pmatrix}, \qquad \mathbf{A}_n = \mathbf{A} = \begin{pmatrix} e^{-\frac{1}{\tau_g}} & 0 \\ 0 & e^{-\frac{1}{\tau_h}} \end{pmatrix}, \qquad \mathbf{B_n} = \mathbf{B} = \begin{pmatrix} e^{-\frac{1}{\tau_g}} \\ e^{-\frac{1}{\tau_h}} \end{pmatrix}$$

$$\mathrm{Var}(\mathbf{v}_n) = \boldsymbol{Q} = \begin{pmatrix} \sigma_g^2 & \sigma_{g,h} \\ \sigma_{g,h} & \sigma_h^2 \end{pmatrix} \tag{2.45}$$

where the state of the system is not observed directly but is accessible by means of indirect measurements of performance p_n from the equation:

$$p_n = p^* + \mathbf{C}_n\mathbf{x}_n + \boldsymbol{\eta}_n \tag{2.46}$$

where $\mathbf{C}_n = \mathbf{C} = \left(k_g, -k_h\right)$ a 1×2 matrix and $\boldsymbol{\eta}_n$ is the observed performance noise described by actual measurement errors with variance denoted by ξ^2.

Expressing the FFM as a state-space model has the advantage that uncertainty in the state and measured performance can be modelled, addressing the reality that all models are limited and contain uncertainty. Kolossa *et al*. (2017) also described how the Kalman filter could be combined with a state-space representation of an FFM to obtain better fitness and fatigue estimates with incoming data. The ability of the Kalman filter to operate on quantities exhibiting statistical noise over time and iteratively update may provide practitioners with an effective resource to optimise training prior to important events.

The goal of the Kalman filter is to generate an "a posteriori" state estimate $\hat{\mathbf{x}}_n$ based on "a priori" estimate $\mathbf{z}_n$ from the deterministic mechanics and the observed performance p_n. The extent to which the a priori estimate is updated depends on the relative extent of the uncertainty in the state to the uncertainty in the measurement (the ratio of $\boldsymbol{Q}$ to ξ^2). When the uncertainty in the state is large relative to the uncertainty in the measurement, the "Kalman gain" will be high and the filter will place more weight on the incoming performance data and relatively large corrections can be made in the a posteriori estimate. In contrast, when measurement uncertainty is large relative to uncertainty in the

state, little weight will be placed on incoming data and there will be minimal correction to the initial a priori estimate.

More formally, the stages of the Kalman filter are expressed in the following procedural flow.

1) Calculate the a priori state estimate $\mathbf{z}_n$

$$\mathbf{z}_n = \boldsymbol{A}\hat{\mathbf{x}}_{n-1} + \boldsymbol{B}\omega_{n-1} \tag{2.47}$$

where $\hat{\mathbf{x}}_{n-1}$ is the previous a posteriori state estimate, or for the case where $n = 1$, $\hat{\mathbf{x}}_0$ is the initial state which may be set to $g(0) = h(0) = 0$ or estimated as unknown parameters.

2) Calculate the Kalman gain $\mathbf{K}_n$, as a 2×1 matrix, defined as

$$\mathbf{K}_n = \boldsymbol{M}_n\boldsymbol{C}^{\boldsymbol{T}}\left(\xi^2 + \boldsymbol{C}\boldsymbol{M}_n\boldsymbol{C}^{\boldsymbol{T}}\right)^{-1} \tag{2.48}$$

where $\boldsymbol{M}_n$ is the covariance matrix of $\hat{\mathbf{x}}_n$ which is iteratively updated (eq. 2.50). The Kalman gain scales the transformed matrix $\boldsymbol{M}_n\boldsymbol{C}^{\boldsymbol{T}}$ by the scalar value $(\xi^2 + \boldsymbol{C}\boldsymbol{M}_n\boldsymbol{C}^{\boldsymbol{T}})$, which describes the total variance in the state plus the variance in the measurement.

3) Calculate the a posteriori state estimate $\hat{\mathbf{x}}_n$, as

$$\hat{\mathbf{x}}_n = \mathbf{z}_n + \mathbf{K}_n(y_n - \mathbf{C}\mathbf{z}_n) \tag{2.49}$$

The third stage of the Kalman filter is the correction stage where the a priori estimate $\mathbf{z}_n$ is updated after observing the performance p_n. The Kalman gain $\mathbf{K}_n$ is expressed as a 2×1 matrix such that the correction to fitness and fatigue can be distinct and influenced by the state noise of each component and the error covariance specified. The Kalman gain for each component is multiplied by a scalar which is equal to the difference between the observed performance and the initial estimated performance ($\mathbf{C}\mathbf{z}_n$). Therefore, the larger the difference between the initial estimated performance and the observed performance the larger the correction. The fourth and final stage of the Kalman filter is

4) Update the estimation error covariance matrix $\boldsymbol{M}_n$

$$\mathbf{M}_{n+1} = \boldsymbol{Q} + \boldsymbol{A}\boldsymbol{M}_n\boldsymbol{A}^{\boldsymbol{T}} - \boldsymbol{A}\boldsymbol{M}_n\boldsymbol{C}^{\boldsymbol{T}}\left(\xi^2 + \boldsymbol{M}_n\boldsymbol{C}^{\boldsymbol{T}}\right)^{-1}\boldsymbol{C}\boldsymbol{M}_n\boldsymbol{A}^{\boldsymbol{T}} \tag{2.50}$$

After initialisation, the updating of $\boldsymbol{M}_n$ governs how the Kalman gain evolves over time and the strength of the filtering effect of the model (for further details on fitting the Kalman filter in practice see appendix). In a non-stationary case like the FFM (due to non-stationary training driving the process), initialisation of $\boldsymbol{M}_0$ falls into three categories: 1) "known" initialisation, where values are

set; 2) "approximate diffuse" initialisation, where $\boldsymbol{M}_0 = \kappa \boldsymbol{I}$ for large κ; 3) and "exact diffuse" initialisation which relies on limits as variances approach infinity (Fulton, 2017). The four stages of the Kalman filter can then be repeated with the estimated state and performance updated based on training input and corrections applied when performance is measured.

When using the Kalman filter, as is the case when fitting general FFMs, the model parameters must be estimated from training and performance data. This is achieved through algorithmically minimising some loss criterion, for example, the residual sum of squares between modelled and measured performance data, which, in the case of gaussian errors, coincides with the likelihood function (Mannakee *et al.*, 2016). With even the standard FFM, optimisation is analytically intractable and numerical procedures are used that require starting values for parameters $\{p^*, k_g, k_h, \tau_g, \tau_h\}$. While the available algorithms differ, they all are iterative in nature, stopping when the loss function falls below some prespecified threshold. When using the Kalman filter, the likelihood is available as a by-product of filtering operations (Fulton, 2017), and thus the Kalman filter can be fit with the same optimisation routines as the standard FFM with additional starting values for the extra parameters. Given that an entire run of the filtering algorithm is required to obtain the likelihood of the sample, a "double loop" results when paired with a numerical optimisation procedure, and time to convergence may be slow for long series. Additionally, given the additional parameters that must be estimated in the Kalman filter model given the same setup as the standard FFM, some "sloppiness" in parameter estimation is likely. The full $\boldsymbol{Q}$ matrix may also be challenging to recover, and difficulties with optimisation are discussed further in the chapter 6 (section 6.4).

Kolossa *et al.* (2017) outlined that more complex FFMs could be expressed as linear state-space models. The authors demonstrated that the VDR model could be easily expressed as a linear state space model and by including the Kalman filter could provide a means of addressing both the limitations of independence between training sessions and failure to take advantage of incoming data (Kolossa *et al.*, 2017). To express the VDR model as a linear-state space model a simple change is required from equation 2.45 where the control matrix $\boldsymbol{B}$ is updated each iteration according to the following equation:

$$\mathbf{B}_{\boldsymbol{n}} = \begin{pmatrix} e^{-\frac{1}{\tau_g}} \\ k_{h_2}^{i} \cdot e^{-\frac{1}{\tau_h}} \end{pmatrix} \tag{2.51}$$

Where $k_{h_2}(i)$ is given in equation 2.39.

In an accompanying experimental study, Kolossa *et al.* (2017) fitted model parameters to the training-performance profiles of 5 athletes (swimmers) with training load quantified daily according to a method previously used (Mujika *et al.*, 1996). Daily training load was taken from the sum of quantified

water and dry-land training. Water-based training was calculated by splitting distance (in km) swam over a session into five levels based on relative speed (i.e., intensity) and then multiplying these distances by an arbitrary weighting coefficient; taking the total. Land training was calculated via an approximation of its equivalence to water-based training. In summary, the training load method followed a basic volume by weighted intensity approach as has been common across the literature. In their experiment, the authors compare the performance of the VDR model with and without the use of a Kalman filter. The quality of the Kalman-filtered response was assessed by mean absolute percentage error (MAPE). For each athlete, an individual model was fitted across the entire range of available data, as well as just to the first half to allow forecast accuracy of the model to be evaluated through hold-out cross-validation. Finally, a generalised set of parameters were computed across all individuals (i.e., a multi-athlete model), with the intention of assessing whether sparse initial data of a new athlete can be augmented for computation of parameter estimates during an initial training period (Kolossa *et al.*, 2017). To achieve this, the optimisation process searches for a parameter set that minimises residual sum of squares over all five test subjects for both the entire data set and just the first half.

For individual models fitted across the entire range of available data, model fit in Kolossa *et al.* (2017) assessed via $MAPE_{KALMAN}$ ranged from 1.81-3.06% (mean = 2.31%), vs. 1.94-7.75% (mean = 3.35%) for the standard VDR model. For individual models fitted to half of the available data, $MAPE_{KALMAN}$ ranged from 1.95-6.69% (mean = 3.56%) vs. 2.07-7.05 (mean = 4.12%) for the standard VDR model, and forecast accuracy appeared visually reasonable across the second half of the data. However, no numerical error value was given specifically for the second half forecasting, although the MAPE values do reflect model output across the entire prediction series (fitted and forecast). Within their multi-level model (minimisation of RSS for all subjects to obtain generalised parameters), Kalman-filtering demonstrated a large reduction in MAPE compared to the standard VDR model (2.43 vs. 3.75%, respectively). What is interesting is that the generalised model fitted to half of the available data also performed better on average at forecasting the second half criterion performance values compared to an individual model fitted to half the available dataset. This implies that individual fitting of the FFM via a Kalman-filtering or standard approach may not be the best route for forecast accuracy in some instances, particularly when limited training and performance data is available for the subject. Multi-level models may offer a productive alternative. In summary, Kolossa *et al*. (2017) found the Kalman-filtered responses performed better at explaining large or rapid changes in observed performance, compared to the standard VDR model which struggled with high variation in observed performance. Conversely, the Kalman-filter has the disadvantage of being heavily reliant on experimental observations (Kolossa *et al.*, 2017). The authors novel FFM study and multi-level modelling approach presents a powerful and exciting perspective for adjusting for the variable nature of measured human performance. Future investigations should strongly consider this approach for further studies in applied settings, particularly where performance measurements are derived using methods of extrapolation

from sub-maximal effort (e.g., reps-in-reserve, load-velocity profiling). In chapter 6, an implementation of the Kalman-filtering process in R is followed and its potential use in future research is further discussed under the code resources developed.

2.4.4 Modification to the model system to include non-linearity

A common criticism of the standard FFM evolving from Banister and colleagues original model system (Banister *et al.*, 1975) is that it is based on linear systems theory (i.e., control systems constructed of linear differential equations), and that this limits accuracy given the observed non-linearity of most human phenomena and training response (Turner *et al.*, 2017). Turner *et al.* (2017) presented a novel refinement of Banister's model, introducing a generic mathematical framework to capture the non-linear effects of training (i.e., the problem of increased performance with arbitrary increases in training load and diminishing rates of return), enabling the search for optimum training programs in theory. The authors suggested that the standard FFM could be updated and specified as a system of non-linear differential equations for fitness and fatigue components, as follows:

$$g'(t) = k_g \cdot \omega(t) - \frac{1}{\tau_g} g(t)^{\alpha} \tag{2.52}$$

$$h'(t) = k_h \cdot \omega(t) - \frac{1}{\tau_h} h(t)^{\beta} \tag{2.53}$$

where α, β are power terms that represent the model's non-linearities, and where the original linear systems model can be recovered by simply setting $\alpha = \beta = 1$.

To explore the features of the non-linear systems model, Turner *et al.* (2017) first investigated the simplistic case of constant daily training (ω). They demonstrated that under this simple analysis model, a steady-state performance would be reached equal to:

$$p = p_0 + \left(k_g \tau_g \omega\right)^{1/\alpha} - \left(k_h \tau_h \omega\right)^{1/\beta} \tag{2.54}$$

And that the constant training load that caused maximum steady-state performance was equal to:

$$\omega_{max} = \left(\left(\frac{\beta}{\alpha}\right)^{\frac{\alpha}{\beta}} \frac{\left(k_g \tau_g\right)^{\beta}}{\left(k_h \tau_h\right)^{\alpha}} \right)^{1/(\beta - \alpha)} \tag{2.55}$$

A limitation of the non-linear model was identified when investigating uniform weekly schedules where the daily training load could fluctuate but the same weekly schedule was repeated. Turner *et al.*

(2017) demonstrated that repeated application of this weekly schedule created a periodic steady-state solution with the maximum obtained when the average training load was equal to that presented in equation 2.55. The analysis demonstrated under this constraint, the equivalence of different patterns of training loads within a week, if the average training load was the same. Therefore, the model addressed the limitations of assumed linearity in the standard FFM but not independence of subsequent training sessions.

Turner et al. (2017) also investigated the predictive validity of the non-linear model on a data set collected from a single cyclist. Training load data were collected over 532 days with performance measured across 18 sessions throughout the period. The non-linear model comprised seven parameters $\{p^*, \tau_g, \tau_h, k_g, k_h, \alpha, \beta, \}$ and was fit to 9 of the performance measures with cross-validation performed on the remaining 9 performances. The cross-validation process was repeated many times for 9 randomly selected performances with parameters estimated each time with a genetic algorithm. Despite the small number of performance sessions and the large number of training days over which the model was fit, good predictive agreement was obtained. Additionally, the repetition of obtaining different parameters across the randomly selected performances was used to assess sensitivity in parameter estimation. Relatively tight variation was identified across all parameters with values generally ranging from 0.5 to 1.5 times the average estimated value. Turner et al. (2017) also used the variation in parameter sets to generate uncertainty in performance predictions and presented this graphically using a normalised 2-D histogram. This model is further examined in chapter 6 with an emphasis on methods for parameter estimation in research and practice, and examination of the implications of the additional non-linear terms α, β.

2.4.5 A recursive delay-differential model

In a further attempt to address criticisms of the standard model for its lack of interaction between training sessions (independence assumption) Matabuena and Rodriguez-Lopez (2016; 2019) proposed a recursive delay-differential model. The authors developed a model variant where athletic performance depends on both current and previous training doses (Matabuena and Rodríguez-López, 2016, 2019). Extending the original model towards a delay-differential model, previous training sessions are accounted for inside of the recursion equations (Matabuena and Rodríguez-López, 2016, 2019). In their novel formulation, the Matabuena and Rodrigeuz-Lopez (2016; 2019) retained the overarching concept that performance was the antagonistic sum of fitness and fatigue, but added additional delays for the fitness and fatigue components (Matabuena and Rodríguez-López, 2016, 2019).

For a single day delay (denoted $d = 1$), indicated as a 1-day previous dose interaction, the authors expressed their model as the following system of differential equations (for the fitness $g(t)$ and fatigue $h(t)$ components, respectively):

$$\hat{p}(t) = p^* + k_g \cdot g(t) - k_h \cdot h(t) \tag{2.56}$$

$$g'(t) = \omega(t) - \frac{1}{\tau_{g_1}} g(t) - \frac{1}{\tau_{g_2}} g(t-1) \tag{2.57}$$

$$h'(t) = \omega(t) - \frac{1}{\tau_{h_1}} h(t) - \frac{1}{\tau_{h_2}} h(t-1)$$

From their system, Matabuena and Rodriguez-Lopez (Matabuena and Rodríguez-López, 2016, 2019) demonstrated that performance on day n as described by this system could be approximated using the following 7-parameter $\{p^*, k_g, k_h, \tau_{g_1}, \tau_{g_2}, \tau_{h_1}, \tau_{h_2}\}$ recurrence formula:

$$\hat{p}(t) = p^* + k_g \sum_{i=1}^{t-1} \left[\omega_i - \frac{1}{\tau_{g_2}} g(i-1) \right] e^{\frac{1}{\tau_{g_1}}(t-i)} - k_h \sum_{i=1}^{t-1} \left[\omega_i - \frac{1}{\tau_{h_2}} f(i-1) \right] e^{\frac{1}{\tau_{h_1}}(t-i)} \tag{2.58}$$

Where initial conditions $g(0) = h(0) = \omega_0 = 0$ were assumed if the athlete has not trained for a while prior to the intervention period. Otherwise, these initial conditions require estimation within the fitting process or derivation from a previous modelling period.

Any number of delays can theoretically be added within Matabuena and Rodriguez-Lopez (2016; 2019) system, although this is not likely to be practical with consideration to the required number of data points to fit the additional parameters and the resultant complexity of the search space. With each additional delay the number of parameters in the model compared to the standard formulation (eq. 2.9) increases by two (Matabuena and Rodríguez-López, 2016, 2019), and this is likely to increase the difficulty of finding the global optima. The authors suggest that no more than 3 delays are likely to be useful in practice, and that interactions between standard training sessions are unlikely to feature beyond three days (Matabuena and Rodríguez-López, 2016, 2019). A special case where this may not apply conceptually is when sessions are extremely depleting or acutely fatiguing, for example long endurance sessions. However, these types of sessions are unlikely to be high frequency and so it may be suitable to forego modelling this longer interaction. Model fit appeared to be extremely strong (R^2 = 0.99) for $d = 1$ (single delay) compared to the standard model (eq. 2.9) in a single case study of their model based on data of a cyclist provided within Clarke and Skiba's supplementary data file (Clarke and Skiba, 2013). However, the classic limitation of this supplementary data file, the low

number of available data points for model fitting and no extra data available for out-of-sample testing, reduces the conclusions that can be drawn from this experimental application. In series notation, the general differential systems with $d \in \mathbb{N}$ delays can be represented as follows:

$$g'(t) = \omega(t) - \frac{1}{\tau_{g_1}} g(t) - \frac{1}{\tau_{g_2}} g(t-1) - \cdots - \frac{1}{\tau_{g_{(d+1)}}} g(t-d) \tag{2.59}$$

$$h'(t) = \omega(t) - \frac{1}{\tau_{h_1}} h(t) - \frac{1}{\tau_{h_2}} h(t-1) - \cdots - \frac{1}{\tau_{h_{(d+1)}}} h(t-d) \tag{2.60}$$

As a final example, for $d = 3$, the authors showed that performance on day n can be approximated by the following recurrence relation:

$$\begin{aligned}\hat{p}(t) = p^* + k_g \Bigg[& \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_{g_1}}} - \sum_{i=1}^{t-3} g(i-1) \left(\frac{1}{\tau_{g_2}} + \frac{1}{\tau_{g_3}} e^{\frac{1}{\tau_{g_1}}} + \frac{1}{\tau_{g_4}} e^{\frac{2}{\tau_{g_1}}} \right) e^{\frac{-(t-i)}{\tau_{g_1}}} \\ & - g(i-3) \left(\frac{1}{\tau_{g_2}} e^{-\frac{2}{\tau_{g_1}}} + \frac{1}{\tau_{g_3}} e^{-\frac{1}{\tau_{g_3}}} \right) - g(i-2) \left(\frac{1}{\tau_{g_2}} e^{-\frac{1}{\tau_{g_1}}} \right) \Bigg] \\ & - k_h \Bigg[\sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_{h_1}}} - \sum_{i=1}^{t-3} h(i-1) \left(\frac{1}{\tau_{h_2}} + \frac{1}{\tau_{h_3}} e^{\frac{1}{\tau_{h_1}}} + \frac{1}{\tau_{h_4}} e^{\frac{2}{\tau_{h_1}}} \right) e^{\frac{-(t-i)}{\tau_{h_1}}} \\ & - h(i-3) \left(\frac{1}{\tau_{h_2}} e^{-\frac{2}{\tau_{h_1}}} + \frac{1}{\tau_{h_3}} e^{-\frac{1}{\tau_{h_1}}} \right) - h(i-2) \left(\frac{1}{\tau_{h_2}} e^{-\frac{1}{\tau_{h_1}}} \right) \Bigg]\end{aligned} \tag{2.61}$$

2.4.6 Time-varying model: Recursive least squares

Under time invariant FFM frameworks, such as the standard model, a unique parameter set is identified or specified to characterise a single time period. This is in contrast to a time-varying framework which includes multiple sets of parameters that are recursively updated over a single time-period in the presence of new data. In 1997, Busso and colleagues proposed a time-varying FFM, and assessed whether this would provide better model fit to observed data (Busso *et al.*, 1997). In particular, the authors hoped that the time-varying model would do a better job at describing variation in performance change during key tapering periods, and when fatigue had accumulated following intense training. First, recall that the standard model in closed form is described mathematically as below (eq. 2.62). In this case, with the inclusion of a decaying initial components (q_g, q_h) to account for the accumulation of prior training, for fitness and fatigue respectively.

$$\hat{p}(t) = p^* + \left(q_g \cdot e^{-\frac{t}{\tau_g}}\right) - \left(q_h \cdot e^{-\frac{t}{\tau_h}}\right) + k_g \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_g}} - k_h \cdot \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_h}} \qquad (2.62)$$

Next, recall that to fit a time invariant FFM via a non-linear least squares approach, the following least-squares objective function is minimised:

$$S_t = \sum_{i \in \Omega} (\hat{p}_i - p_i)^2 \qquad (2.63)$$

Where Ω is a set of (generally) non-consecutive time-points $[i_1, i_n]$ $(n \in \mathbb{N})$ from the time period $[t_0, t]$, each element representing a position at which a real-world performance measurement has been taken; t_0 is also the position of the first training load value in the time period of interest. This concept is illustrated graphically below in Figure 2.9.

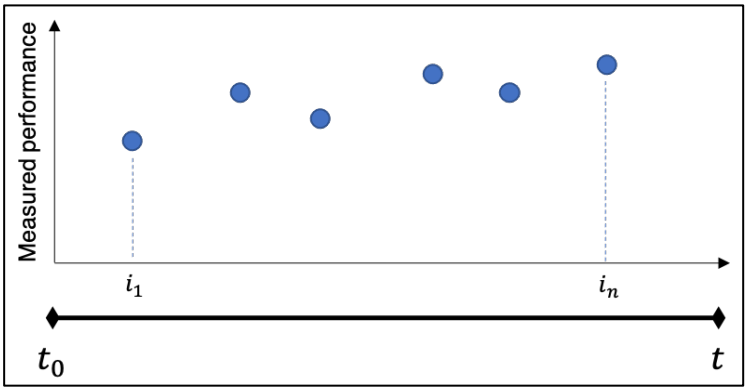


Figure 2.9: Graphical representation of series data used when fitting the time invariant FFM via a non-linear least squares approach

In contrast, a recursive least-squares algorithm is a type of adaptive filter, similar to the Kalman filter, where model parameters are estimated each time new data is collected. The process involves recursively searching for coefficient estimates that minimise a weighted least-squares cost function. It is computationally complex, but convergence rates are typically good. In the time-varying FFM, the effect of new data is artificially emphasised by exponentially weighting past-values such that on day η (where η corresponds to some day on which performance was measured) fitting the model is performed by minimising the following recursive function:

$$S_\eta = S_{\eta-1}\alpha + S_\eta \qquad (0 < \alpha < 1) \qquad (2.64)$$

A small value for the weighting parameter α in the recursive formulation of the standard model was shown to allow rapid change in model parameters in the standard model, but made them sensitive to noise in the criterion performances (Busso *et al.*, 1997). A large value of α was shown to limit the ability of the fitting process to follow variations in parameters, but helped to reduce this sensitivity (Busso *et al.*, 1997). In their study, Busso *et al*. (1997) selected a value of $\alpha = 0.9$ to limit the influence noise would have in the parameter values. Freedom given to the parameters to vary over time proved, as the authors expected, to result in a reduction in the residuals of the cost function (RSS), resulting in better model fit (R^2 = 0.875-0.879, $n = 2$) compared to the time-invariant formulation (R^2 = 0.666-0.682, $n = 2$). In a subsequent experiment, strong model fit (R^2 = 0.957-0.982, $n = 6$) was observed for the time-varying formulation under two configurations (time-varying scaling and decay parameters, and time-varying scaling parameters only), and under slightly improved measurement frequency conditions (40-46 per subject over 15 weeks). However, the authors did not include the time-invariant formulation for comparison (Busso *et al.*, 1997).

While this time-varying model appears to better describe observed data, it lacks a clear method that allows its use to generate model predictions for comparison with out-of-sample data. In particular, there is no specified direction set, and therefore the proposed time-varying model in its current form is simply another method of obtaining improved model fit, and lacks clear predictive benefits unless it were possible to predict change in the parameters themselves (Pfeiffer, 2008). Conceptually, however, the time-varying model may be more realistic if attempting to build general principles of the way individuals respond to inform future training, due to its recursive nature and likelihood of training sessions interacting with each other. It may be that time-varying models could be used to plan the final two-weeks of tapering prior to competition if models are built over the whole prior training cycle.

2.4.7 An exponential growth model

As discussed so far, a maximal effect immediately after a training session may be conceptually appropriate for fatigue, but is unlikely to correctly reflect the fitness response to training (Calvert *et al.*, 1976; Philippe *et al.*, 2018; Rasche and Pfeiffer, 2019). In particular, the physiological processes of adaptation (and subsequent effects on performance) are not instantaneous and require time to resolve (Calvert *et al.*, 1976). In 2018, Philippe and colleagues suggested that exponential growth kinetics may be more appropriate to describe the biological processes that occur in response to physical training (Philippe *et al.*, 2018). However, a limitation of incorporating these dynamics is that increases in degrees of freedom increases the frequency and total number of measures required to adequately fit the model (Philippe *et al.*, 2018). Philippe and colleagues attempted to test the addition of an exponential growth term within the positive component of the classical Banister model of training effects (Philippe *et al.*, 2018).

Recall that in a general FFM, performance at time t is found from convolving the transfer function $H(t,\theta)$ with the training load values $\omega(t)$, where θ denotes the set of free parameters characterising the model behaviour and p^* is the additive term; written as:

$$\hat{p}(t) = p^* + \omega(t) * H(t,\theta) \tag{2.65}$$

where:

$$\omega(t) * H(t,\theta) = \int_0^t \omega(s) \cdot H(t-s,\theta)ds \tag{2.66}$$

In a general FFM, performance results from the difference of two antagonistic components (fitness and fatigue), which evolve in parallel, and thus the general transfer function is characterised by the set of model parameters with two gain and two decay time constants:

$$H(t,\theta) = k_g \cdot e^{-\frac{t}{\tau_g}} - k_h \cdot e^{-\frac{t}{\tau_h}} \tag{2.67}$$

where $\theta = \{k_g, \tau_g, k_h, \tau_h\}$.

Philippe and colleagues proposed two exponential-type models, the first of which did not account for fatigue and so the general transfer function $H(t,\theta)$ is defined by a serial and bi-exponential function ascribed to the fitness component, as follows:

$$H(t,\theta) = k_{g_1} \cdot \left(1 - e^{-\frac{t}{\tau_{g_1}}}\right) \cdot U + k_{g_2} \cdot \left(e^{-\frac{(t-TD)}{\tau_{g_2}}} \cdot |U-1|\right) \tag{2.68}$$

Where, $U = 1$ when $t < TD$, and $U = 0$ when $t \geq TD$, where $TD = 4 \cdot \tau_{g_1}$. TD is the abbreviation for 'time-delay' and represents the time-to-completion of the growth phase before the decay phase can begin.

In the authors second exponential model, the transfer function $H(t,\theta)$ is defined by the summation of the serial bi-exponential component ascribed to fitness (positive effects), with an impulse-response function incorporated for 'fatigue' (negative effects). This is equivalent conceptually to the early delay model by Calvert et al. (Calvert *et al.*, 1976). Both components evolve in parallel, such that $H(t,\theta)$ is defined by:

$$H(t,\theta) = k_{g_1} \cdot \left(1 - e^{-\frac{t}{\tau_{g_1}}}\right) \cdot U + k_{g_2} \cdot \left(e^{-\frac{(t-TD)}{\tau_{g_2}}} \cdot |U-1|\right) - k_h \cdot e^{-\frac{t}{\tau_h}} \tag{2.69}$$

Where the additional parameters (k_h, τ_h) in comparison to the first exponential-type model (eq. 2.67) are the scaling and decay constants for the impulse-response component ascribed to fatigue. The other parameters are described as above.

No discrete solution is provided for the proposed models, and they can be solved in a similar fashion to Turner *et al.* (2017), via numerical solvers incorporated into an optimisation process. To experimentally assess their proposed modifications, the authors conducted a rodent $(n = 15)$ resistance training intervention, inclusive of 4-weeks daily training and performance measurements (Philippe *et al.*, 2018). The models were fit to the entire group of rodents (pooling the sample), using a mixed-effects model with a systematic component for population mean response and a random component for each rodent's response around the mean. The time-constants were estimated for the pooled model using a genetic algorithm, whereas the multiplicative factors were subject-specific and estimated via iterative minimisation of the mean of the squared errors between modelled and measured performance values (i.e., $\frac{RSS}{n_r}$, where n_r is the number of rodents and RSS is as shown in eq. 2.33 and eq. 2.63). Goodness-of-fit analysis (Adjusted R^2 and statistical significance assessed by analysis-of-variance of RSS in accordance with the degrees of freedom of each model) allowed the authors to test the models of different complexity adjusted for the influence of more free parameters. Decreases in RSS explained by the introduction of more free parameters in the model was tested using the F-ratio test. Model fit was significant for all models and ANOVA revealed improvements in fit for the two complex models (Standard impulse-response and exponential model with impulse-response (eq. 2.69) compared to the simple exponential model for fitness only (eq. 2.68). However, the time-constants for the standard impulse-response model were very similar (48.3 and 49.1 days for fitness and fatigue, respectively) and the authors highlighted that these do not seem realistic and not appropriate in the context of their experiment. For the exponential model with impulse-response ascribed to fatigue, model fit was only significantly increased for 6 of the 15 rodents ($P < 0.001$) and the time-constant ascribed to this component was not realistic (49.9 days) (Philippe *et al.*, 2018). The proposed models are certainly an interesting avenue for future research; however, the model is not significantly dissimilar to the fitness-delay model by Calvert *et al.* (1976) for which a closed solution already exists and has not been adequately explored in experimental research.

2.4.8 Secondary-signal model

In the standard two-component fitness-fatigue model, fitness and fatigue components are modelled in an identical fashion using first-order kinetics, with each component increasing as a function of its respective scaling factor, and in response to a training session decaying away at a rate determined by its time-constant (Busso, 2017). These dynamics are widely considered conceptually intuitive as a starting effort to describe the changes in physical condition with training (Chiu and Barnes, 2003).

However, it has been suggested that physical training should be viewed as the primary stimulus for training-induced adaptations that in turn activate secondary signals, and it is these secondary signals that are responsible for driving adaptations, dissipate during post exercise recovery, and the accumulation of which increases production of a training effect that counterbalances loss of adaptation (Busso, 2017). To this end, Busso (2017) developed a series of secondary signal models which consider fatigue as both the counterbalance to the positive response from training and also as an inhibitory influence on training-induced adaptations. Busso constructed models which can distinguish between acute fatigue affecting performance, and maladaptation due to excessive loads (Busso, 2017). The common structure amongst the four proposed secondary-signal models proposed is that performance is considered the cumulative sum of responses to each training bout over time. Each response to a training bout is considered to be the result of an indirect mechanism which could stimulate or inhibit the product of a training effect, and is counterbalanced by its dissipation (Busso, 2017). In the author's basic secondary-signal model, the response to training is described by the production of a secondary signal ($Signal$) which is the mediator for change in performance through production of fitness effects ($Prod$) which counterbalances removal. In this model, greater rates of removal occur when performance is elevated. Mathematically, this was described by the following set of recursive equations:

$$\hat{p}(t) = \left(p(t-1) \cdot e^{-k_{off}}\right) + Prod(t-1) \tag{2.70}$$

Where:

$$Prod(t) = k_{on}^{0} + \left(k_{on}^{s} \cdot Signal(t)\right) \tag{2.71}$$

And:

$$Signal(t) = Signal(t-1) \cdot e^{(k_{out}^{s} - k_{on}^{s})} + \omega_t \tag{2.72}$$

Where k_{off} is the first-order rate constant for loss of performance. k_{on}^{0} is a parameter representing the rate of production of performance with no signal (i.e. no training) and which is set so that performance and removal are balanced at baseline performance p^*, that is: $k_{on}^{0} = p^* \cdot \left(1 - e^{-k_{off}}\right) \cdot k_{on}^{s}$ is the first-order rate constant that transforms the secondary signal into performance, with k_{out}^{s} the parameter determining the rate at which the signal dissipates. Finally, $\omega(t)$ represents the training load on day t. To extend this basic secondary-signal model, Busso then included a variable function term ($Inhib$) which diminishes the production of performance proportional to the training dose (Busso, 2017), as follows:

$$Inhib(t) = k_{in}^{t} \cdot \omega_t \tag{2.73}$$

Where k_{in}^{t} is the constant of proportionality, and therefore:

$$Signal(t) = Signal(t-1) \cdot e^{-\left(k_{out}^{s} - k_{on}^{s} \cdot (1 - Inhib(t))\right)} + \omega_t \quad (2.74)$$

$$Prod(t) = k_{on}^{0} + \left(k_{on}^{s} \cdot Signal(t) \cdot \left(1 - Inhib(t)\right)\right) \quad (2.75)$$

In a further model, a fatigue component is added represented by an equation computed as follows:

$$Fatigue(t) = \left(k_{in}^{f} \cdot \omega_{t-1} + Fatigue(t-1)\right) \cdot e^{k_{out}^{f}} \quad (2.76)$$

Where k_{in}^{f} is the rate-constant at which training results in an increase in fatigue, and k_{out}^{f} is the rate at which it dissipates. Initial conditions are set to $Fatigue(0) = \omega(0) = 0$.

Modelled performance then becomes:

$$\hat{p}(t) = \left[\left(p(t-1) \cdot e^{-k_{off}}\right) + Prod(t-1)\right] - Fatigue(t) \quad (2.77)$$

Busso (2017) attempted to experimentally validate his model, using data from a previous experimental study (Busso, 2003). However, as is common across the literature, the authors did not assess any measure of out-of-sample prediction accuracy. Across the three secondary signal models proposed, goodness-of-fit (coefficient-of-determination) adjusted for degrees of freedom (number of parameters in the model) ranged from R^2_{adj} = 0.826-0.967, demonstrating strong model fit under all three proposed secondary-signal models. Robust evaluation of this model is required under further datasets and using out-of-sample data to assess forward prediction accuracy or association with underlying physiological mechanisms. However, there has been no subsequent experimental work on this model since the authors initial paper (Busso, 2017).

2.5 Review summary

Modelling complex human processes is not straightforward. The scientific field of systems biology is at the forefront of modern inquiry and leading the way in several aspects of systems modelling. Even so, systems biologists are frequently confronted by and still make use of heavily parameterised, incomplete, or partially inaccurate mathematical models of complex processes (Transtrum *et al.*, 2015). Moreover, with any mathematical model of the real world there exists an accuracy vs. complexity trade-off, with misspecification inevitable. With the use of modelling to describe complex real-world systems now viewed as an important component of modern scientific investigation, it is hasty to dismiss one of the only existing systems models in sport science without rigorous study, at the very least to inform and guide future modelling attempts (Transtrum *et al.*, 2015). Sport science

has very few models to describe the interaction of complex human processes underpinning training and performance (Arandjelovic, 2013; Arandjelović, 2017; Herold and Sommer, 2020; Jeffries *et al.*, 2020), and this art currently exists at the fringe of our existing literature body and outside the purview of most practitioners. This literature review, and associated publications (Stephens Hemingway *et al.*, 2021; Swinton *et al.*, 2021) have attempted to present a methodical history and refreshed perspective of the study of these models and bring this area of research back into focus. Whilst FFMs are not exempt from common challenges associated with complex systems modelling, their potential utility and notional value reside in being theory-driven mathematical models of athletic performance from cumulative training loads, with counterfactual properties and the ability to generate testable predictions (Pearl, 2010). Furthermore, as presented in section 2.4 (model developments), there have been several valuable and informative attempts within the literature to address existing limitations and common criticisms within the literature body. These model extensions must be studied and tested on their own merit, rather than grouped with prior criticisms of the standard model structures. It is argued that these model extensions have been largely ignored with respect to further experimental investigation due to historical criticisms of the original models, and a lack of awareness amongst sport science researchers of several recent and innovative advancements in this research area in the last decade. Furthermore, despite certain theoretical shortcomings of the standard model structure, it forms the kernel of most FFM systems and therefore remains worthy of continued study, particularly in areas such as model parameter estimation and where findings may inform methods to be used with other models. Insights from study of the standard model may also provide enhanced understanding of the lower bound requirements on study design elements such as fitting data (e.g., volume, frequency, error), and be used to demonstrate theoretical approaches of their use to inform training program design.

To explain the lack of out-of-sample assessment of model predictions across fitted models, one suggestion is prior insufficiency in the availability and access to high-quality and high-frequency long term training and performance data. In the past, some authors have even turned to conducting interventions on themselves as sole participants to generate sufficient data to conduct FFM research (Banister, Morton and Fitz-Clarke, 1992). It has only been in the last decade or so that data collection on training load and performance has substantially increased, making these types of investigations far more feasible across many sporting domains. Notably, some recent studies have begun to take advantage of the opportunity to utilise convenient field data with FFMs (Ludwig and Asteroth, 2016; Kolossa *et al.*, 2017; Rozendaal, 2017; Williams *et al.*, 2018; Scarf *et al.*, 2019; Mitchell *et al.*, 2020). However, the majority of historical experimental research has focussed on quantifying relationships between underlying physiological processes of training, through iterative refinement of model structure and comparison with observed physiological phenomena (Busso and Thomas, 2006). Although there may still be value in this approach, there is an overwhelming volume of this type of

work compared to the study of FFM predictions, and attempts should now be made to restore a balance between these two modelling objectives.

To improve the practical application of future research, more effective collaboration between researchers and practitioners is needed and development required in each of the three practical areas highlighted in this literature review (training load quantification, criterion performance selection, parameter estimation). With regards to training load quantification, greater discussion of how to selectively weight different training modalities, exercises and intensities on the basis of the target performance is needed. Additionally, researchers may consider including training load quantification parameters as part of the overall fitting process, potentially with a priori constraints suggested by practitioners. With regards to criterion performance measures, the biggest challenge with fitness-fatigue modelling appears to be the frequency of measurement. It is hypothesised that a method that seems most suited to generating useful high frequency measurements, particularly with regards to fitness aspects, is individually predicted maximum performances derived from actual training data. Given the large increase in applied research in this area (e.g., reps-in-reserve, barbell-velocity profiling) there are a range of candidate measurements that could be used and would benefit from the collaboration between researchers and practitioners. Finally, with regards to parameter estimation there is a clear requirement for extensive further development in methods and understanding. Of all areas of the modelling process, parameter estimation, associated model evaluation, and reporting are the least well described aspects. In addition to the systematic study of which algorithms perform the best for obtaining suitable parameters under a range of conditions, there is a need to explore in greater depth issues such as parameter stability and estimation sensitivity, and methods to obtain parameters that are best able to predict future training such as the potential of tuning-based cross-validation and the introduction of regularisation terms. Future collaborations between researchers and practitioners may wish to consider setting a priori constraints on likely parameter values given the specific athlete (alternatively using informative priors via a Bayesian approach) or matching athletes and fitting multiple models simultaneously sharing information to obtain parameter values that are representative of athletes with similar adaptive responses. As with other practical areas of fitting FFMs, there is considerable scope for ingenuity and interdisciplinary work that will eventually require and could utilise large amounts of data routinely collected by practitioners working with athletes.

There is also scope for researchers and practitioners to more creatively apply the standard FFM. In the context of resistance training and the development of fitness aspects, the FFM has been used in its conceptual form to highlight that different forms of training (e.g. strength, power, and hypertrophy) appear to generate distinct fitness and fatigue profiles (Chiu and Barnes, 2003). The general FFM with N additive first-order components and other properties such as fitness-delays, variable dose-response terms, and threshold saturation, may provide a suitable mathematical framework to model these

empirical observations with for example strength, power and hypertrophy training sessions inputting their effects through specific first-order components with scaling coefficients and decay time constants dependent on the fitness attribute being modelled. With all such innovative procedures, it is critical that emphasis is placed on determining the conditions under which suitable predictions to future training can be obtained.

Finally, it is clear from this review that historical and future research would benefit from increased transparency and reproducibility around data, assumptions, and methods of implementation. It is therefore recommended that where possible future experimental study follow basic tenants of reproducible research when disseminating work in peer-reviewed journals and preprint (Peng, 2011). These include: 1) documentation of code/software used to implement the models and perform the analysis; 2) presentation of computational output with explanatory text; 3) inclusion of raw data alongside code and analysis to facilitate verification and reproduction of findings and methods. Where public release of raw or anonymised data is unrealistic due to reasonable concerns for privacy or commercial sensitivity, authors should attempt to make this available to reviewers within the peer-review process where possible. Code notebooks or analysis files (e.g., written in the markdown language) hosted on public or private repositories may provide convenient and simple methods for achieving the recommendations outlined above. The aim of this literature review was to provide a detailed insight into the historical development of and practical considerations surrounding fitness-fatigue modelling, and to present the background underlying the justification for the research both within this thesis and beyond. While acknowledging several limitations in the standard model structure and criticisms across prior research, it is the position taken in this work that sport science should not yet give up on FFMs until they have been robustly studied, particularly given several promising approaches by researchers to address prior limitations in recent times. To achieve progress, this small field of sport science must pool ideas, concepts, and develop communal tools that spur collaborative efforts. Establishment of a coherent framework for future study would be a further helpful step, particularly one that prioritises tackling specific factors currently limiting model utility, and that outlines steps for rigorous assessment of new and existing models/methods.

2.5.2 A visual timeline of FFM developments

To summarise the literature body and historical journey of FFMs discussed in this review, a graphical timeline is presented in Figure 2.10. This timeline spans the initial conception of the FFM in 1975 to key literature up to the current point of this thesis. Complete summary tables of the historical literature are also included in Appendix B, to provide a quick reference guide to - and overview of - the key literature.

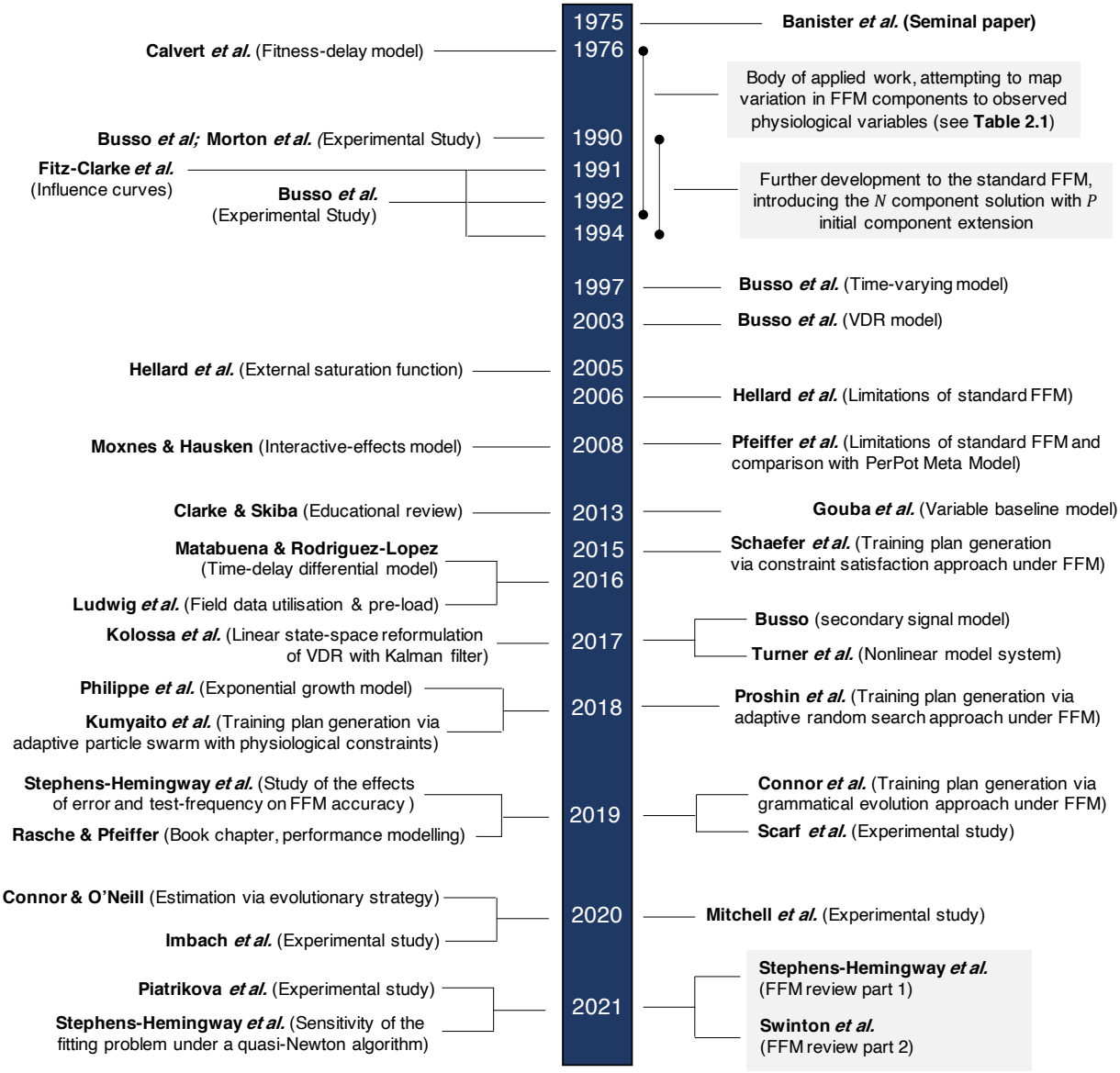


Figure 2.10: A timeline of model development

2.5.3 Reference Table (FFM formulae)

Table 2.4: A quick reference table of FFM formulae

Model	Type	Formula
Basic model	System of ODE's	(Banister *et al.*, 1975) One component (i.e., fitness or "training" only) $$p(t) = p^* + Kp(t)$$ $$p'(t) = \omega(t) - \frac{1}{\tau}p(t)$$
Standard model	System of ODE's	(Banister *et al.*, 1975) Two components (i.e., fitness and fatigue) $$p(t) = p^* + k_g \cdot g(t) - k_h \cdot h(t)$$ $$g'(t) = \omega(t) - \frac{1}{\tau_g}g(t)$$ $$h'(t) = \omega(t) - \frac{1}{\tau_h}h(t)$$
Fitness-delay model	System of ODE's	(Calvert *et al.*, 1976) Two components (i.e., fitness (+ delay) and fatigue) $$p(t) = p^* + k_g \cdot g(t) - k_h \cdot h(t)$$ $$\left(\frac{1}{\tau_{g_2}} - \frac{1}{\tau_{g_1}}\right)^{-1} g''(t) = \omega(t) - \left(\frac{1}{\tau_{g_2}} - \frac{1}{\tau_{g_1}}\right)^{-1}\left(\frac{1}{\tau_{g_1}} + \frac{1}{\tau_{g_2}}\right)g'(t) - \left(\frac{1}{\tau_{g_2}} - \frac{1}{\tau_{g_1}}\right)^{-1}$$ $$h'(t) = \omega(t) - \frac{1}{\tau_h}h(t)$$
Non-linear model	System of ODE's	(Turner *et al.*, 2017) Two components (i.e., fitness and fatigue) $$p(t) = p^* + k_g \cdot g(t) - k_h \cdot h(t)$$ $$g'(t) = \omega(t) - \frac{1}{\tau_g}g(t)^{\alpha}$$ $$h'(t) = \omega(t) - \frac{1}{\tau_h}h(t)^{\beta}$$ where α, β are power terms that represent the model's non-linearities. The standard model system can be recovered by $\alpha = \beta = 1$

Model	Type	Formula
Delay-differential model	System of DDE's	(Matabuena and Rodríguez-López, 2016, 2019) (Single delay, $d = 1$) $$\hat{p}(t) = p^* + k_g \cdot g(t) - k_h \cdot h(t)$$ $$g'(t) = \omega(t) - \frac{1}{\tau_{g_1}} g(t) - \frac{1}{\tau_{g_2}} g(t-1)$$ $$h'(t) = \omega(t) - \frac{1}{\tau_{h_1}} h(t) - \frac{1}{\tau_{h_2}} h(t-1)$$
Exponential-growth model	General convolution model	(Philippe *et al.*, 2018) General convolution model: $$\hat{p}(t) = p^* + \omega(t) * H(t,\theta)$$ Where performance at time t is found by convolving the transfer function $H(t,\theta)$ with the training load values $\omega(t)$: $$\omega(t) * H(t,\theta) = \int_0^t \omega(s) \cdot H(t-s,\theta) ds$$ Where θ denotes the set of free parameters characterising model behaviour and p^* is the normal additive term. **Transfer function 1:** A serial and bi-exponential function ascribed to the fitness component (no fatigue): $$H_1(t,\theta) = k_{g_1} \cdot \left(1 - e^{-\frac{t}{\tau_{g_1}}}\right) \cdot U + k_{g_2} \cdot \left(e^{-\frac{(t-TD)}{\tau_{g_2}}} \cdot |U-1|\right)$$ Where $U = 1$ when $t < TD$, $U = 0$ when $T \geq TD$, and (time-delay) $TD = 4\tau_{g_1}$ **Transfer function 2:** Serial and bi-exponential function ascribed to fitness component, impulse-response component for fatigue (conceptually equivalent to fitness-delay model): $$H_2(t,\theta) = \underbrace{k_{g_1} \cdot \left(1 - e^{-\frac{t}{\tau_{g_1}}}\right) \cdot U + k_{g_2} \cdot \left(e^{-\frac{(t-TD)}{\tau_{g_2}}} \cdot |U-1|\right)}_{fitness} - \underbrace{k_h \cdot e^{-\frac{t}{\tau_h}}}_{fatigue}$$

Model	Type	Formula
Standard model	Closed form approximation	(Banister *et al.*, 1975; Morton, Fitz-clarke and Banister, 1990; Busso *et al.*, 1992) $$\hat{p}(t) = p^* + k_g \cdot \sum_{i=1}^{t-1} \omega_i \cdot e^{-\frac{(t-i)}{\tau_g}} - k_h \cdot \sum_{i=1}^{t-1} \omega_i \cdot e^{-\frac{(t-i)}{\tau_h}}$$ Alternatively, this can be written with a single summation as follows: $$\hat{p}(t) = p^* + \sum_{i=1}^{t-1} \left(k_g \cdot e^{-\frac{(t-i)}{\tau_g}} - k_h \cdot e^{-\frac{(t-i)}{\tau_h}} \right) \cdot \omega_i$$ Or in recursive form: $$\hat{p}(t) = p^* + \left(k_g \cdot g(t)\right) - \left(k_h \cdot h(t)\right)$$ $$g(t) = g(t-1) \cdot e^{\frac{-1}{\tau_g}} + \omega_t \qquad g(0) \geq 0$$ $$h(t) = h(t-1) \cdot e^{\frac{-1}{\tau_h}} + \omega_t \qquad h(0) \geq 0$$ The standard model could also be adjusted to incorporate a single scaling factor to address criticisms (Hellard *et al.*, 2006; Pfeiffer, 2008) of parameter interdependence: $$\hat{p}(t) = p^* + K \cdot \sum_{i=1}^{t-1} \left(e^{-\frac{(t-i)}{\tau_g}} - e^{-\frac{(t-i)}{\tau_h}} \right) \cdot \omega_i$$
Fitness-delay model	Closed form approximation	(Calvert et al., 1976) $$\hat{p}(t) = p^* + k_g \cdot \sum_{i=1}^{t-1} \omega_i \cdot \left(e^{-\frac{(t-i)}{\tau_{g_1}}} - e^{-\frac{(t-i)}{\tau_{g_2}}} \right) - k_h \cdot \sum_{i=1}^{t-1} \omega_i \cdot e^{-\frac{(t-i)}{\tau_h}}$$
The general model – *N* components	Closed form approximation (general form)	(Busso, Carasso and Lacour, 1991) $$\hat{p}(t) = p^* + \sum_{r=1}^{N} \left(k_r \cdot \sum_{i=1}^{t-1} \omega_i \cdot e^{-\frac{(t-i)}{\tau_r}} \right)$$ $$N \in \mathbb{N}; (r = 1, \dots, N)$$
The general model with *P* initial components	Closed form approximation (general form)	(Busso *et al.*, 1992) $$\hat{p}(t) = p^* + \sum_{r=1}^{N} \left(k_r \cdot \sum_{i=1}^{t-1} \omega_i \cdot e^{-\frac{(t-i)}{\tau_r}} \right) + \sum_{p=1}^{P} q_p^0 \cdot e^{-\frac{t}{\tau_p}}$$ Where q_p^0 denotes the initial level of the component, and τ_p is the decay time constant on the corresponding initial level.

Model	Type	Formula
VDR	Closed form approximation	(Busso, 2003) $$\hat{p}(t) = p^* + k_g \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_g}} - k_h \sum_{i=1}^{t-1} k_{h_2}(i) \cdot e^{\frac{-(t-i)}{\tau_h}}$$ Where the first-order filter $k_{h_2}(i)$ is calculated by a series of decaying exponentials with scaling factor $k_h \geq 0$ and time constant $\tau_{h_2} \geq 0$, such that: $$k_{h_2}(i) = \sum_{j=1}^{i} \omega_j \cdot e^{\frac{-(i-j)}{\tau_{h_2}}}$$ **As a recursive system** (Rasche and Pfeiffer, 2019) $$\hat{p}(t) = p^* + \Big(k_g(g(t) - \omega_t)\Big) - \big(k_h \cdot h(t)\big)$$ $$g(t) = g(t-1) \cdot e^{-\frac{1}{\tau_g}} + \omega_t$$ $$h(t) = \Big(h(t-1) + \omega_{(t-1)} \cdot h_2(t-1)\Big) \cdot e^{-\frac{1}{\tau_h}}$$ $$h_2(t) = h_2(t-1) \cdot e^{-\frac{1}{\tau_{h_2}}} + \omega_t$$
External threshold saturation	-	(Krzyzanski, Perez-Ruixo and Vermeulen, 1999; Hellard *et al.*, 2005) $$Hill(\omega) = \kappa\left(\frac{\omega^\gamma}{\delta^\gamma + \omega^\gamma}\right)$$ Hill function parameters γ, δ, κ Specific and fatigue component training load functions: $$\omega_g(i) = \kappa_g\left(\frac{\omega_i^{\gamma_g}}{\delta_g^{\gamma_g} + \omega_i^{\gamma_g}}\right), \qquad \omega_h(i) = \kappa_g\left(\frac{\omega_i^{\gamma_h}}{\delta_h^{\gamma_h} + \omega_i^{\gamma_h}}\right)$$ Combining the VDR model with Hill saturation and fitness-delay: $$\hat{p}(t) = p^* + k_g \sum_{i=1}^{t-1} \omega_g(i) \cdot \left(e^{-\frac{(t-i)}{\tau_{g_1}}} - e^{-\frac{(t-i)}{\tau_{g_2}}}\right) - k_h \sum_{i=1}^{t-1} k_{h_2}(i) \cdot e^{\frac{-(t-i)}{\tau_h}}$$ $$k_{h_2}(i) = \sum_{j=1}^{i} \omega_h(j) \cdot e^{\frac{-(i-j)}{\tau_{h_2}}}$$ With model parameters $(p^*, k_g, \tau_{g_1}, \tau_{g_2}, k_h, \tau_{h_2}, \tau_h)$ and external Hill function parameters $(\kappa_g, \delta_g, \gamma_g, \kappa_h, \delta_h, \gamma_h)$

Model	Type	Formula
Baseline performance model	Closed form approximation	(Gouba *et al.*, 2013) $\hat{p}_{t_j} = p_{t_{j-1}} + k_g \sum_{i=t_{j-1}}^{t_j-1} \left(e^{-\frac{(t_j-i)}{\tau_g}} \cdot \omega_i \right) - k_h \sum_{i=t_{j-1}}^{t_j-1} \left(e^{-\frac{(t_j-i)}{\tau_h}} \cdot \omega_i \right)$ Where $j \geq 1, \Delta_i = 1$, j is the number of measured performances, t_j is the day of the j-th measured performance, and $p_{t_{j-1}}$ is the observed performance value on t_{j-1}.
Kalman filtering (Standard and VDR models)	State-space model	(Kolossa *et al.*, 2017) $\mathbf{x}_{n+1} = \mathbf{A}_n \mathbf{x}_n + \mathbf{B}_n \omega_n + \mathbf{v}_n$ Where $\mathbf{x}_{n+1}$ is a vector comprising fitness and fatigue on day $n+1$, $\mathbf{A}_n$ is a diagonal 2×2 "transition matrix" of coefficients that multiply the current fitness and fatigue values, $\mathbf{B}_n$ is a 2×1 matrix of coefficients that multiplies the scalar training input, and $\mathbf{v}_n$ is the state noise quantified by a 2×2 covariance matrix that is generally denoted by $\boldsymbol{Q}$. The state noise describes the random changes in state (e.g. fitness and fatigue) above and beyond the deterministic component involving training and past fitness or fatigue. To match the standard FFM, the matrices above are set as follows: $\mathbf{x}_n = \begin{pmatrix} g(n) \\ h(n) \end{pmatrix}, \quad \mathbf{A}_n = \mathbf{A} = \begin{pmatrix} e^{-\frac{1}{\tau_g}} & 0 \\ 0 & e^{-\frac{1}{\tau_h}} \end{pmatrix}, \ \mathrm{Var}(\mathbf{v}_n) = \boldsymbol{Q} = \begin{pmatrix} \sigma_g^2 & \sigma_{g,h} \\ \sigma_{g,h} & \sigma_h^2 \end{pmatrix}$ $\mathbf{B}_n = \mathbf{B} = \begin{pmatrix} e^{-\frac{1}{\tau_g}} \\ e^{-\frac{1}{\tau_h}} \end{pmatrix}$ Where the state of the system is not observed directly but is accessible by means of indirect measurements of performance p_n from the equation: $p_n = p^* + \mathbf{C}_n \mathbf{x}_n + \boldsymbol{\eta}_n$ Where $\mathbf{C}_n == \left(k_g, -k_h\right)$ a 1×2 matrix and $\boldsymbol{\eta}_n$ is the observed performance noise described by actual measurement errors with variance denoted by ξ^2. For the VDR model, the matrix **B** changes to: $\mathbf{B}_n = \begin{pmatrix} e^{-\frac{1}{\tau_g}} \\ k_{h_2}^i \cdot e^{-\frac{1}{\tau_h}} \end{pmatrix}$ Where $k_{h_2}(i)$ is as above in the formulae table. Such that the state-space model can be combined with a Kalman-filter to address concerns with model error and uncertainty in model predictions.

Chapter 3: Research design

This chapter explores the lens through which this project has been conceptualised and developed, discusses the key aims and objectives that were introduced in Chapter 1, and describes the research model developed and used to guide to the PhD process. It has been purposely placed after the literature review so that the reader has an awareness of the FFM literature. Section 3.1 begins by describing some of the early challenges of the project, and then explains how the original research component evolves from, and has been guided by, a novel research model developed to match the current state of the field. At the end of the chapter, in section 3.2, the aims, objectives, and resultant outcomes of the project are summarised.

3.1 Challenges, research model development, and key aspects of the project

Doctoral projects in the field of sport science are commonly carried out in mature research areas, often where the path to further development is considerably well mapped by established research frameworks with applied methodologies and a clear literature body (Bishop, 2008). In contrast, the small research space of fitness-fatigue modelling within the sport sciences has seen a lack of progress relative to its long historical timeframe. The area is also largely comprised of a collection of disorganised experimental work deprived of clear and consistent recommendations for the direction and requirements of future study. Despite the conceptual significance of the standard FFM (Chiu and Barnes, 2003; Bompa and Buzzichelli, 2018), there is surprisingly little notice paid toward mathematical FFMs by the broader sport science community. This is difficult to attribute to a lack of general interest in mathematical modelling within sport science (Jeffries *et al.*, 2020), for example the acute-chronic workload ratio and its various extensions are conceptually simple and have drawn significant attention (both positive and negative) across modern exercise research and practice (Windt and Gabbett, 2019; Impellizzeri *et al.*, 2020; Wang *et al.*, 2020). There also exists the rapidly developing and popular area of 'performance analysis' in research and practice, that places large emphasis on quantitative devices (McGarry, 2009; O'Donoghue, 2014). Most probable, a lack of research interest amongst sport scientists in mathematical FFMs is attributable to the nature of the underlying literature body, and in particular a combination of factors including: 1) the multidisciplinary skillset required to understand, implement, and further improve models and methods in existing research (and associated high bar to entry); 2) several instances of unclear communication of concepts, methods, limitations, and the processes involved in implementation within studies; 3) frequent publication of novel FFM research in computing, engineering, and statistical journals that are, unsurprisingly, out of the purview of most sport scientists; 4) limitations in the conceptualised structure (and behaviour) of existing models; and 5) very little consideration of key experimental factors that may affect model accuracy or implementation and a clear deficit in practical guidance. These issues

were first introduced in chapter 1 and have all been discussed in chapter 2, however it is the combination of these factors that informed the four primary aims of the thesis. These were to:

1. Systematise the FFM literature body, providing sufficient detail and structure to where there exists a consistent narrative threading the historical literature, pertinent concepts, and contemporary work to address limitations in basic FFM structure
2. Conduct original research studying key experimental factors (measurement error and testing-frequency), and methods (parameter estimation), that may affect model accuracy or utility
3. Identify and raise awareness of alternative FFMs beyond the basic model, and advanced methods, that reflect more promising avenues for future research
4. Develop flexible code tools that facilitate future study and address the current gap in availability of resources

This first aim was operationalised by two main objectives: 1) composition of a review (Stephens Hemingway *et al.*, 2021; Swinton *et al.*, 2021) that focussed on balancing clarity with mathematical rigor in the communication of concepts and methods; and 2) development of educational code resources to supplement the learning process and in the particular aspects of fitness-fatigue modelling that were examined by the reviews. The second of these objectives also links to aim four, where a proportion of the software resources presented as supplementary materials to these reviews have been extended or incorporated into the resources developed within this thesis to facilitate future study (Chapter 6). It was anticipated that these two objectives together would suffice to improve accessibility of the research area and broaden shared understanding of existing limitations. It was also hoped this may prevent further applied research proliferating a cycle of blindly duplicated methods that suffer the same underlying afflictions in research design (e.g., a lack of robust model evaluation, insufficient data reflected in the low frequency of measured physical performances, and/or the selection of noisy performance measures). Much of the narrative and theory presented in the reviews is absorbed into the work of chapter 2 in this thesis, and the educational code resources outlined form a share of chapter 6 (the rest of which are directly related to aim 4). Collectively, this thesis and in particular these two chapters (2 and 6) represent the most up to date and complete critical synthesis of FFM research and source of practical resources for researchers to learn about and apply FFMs.

The second aim, to carry out original research studying key experimental factors (measurement error and testing-frequency), and methods (parameter estimation), that may affect model accuracy or utility, was informed by the limitations of the research body identified in chapter 2 and logistical challenges associated with typical primary research. What became clear was that investigation into these factors was a prerequisite to any further experimental study of model validity from training-intervention designs. Particularly, as no attention had been paid previously as to how these factors interacted with

model accuracy, and methods in experimental FFM research appeared to be replicated from study to study (see tables in appendix B) without much consideration of how experimental design could bias (positively or negatively) the quantification of model validity. Historically, the effect of these factors had been absorbed within conclusions of model misspecification without accurate quantification of or adjustment for their influence. Toward this end, a novel *in silico* experimental approach was developed to facilitate study of these important aspects of FFM implementation. The term *in silico* (pseudo-Latin or New-Latin translation: 'in silicon') is broadly defined as an experiment performed in or on a computer (often by computer simulation of a model) and is an allusion to the common Latin terms *in vitro* and *in vivo* within scientific research (Ekins, Mestres and Testa, 2007). The term is also closely related and often used interchangeably with the term 'computer experiment', defined as an experiment to study a computer simulation. Stated in more formal terms, the first objective associated with aim 2 was to investigate the effects of two factors, measurement error (quality) and testing frequency (quantity) of typical modelling data, on standard FFM prediction accuracy. The second objective of aim 2 was to assess the suitability of a quasi-Newton optimisation algorithm for fitting an FFM, specifically studying the sensitivity of this algorithm to starting point selection and existence/implications of local extrema in the search space. The theory and specific detail of the experimental approaches developed to match each objective is covered in detail in the respective chapter (4, 5). However briefly, the central idea to the designs is the assumption the FFM is deterministic of training response and therefore allows the confounding effects of model misspecification to be detached from the study of model accuracy under simulated 'true' data. In turn, this enables lower-bound study of the isolated effects of experimental factors on modelling aspects including prediction accuracy, whereby in the real world the model would only be expected to perform worse due to model misspecification and other unknown factors not studied. If certain conditions on factors such as measurement error (quality) and testing frequency (quantity) can be identified that cause poor model accuracy, applied future research can avoid similar issues and be better informed regarding appropriate practice in FFM implementation to minimise negative bias of model validity.

The third aim of this project was to identify and raise awareness of alternative FFMs beyond the basic model, and advanced methods, that reflect more promising avenues for future work. For example, as discussed in chapter 2, the VDR model (Busso, 2003) addresses the limitation of a lack of interaction between training sessions in the standard model, the fitness-delay and exponential-growth models (Calvert *et al.*, 1976; Philippe *et al.*, 2018) modify the structure of the fitness response to better match the conceptual notion of parabolic growth rather than an instantaneous impulse, the external threshold saturation function (Hellard *et al.*, 2005) and non-linear FFM system (Turner *et al.*, 2017) address nonlinearity in the response to training (in different ways), and the state-space reformulation with Kalman filter addresses model uncertainty (Kolossa *et al.*, 2017). In addition, substantive critical analysis was added to discussion of experimental methods such as cross-validation that are arguably

crucial for future experimental research to apply when studying FFMs. Collectively, and particularly if combined in some respect, these model extensions reflect good avenues for future research, and are promising attempts at addressing the historical conceptual limitations of FFMs commonly cited by sport scientists (Taha and Thomas, 2003; Hellard *et al.*, 2006). Chapter 2 and the associated reviews go some way to toward improving the profile and awareness of these extensions and the vital methods of model evaluation, however, to maximise impact in this respect it was felt that practitioners would also benefit from practical tools that can assist with understanding and operationalising these models for further research purposes. Toward this end, the single objective guiding realisation of aim 4 was to develop practical and flexible code resources in the programming language R to facilitate future FFM study in sport science under the models and requisite models identified by aim 3. Chapter 6 takes the reader through the development of code in an educational style, beginning from fitting the standard model through to the VDR model, nonlinear FFM system, and state-space reformulation with Kalman filter. Code is presented inline within the text, and synthetic data developed via simulation to demonstrate the concepts and keep the work self-contained. Factors such as cross-validation and parameter estimation (optimisation) are examined at a practical level in Chapter 6, with theory accompanying implementation in R. Shortly, a research model that was developed to guide this project is explained, demonstrating how the chapters in this thesis that support the primary aims already described fit within an overall research methodology. First, additional challenges of the project are explored that were important influences in shaping the PhD, and in particular justification of the novel *in silico* experimental design facilitating the original research. The aims described above are also summarised in table form in Table 3.2 at the end of this chapter.

At the start of the project, this research area presented a steep learning curve, particularly as a new graduate researcher with formal training predominantly in the areas of sport, health science and exercise prescription. Due to the combination of the unorganised literature body, a lack of existing research frameworks to accommodate modelling in sport, and the necessity to cultivate (rather hastily) a broad and serviceable interdisciplinary skillset to begin to understand and address these issues. From the outset, this steered the project toward a structure that is somewhat atypical of a doctoral research project in sport and exercise science, with an almost equal focus given to the areas of: 1) analysis of the literature and scientific communication; 2) development of novel cost-effective approaches to FFM experimentation for studying key experimental factors; and 3) examination of prospective methods for future study under real-world data. It would have been straightforward in some respects to adopt a standard data collection approach for this PhD, comprising of primary research studies that adopt similar experimental practices as previous FFM study and that comprised roughly of the following process:

1. Design a structured training intervention and recruit participants (typically an accessible population, i.e., non-elite undergraduate sport science students)
2. Conduct semi-controlled, short duration training study that is not realistic of real world practice
3. Fit a basic FFM via a generic method previously used (e.g., one-shot optimisation of model parameters by a local optimisation algorithm that minimises NLS)
4. Test the fitted model against in-sample data, reporting model fit

Whilst there is clear value in such an approach and in some ways, there is no getting around their eventual use to better understand model validity, there are several limitations. The approaches are expensive, and most importantly would likely have limited the extent of any insights and understanding that could be gained from this project with the work not particularly well suited to the existing literature body. For example, at the start of this project there was a clear deficit in the current recommendations (or lack thereof) for robust experimental assessment of model performance, and no work separating appropriate models that address existing conceptual limitations from those that don't. In other words, beginning with the approach outlined above would have possibly resulted in the big picture of FFMs being missed (e.g., limited knowledge of FFM limitations, pitfalls with experimental methods, common flaws in experimental design that can influence modelling results, and little work to identify suitable practices), with no guarantees any of these issues would have been revealed by the end of the PhD. Therefore, in contrast to the applied primary research approach outlined, a significant portion of allotted time within this project was dedicated to deconstructing the implications of several technical concepts, thoroughly scoping and reconnecting the literature to identify significant gaps in understanding and to develop an awareness of the limitations that had persistently troubled the field over its history. In this way, original insight was created from existing knowledge, particularly in the areas of parameter estimation, model evaluation, and argument for study of these types of models. Secondary to this but perhaps even more important was communication of these aspects identified in the FFM literature to the wider sport science community, in as clear a manner as possible, to deliver a consistent narrative that threads the whole of the historical research up to the present state. The result of this process was a clearer understanding of the type of experimental research required to progress the research area. The downside of this process was that it drew significant time from allotted window to complete the PhD. In addition, a more cost-effective approach had to be developed to study these models in the absence of any research funding, leading to the development of the novel *in silico* designs described. Although cost effective, it was primarily identified that these approaches offered an extensive approach to studying the effects of these types of factors (e.g., error frequency) and methods of parameter estimation.

With the final year of the project taking place in an international pandemic, this removed the chance of collecting primary research data within the remaining timescale, and therefore represents a

limitation of the thesis toward answering the central research question of FFM validity, but also leaves the door open for continuing the work. However, this led to the alternative decision to develop extensive software resources that provide the capability to help researchers conduct future experimental studies, likely with far bigger sample sizes than this project could have achieved within the constraints of its already limited resources. The decision to develop code resources in lieu of further experimental study was also informed by the source "Opening Science: The Evolving Guide on How the Internet is Changing Research, Collaboration and Scholarly Publishing" (Bartling and Friesike, 2014), and in the particular chapter "Open Science: One Term, Five Schools of Thought" (Fecher and Friesike, 2014). The chapter defines and discusses the "infrastructure school of thought". This school of thought in Open Science assumes that the efficiency of research is dependent upon the availability of tools and applications available to researchers (Fecher and Friesike, 2014). The goal of the infrastructure school of thought is to "*create openly available platforms, tools, and services for scientists*" (Fecher and Friesike, 2014). Arguably, the tools developed in chapter 6 offer more impact to the FFM research domain and contribution to knowledge than a single small scale primary research study could have provided given the existing literature body and logistical constraints of the project. In addition, if researchers now seek to develop collaborations with industry practitioners to obtain data that could be used to retrospectively backtest FFMs, then stakeholder interest and understanding in the research area is critical. The material in chapter 2 and associated reviews (Stephens Hemingway *et al.*, 2021; Swinton *et al.*, 2021) offer a clear source of background information and discussion that can be offered to potential stakeholders (e.g., practitioners, clubs, teams). The code tools developed provide well-defined methods for fitting and evaluating various FFMs with data and could be used (potentially in the form of real time demonstrations) to improve stakeholder knowledge and buy-in to this type of scientific study, and to expedite the research process. Both are important factors (buy-in, expeditious timeframes) when it comes to research collaborations at the elite population level.

As discussed, with no existing research framework or model available to guide this type of research project within sport science, and following the challenges already described at an early stage in this thesis, a novel research model was developed to guide the project. This research model may hopefully be useful to other researchers and doctoral students working in the area of performance modelling or FFMs, particularly for envisioning the steps involved in the progression from research to integration of performance models into practice. The framework is presented in Table 3.1.

Table 3.1: Research model for studying performance modelling in the sport and exercise sciences

	Stage		Description	Location in thesis
Description and evaluation of the problem ↓	**1**	Scoping	• Definition of initial modelling problem to be solved • Identification of barriers to study • Development of background and context	Chapter 1
	2	1. Research synthesis 2. Methodology development 3. Researcher preparation	• Refinement of central question or hypothesis • Identification of gaps in existing knowledge • Identification of gaps in communication of prior research • Identification of limitations and quality of prior research • Identification of barriers, challenges, and impact of further study including cost-benefit analysis for prospective work • Identification of issues surrounding scientific communication in the area in general, and broader awareness of the subject area in the discipline • Development of the required skillset (as appropriate) to facilitate further required study	Chapters 2 and 3
Experimentation ↓	**3**	Theoretical study and/or experimental assessment (original research)	A progressive feedback-loop comprising problem solving research, and involving: 1. Experimental testing and/or theoretical study of model validity and factors associated with experimental design 2. Reorganisation of existing knowledge to incorporate new findings, update current state-of-art, determination of direction for future work 3. Summary judgement of model validity or suitability of a method as a viable option for use in practice. Only if appropriate based on meeting a threshold of available research evidence, else back to step (1).	Chapters 4, 5, 6 (Step 1) Chapter 6, 7 (Step 2)
Implementation ↓	**4**	Identification stage	Identification of and work to address barriers to uptake in practice	-
	5	Development and integration into practice	Development or refinement of tools and resources to integrate existing validated models into practical problem-solving tools for the field (e.g., tools to assist in training planning and/or decision making)	-
	6	Implementation study	Study of the implementation process and outcomes from the use or uptake of tools developed at stage 5 in practice.	-

Finally, methods of scientific communication used within this project are discussed, as they relate to the research framework in Table 3.1. At stages 2 (research synthesis) and 3 (experimentation) consideration was given to the typical approach of disseminating scientific findings and knowledge within sport science, and how this communication model might be improved upon in relation to the work within this thesis. Historically, the prevailing reporting method for scientific findings has been formal peer-review and entry into scientific journals. Although peer-review is an important aspect of science, in its current form the process is slow, unpredictable, and not well suited to sharing information with other researchers in order to facilitate rapid development in research areas that benefit from momentum (Binswanger, 2014). In contrast, fields such as computer science place importance on early dissemination, with preprint (pre-peer-review) publication of findings largely the norm, usually via conference proceedings and other online platforms (e.g., arχiv, ResearchGate). Computing research is also typically accompanied by code, data, and transparent or reproducible analysis (Peng, 2011), naturally increasing the credibility of the work. Open-research and open-science perspectives can nurture a community driven feedback loop, foster fresh interest amongst researchers and practitioners, and ideally seed rapid iteration and refinement of methods in expeditious timeframes compared to the expected timeframe of the peer review model on its own (Peng, 2011; Fecher and Friesike, 2014). Uptake of preprint dissemination has been slowly increasing in sport science over the last few years, particularly with the introduction of platforms such as sportR$x$iv. However, it is still far away from the norm, conceivably due to pressures to publish in academia and uncertainty regarding publication success following preprint submissions. Nevertheless, there exists a growing call for change in the current peer-review model (Binswanger, 2014; Fecher and Friesike, 2014), and the nature of the aims and objectives of this project lend themselves well to an early communication approach. Toward this end, the reviews (Stephens Hemingway *et al.*, 2021; Swinton *et al.*, 2021) were published in preprint, along with the study in Chapter 5 (Stephens Hemingway, Swinton and Ogorek, 2021). This process generated significant peer feedback, led to multiple iterations of improvement, and collated invaluable input from sport scientists including those who had contributed to the historical literature and those with a curious interest in the subject. The results of the preprint process was several informal peer review cycles, exposing the work to multiple different perspectives in its early stage and ultimately improving the final version submitted for peer review. This process is restricted in scope within the standard model of scientific communication, and it is hoped that communicating as much work as possible within this thesis will play a small role in encouraging others to move toward models of early dissemination to supplement the peer-review process. This may also be important for stimulating the field of performance modelling at an acceptable rate relative to the lack of progress over the historical timeframe.

3.2 Aims and objectives (summary)

Table 3.2: Aims and objectives of the thesis

Aim		Objective	Thesis location
1	Systematise the FFM literature body, providing sufficient detail and structure to where there exists a consistent narrative threading the historical literature, pertinent concepts, and contemporary work to address limitations in basic FFM structure	(1) Composition of a review series that focusses on balancing clarity with mathematical rigor in the communication of concepts and methods (2) Development of educational code resources to supplement the learning process in the particular aspects of fitness-fatigue modelling examined by the review	Chapter 2 (Objective 1) Chapter 6 (Objective 2)
2	Conduct original study of key experimental factors (measurement error and testing-frequency), and methods of model estimation that may affect model accuracy or utility	(1) Investigate the effects of two factors, measurement error (quality) and testing frequency (quantity) of typical modelling data, on standard FFM prediction accuracy (2) Assess the suitability of a quasi-Newton optimisation algorithm for fitting an FFM, specifically studying the sensitivity of this algorithm to starting point selection and existence/implications of local extrema in the search space	Chapter 4 (Objective 1) Chapter 5 (Objective 2)
3	Identify and raise awareness of alternative FFMs beyond the basic model and advanced methods, that reflect more promising avenues for future research	(1) Identify and prioritise remaining research problems in the area of fitness-fatigue modelling (2) Illuminate the theory of existing models and methods representing reasonable or requisite approaches to addressing remaining research problems	Chapters 2,4,5,7 (Objective 1) Chapters 2,4,5,6 (Objective 2)
4	Develop flexible code tools that facilitate future study and address the current gap in availability of resources	Develop practical and flexible code resources in the programming language R to facilitate future FFM study in sport science under the models and requisite models identified by aim 3	Chapter 6

Chapter 4: The effects of measurement error and testing frequency on the standard fitness-fatigue model applied to synthetic resistance training data: An *in silico* experimental design

4.1 Preface

The notion that a resistance training stimulus has influential effects on the fitness and fatigue states of an athlete underpins the practice of strength and conditioning professionals (Chiu and Barnes, 2003; Bompa and Buzzichelli, 2018). Primary training decisions made by coaches around simple sounding questions of "how much? and "how often?" are influenced by perception of an athlete's individual tolerance to training (in an acute and cumulative sense) before recovery thresholds are surpassed. If the physical and mental demands of training repeatedly and unremittingly overwhelm an athlete's tolerance, such that they cannot recover adequately between training sessions, their ability to adapt, readiness to perform and effort in training can become suppressed below a normal range. This phenomenon is referred to in the literature as over-training (Kellmann *et al.*, 2018). Not to be confused with over-reaching, where periods in which demands higher than an athlete's tolerance are purposefully planned by a coach to stimulate adaptation, and which are interspersed by periods of low demands. If over-training occurs, it can take a lengthy and often unpredictable period of time for the athlete to return to a normal state of performance and overall well-being. At a more detailed level, consideration of short- and long-term competitive schedule, training emphasis, training history, age, gender, location, sporting demands and many other factors are integrated into training decisions. Nevertheless, it is the fundamental desire of a coach to maximise the rate and magnitude of physical adaptation (fitness) under known environmental, individual, and logistical constraints; whilst simultaneously managing and ideally minimising fatigue (Bompa and Buzzichelli, 2018). Thus, the fitness-fatigue model is ingrained within the workings of most sport scientists and strength and conditioning coaches as a conceptual framework of response, whereby performance is thought of in terms of the antagonistic effects of training (fitness and fatigue) with associated decay over time. It is this conceptual framework that guides thought processes toward preventing an athlete reaching a state of being over or under-trained. To move beyond a conceptual framework, it can be proposed that it may be appropriate to have coaches apply existing quantitative fitness-fatigue models to assist with training prescription (Busso and Thomas, 2006). Additionally, a shift in thinking should occur where criterion performances used to fit the model are no longer capturing competitive outcome but are instead measures that comprise more direct relationships with physical training. Toward this end, initial preparatory work must be conducted on these models to study their behaviour, conceptualisation, and factors that may negatively influence their performance in practice.

A novel approach for isolating operational factors that may negatively influence prediction accuracy within real-world research, such as the quality and quantity of data required to fit an FFM, begins by employing a deterministic assumption that states an FFM completely specifies an athlete's response to training. In doing so, the contribution of model misspecification to prediction error is detached from the problem, and these factors can then be studied directly within an extensive computer experiment design (*in silico*). The process involves first simulating the FFM under a series of synthetic inputs (training loads) and carefully selected parameter values. The output (model simulated performance) over the length of the training loads represents the *true performances* for a 'hypothetical athlete'. The manually selected parameter values are also regarded as the hypothetical athlete's *true parameters* (i.e., the parameters that characterise their response, as the model is assumed to be completely deterministic). The set of model-generated true performances are then duplicated a set number of times, and these duplicates transformed to reflect different conditions of two factors (measurement error and testing frequency). If for example there were three measurement error conditions ($\epsilon_1, \epsilon_2, \epsilon_3$) and three testing frequency conditions (ν_1, ν_2, ν_3), all combination of the conditions of these two factors (ϵ, ν) would yield 15 'scenarios' in total. Each scenario is then assigned 10^4 of the duplicated sets of true data, with each set transformed to match the scenario by adding random noise sampled according to a distribution $N(0, \epsilon^2)$, and removal of datapoints according to ν_i (for example, so that there was now a data point only for every other time-step). Note that within each scenario, transformed sets will differ based on random variation in the sampled noise (uniqueness), and scenarios will differ based on average magnitude of noise added and frequency of data comprising the sets. Each set of transformed true performance data in each scenario (i.e., true + noise, subsetted according to ν) is then fitted back to an FFM via an iterative optimisation algorithm. The parameter estimates obtained from this process are used to re-simulate the model and derive performance predictions that can be compared against the true values to generate estimates of model prediction error. Within each scenario, the average model error is reported over the 10^4 sets, such that differences between the conditions of each factor can then be compared. Note that the inputs (training loads) stay consistent across the whole experiment. Following this approach, lower bounds can be identified on the negative effects of certain conditions of these factors on model utility, such that in the real-world additional model misspecification and other unknown factors will likely only result in worse model performance (i.e., higher prediction errors). This type of computer experiment design is novel to sport science and is powerful because it provides a cost and time-efficient approach to studying factors influencing model utility under a range of distributional and operational conditions, that would otherwise be challenging to study within standard *n*-of-1 experimental designs with real data. This approach may also offer a vehicle for researchers and practitioners to study other phenomena, generate practical insights, and assist with identifying factors which should be considered in future empirical research.

This chapter follows the implementation of the novel approach described to ascertain minimum acceptable measurement error and criterion performance frequency conditions that should be met for the use of an FFM to be considered a worthwhile vehicle for further scientific exploration of performance modelling [4]. It is crucial that future research does not negatively prejudice results of any model validity work (e.g., estimation of model misspecification) by introducing excessive prediction errors due to poor operational practices (e.g., poor data quality or low quantity), possibly resulting in an overestimation of model misspecification and underestimation of model utility. Toward this end, this study also presents an opportunity to provide clearer recommendations for researchers with regard to factors such as measurement quality and quantity for use in fitness-fatigue modelling.

4.2 Introduction

Each FFM can be described by a mathematical equation that is tailored to an individual athlete, and which links the magnitude and decay rate of the positive and negative after-effects experienced by the athlete (Banister *et al.*, 1975; Chiu and Barnes, 2003; Taha and Thomas, 2003; Jobson *et al.*, 2009). As has been described in detail in the literature review, this is achieved in practice by conducting a period of training comprising frequent performance measurement, with model parameters retrospectively fit to best match the input and output data generated. This fitting process itself is also commonly referred to as *model training* and once complete the model and fitted parameters can theoretically be used to predict future response to physical training and inform program design (Banister *et al.*, 1975; Calvert *et al.*, 1976; Busso *et al.*, 1990; Taha and Thomas, 2003; Schaefer, Asteroth and Ludwig, 2015; Stephens Hemingway *et al.*, 2019). Accurate quantification of model input (training load) and regular best-effort criterion trials (e.g. timed run, or maximal load lifted) revealing the athlete's current capabilities are therefore required (Morton, Fitz-clarke and Banister, 1990). Across prior experimental research, FFMs have been traditionally applied to endurance athletes (e.g., runners, swimmers, cyclists and triathletes) as training loads appear simpler to calculate in these closed-skill sports and criterion trials closely match sporting performance. Additionally, a small number of studies have investigated fit of FFMs with performance in individual and team sports where strength and power are the primary fitness components (Busso *et al.*, 1990, 1992; Busso, Candau and Lacour, 1994; Sanchez *et al.*, 2013; Agostinho *et al.*, 2015). This matches the conceptual framework adopted by many strength and conditioning coaches, where the physical capability of an athlete is

[4] **A version of this chapter was published in the International Journal of Sport Science and Coaching:**

Stephens Hemingway, B., Burgess, K., Elyan, E., & Swinton, P. (2019). The effects of measurement error and testing frequency in applying the Fitness Fatigue Model to resistance training: A simulation study. *International Journal of Sports Science and Coaching*, 0(0), 1–12. doi.org/10.13140/RG.2.2.19730.56005

Associated files (code): github.com/bsh2/thesis/c4

assessed via standardised movements (e.g. 1RM squat or 40m dash) that comprise relevant dimensions of fitness (e.g. strength, power) (Carlock *et al.*, 2004; Al-Otaibi, 2017; Revie *et al.*, 2017). In elite sport, even the smallest change in sporting performance may have a significant influence on the outcome of a competitive event (Jobson *et al.*, 2009). Likewise, small-moderate variations in dimensions of an athlete's physical capability during a competitive period may significantly alter individual performance in said sporting event. It follows, and has been previously stated, that in order to be useful in practice as a tool to derive training programs that match the desired time-course of performance it is important that fitness-fatigue models are able to predict observed performance change with only a minor error margin (Jobson *et al.*, 2009).

Model predictions from estimated parameters are thought to be highly dependent on the density of measured performance data over the modelled period (Stevens, 1986; Bates and Watts, 1988; Sen and Srivastava, 1990; Davidian and Giltinan, 2003; Hellard *et al.*, 2006; Pfeiffer, 2008; Jobson *et al.*, 2009). However, high frequency performance measurement is challenging to achieve for most researchers and practitioners (Jobson *et al.*, 2009). There is also little contextual research available to guide selection of a measurement frequency that is likely to reduce existing concerns around fitting and testing of FFMs with insufficient data. Several challenges exist in researching the effectiveness of fitness-fatigue models to predict training response, and to identify the importance of factors such as measurement error and testing frequency. Within standard research designs, a primary challenge is recruitment of large enough sample sizes to conduct robust long-duration studies comprising a substantial training intervention and high frequency performance measurement to accurately isolate the effects of measurement error and testing frequency on prediction accuracy. Conducting an experimental study that meets these requirements is likely to necessitate, at the very minimum, substantial resources in terms of time, money, expertise, equipment. In addition, the existence of error in all measurements precludes true underlying performance of an athlete to be known (Swinton *et al.*, 2018), placing limits on the ability to assess predictions. Across the prior literature there are several limitations that frequently arise, and which may be attributable to challenges discussed that are common amongst standard research designs. For example, the majority of fitness-fatigue model experimental studies have comprised an insufficient number of observed performance measurements to appropriately fit the model, were it even a simple linear polynomial (Hellard *et al.*, 2006; Pfeiffer, 2008). The non-linearity of the model, and degrees of freedom from the large number of parameters have been suggested to add substantial requirements on the number of data points per model parameter (Bates and Watts, 1988; Sen and Srivastava, 1990; Davidian and Giltinan, 2003; Hellard *et al.*, 2006). Additionally, with the exception of a handful of more recent studies (Kolossa *et al.*, 2017; Williams *et al.*, 2018; Mitchell *et al.*, 2020), prior research has only assessed ability to retrospectively fit input and output data as part of the model training phase, with no treatment given to model predictive validity (either with regards interpolation, extrapolation, or both on out-of-sample data). This has represented

a major limitation of the research base to date, as the central premise of the mathematical fitness-fatigue model is to predict future response to training (Banister *et al.*, 1975; Taha and Thomas, 2003). For most potential research cases, particularly those without substantial funding, a standard experimental design appears an ardent first approach to studying either existing or new FFMs. Furthermore, without some understanding of the effects of factors such as precision and frequency of measurement on model validity in the basic case, researchers may unknowingly confound their studies. It follows that all future experimental fitness-fatigue model research must be tailored with some basic consideration given to factors such as measurement error and testing frequency, to reduce the likelihood of adding more confounding research to the literature body. However, there is clear need in the literature for contextual work available to guide these operational decisions.

One novel alternative to the standard research design is a computer experiment (*in silico*) design that establishes synthetic data to facilitate study. These types of experimental designs are highly novel in the sport and exercises sciences but afford a relatively inexpensive approach to studying lower-bound validity of the model under *small-world* conditions. Loosely, small-world conditions can be thought of as the best-case scenario for the phenomena studied. Therefore, the training response is assumed to be completely specified by a fitness-fatigue model, and observed performances are believed to deviate only due to defined factors such as the magnitude of measurement error. Adoption of this approach and associated assumption establishes a lower-bound, whereby similar practices in real-world settings can on average only lead to greater predictive errors, due to the influence of factors not considered (e.g., model misspecification). Figure 4.1 attempts to provides an intuitive graphical representation of this concept. Via computer experiment, thousands of fitted training responses can be systematically studied under pre-defined distributions of measurement error, and different levels of testing frequency (i.e., quantity of performance measurements available to fit the model). In this way, operational conditions that are likely to yield unreasonable results in practice can be identified and hopefully avoided prior to further investment in real-world laboratory or field-based study. Therefore, researchers can use extensive computational processes to eliminate prospective applications that are likely to lead to poor modelling outcomes. This type of experimental design may also provide insight into the limitations of methods, results, and conclusions within and between historical fitness-fatigue model research. Unequivocally, an in-silico approach under deterministic assumptions (with associated synthetic data) is not a replacement for experimental studies modelling data from real world human intervention trials. Rather, it can be thought of as a shrewd first step to establish whether there exists further justification for investment in more costly experimental approaches based on the practicalities of operational requirements. Occasionally, and in particular in this instance, it is argued that it may be the only cost-effective method for obtaining estimates of the influence of certain factors such as measurement error and testing frequency in isolation.

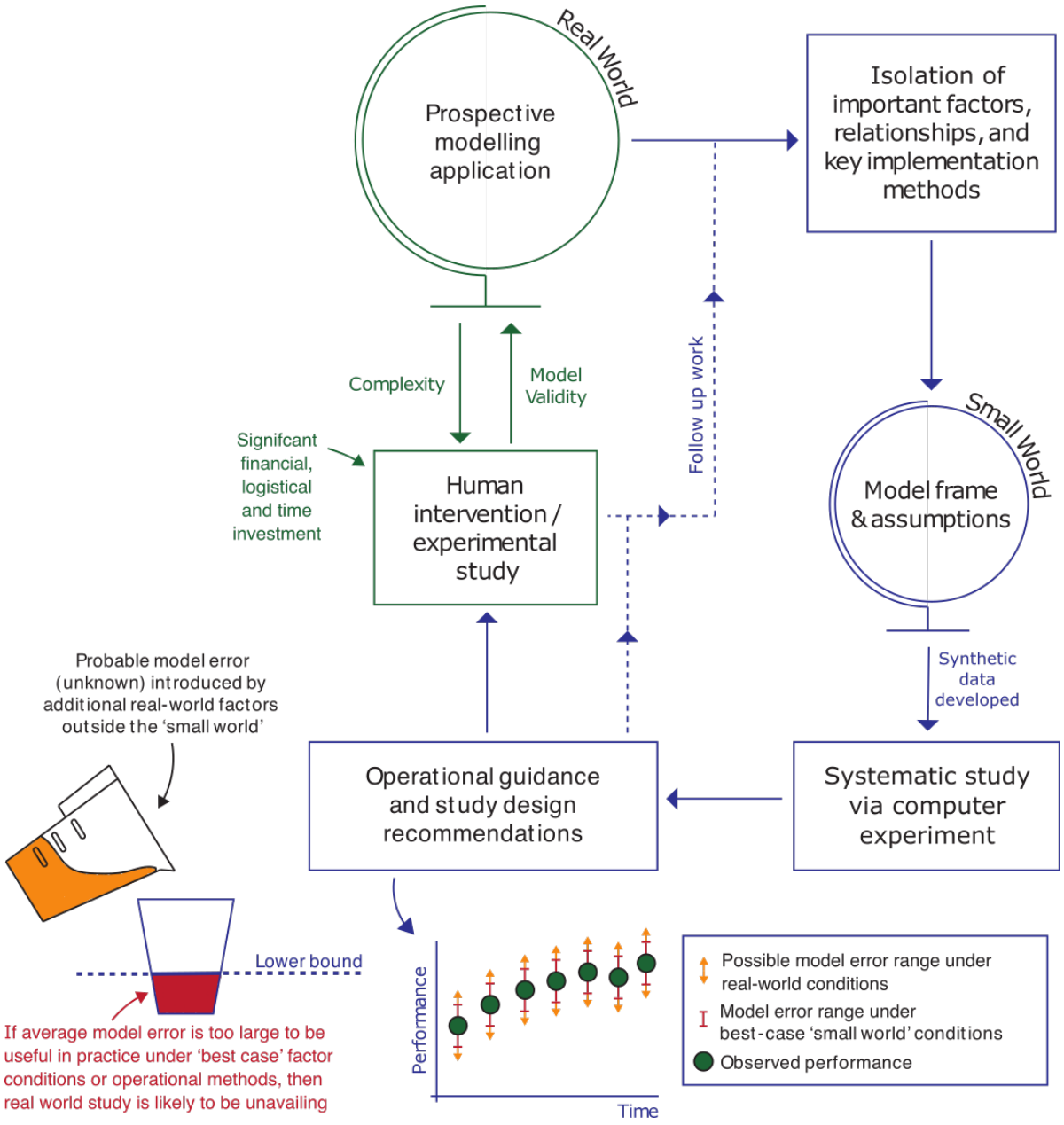


Figure 4.1: An illustration of the general approach to applying an in-silico approach to study FFMs

Finally, most high frequency performance measurements are infeasible for practitioners to achieve. Therefore, prospective FFM applications in the field of strength and conditioning must look to apply cheap-to-operate performance measures that can be tested daily, have high test-retest reliability, and do not cause acute or chronic declines in performance. Compiling lists of outcomes that meet these requirements is still needed, but as a starting point the use of activities such as the vertical jump may provide one suitable option (Watkins *et al.*, 2017). The vertical jump is a popular means of assessing an athlete's physical capability, and can be performed repeatedly within a testing session, and at a high frequency between sessions, without causing decline in performance (Watkins *et al.*, 2017).

Additionally, a range of mechanical variables (e.g. impulse, power, rate of force development) can be extracted during vertical jumps to assess various features of the neuromuscular system (Cormack *et al.*, 2008; Revie *et al.*, 2017). The experimental work presented in this chapter adopts an *in silico* experimental approach that develops a computer experiment to assess the effects of testing frequency and the magnitude of measurement error on the model fit and prediction accuracy, under an extensive range of possible conditions for each factor. The experiment applies a hypothetical resistance training intervention to develop synthetic (true) data, and then considers modelled change in the vertical jump via the standard 5-parameter FFM in the presence of additional noise and varied testing frequencies. Model fitting within the experiment is completed using a common Hill-climbing algorithm under a nonlinear least-squares (NLS) approach.

4.3 Materials and Methods

4.3.1 Experimental approach to the problem

As described in the introduction, an *in silico* approach was adopted to quantify the effects of measurement error and testing frequency from fitted FFM parameter estimates and associated predictions of performance for two hypothetical athletes (referred to as *intermediate* and *advanced*). The vertical jump was selected as the theoretical performance tool due to its popularity in athlete monitoring and potential to be used daily (Watkins *et al.*, 2017). A range of mechanical variables including power, impulse and jump height were considered as the model target for the study. However, each of the variables demonstrated similar relative profiles with regard to change in magnitude across an intervention compared to measurement error, and therefore each outcome would result in the same conclusions produced by this study. Power (Watts) produced during the vertical jump was ultimately selected for the model simulations, as it is a commonly understood metric across practice, and as it has previously been used in mathematical models to predict player fitness in response to training dose in Rugby union athletes (Revie *et al.*, 2017). To construct the synthetic data, it was first assumed that the fitness-fatigue model completely specifies a hypothetical athlete's response to training. Two popular training load distributions (*summated microcycles,* and *wave-like*) were combined with athlete-specific parameters in a model simulation process to generate realistic daily power values over a 16-week period. The resultant data reflects the true data, with regard to both the parameters (that 'define' the hypothetical athlete in terms of response) and the simulated performances. The construct of *true* values are only accessible within in-silico approaches where the assumption of complete model specification has been made. Generated values were split in half to create an initial *model training set* (weeks 1-8) and a *testing set* (weeks 9-16), under a simple 'hold-out' cross validation approach, to assess prediction error. The effects of measurement error and testing frequency were assessed by adding realistic errors to true values in the training set whilst fitting the fitness-fatigue model to varying proportions of this

augmented data representing 'observed' values (i.e., $observed = true + error$). The process replicated the situation adopted in real-world settings where observed scores on a physical test comprise the athlete's true score and measurement error (Swinton *et al.*, 2018). At each iteration, parameter estimates obtained from fitting the model training set were then combined with training loads corresponding to time-course of the testing set to obtain predicted power values and their associated prediction errors. Finally, extensive iterations were completed for each scenario to obtain distributional estimates of prediction error. A detailed flowchart illustrating the simulation process is presented in Figure 4.2.

4.3.2 Development of hypothetical athletes

The simulated data representing *true* performance change (i.e., without additional noise) for two hypothetical athletes (intermediate and advanced) were developed based on the inverse relationship between experience and improvement (Appleby, Newton and Cormie, 2012). Research investigating change in vertical jump power from a single intervention has demonstrated that improvements in peak power (W) for moderately trained athletes generally range between 0 and 20% (McBride *et al.*, 2002), whereas improvements for advanced athletes generally range between 0 and 5% (Harris *et al.*, 2000; Mangine *et al.*, 2008). Based on these findings, increases of 15% and 5% were selected for the intermediate and advanced athletes over the 16-week period, respectively. The same research base (Harris *et al.*, 2000; McBride *et al.*, 2002; Mangine *et al.*, 2008) was also used to identify realistic baseline values for performance p^* which was fixed within the study as a known parameter to simplify the problem of estimating prior performance and reduce the dimensionality of the parameter space.

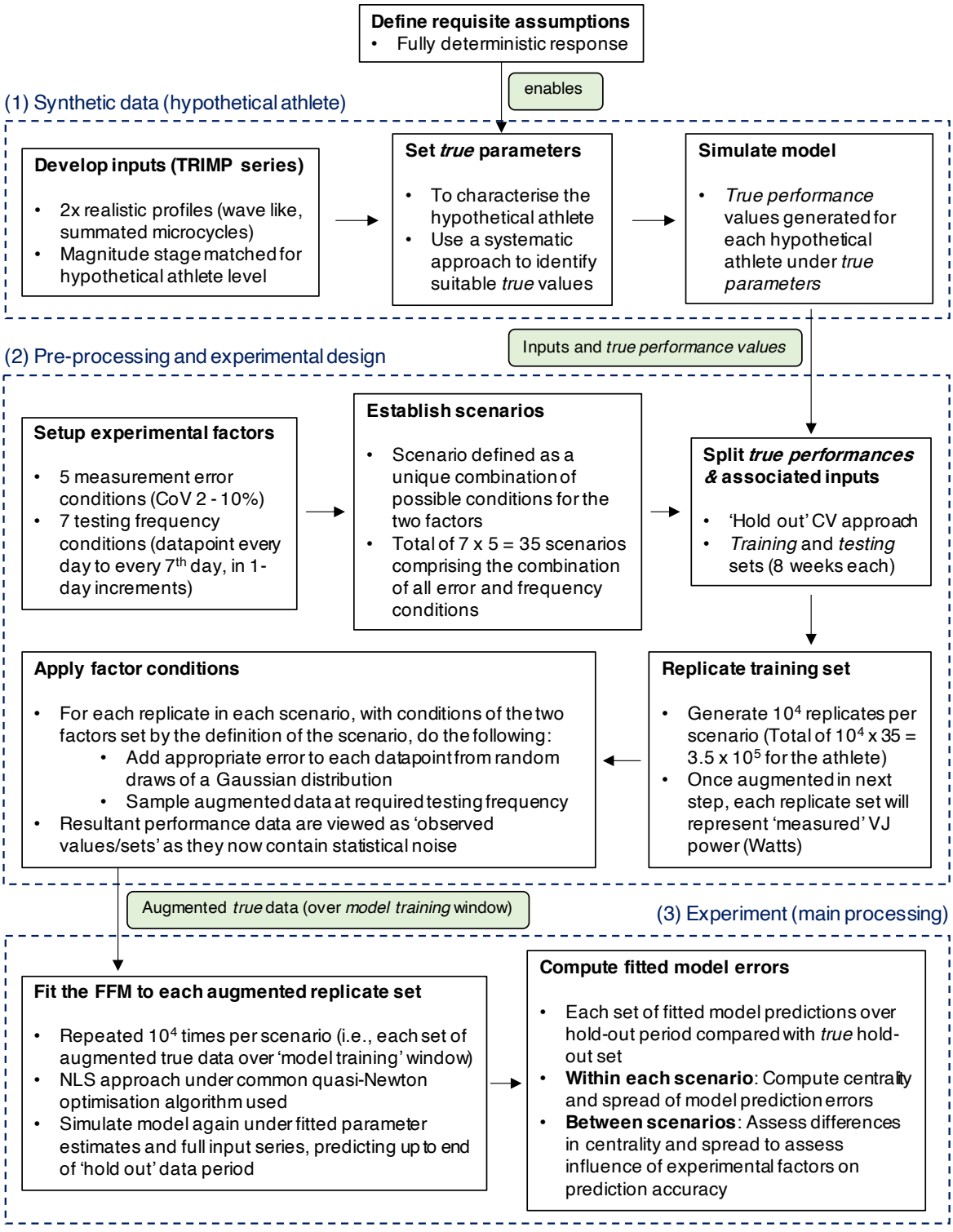


Figure 4.2: Flowchart describing the *in silico* experimental approach developed (repeated for each hypothetical athlete)

4.3.3 Development of training loads

A practitioner orientated approach was adopted to construct model input values that provide two realistic training load (TRIMP) series for the two hypothetical athletes across the 16-weeks. The first (TRIMP-1) followed a summated micro-cycles distribution, in which each four-week mesocycle comprised three weeks of progressive loading followed by one week of de-loading (Plisk and Stone, 2003). The second (TRIMP-2) followed a wave-like pattern where training load gradually increased and oscillated over each four week mesocycle (Baker, 1998, 2007). TRIMP values and their scaling (stage-matching) across the two hypothetical athletes are presented in Figure 4.3.

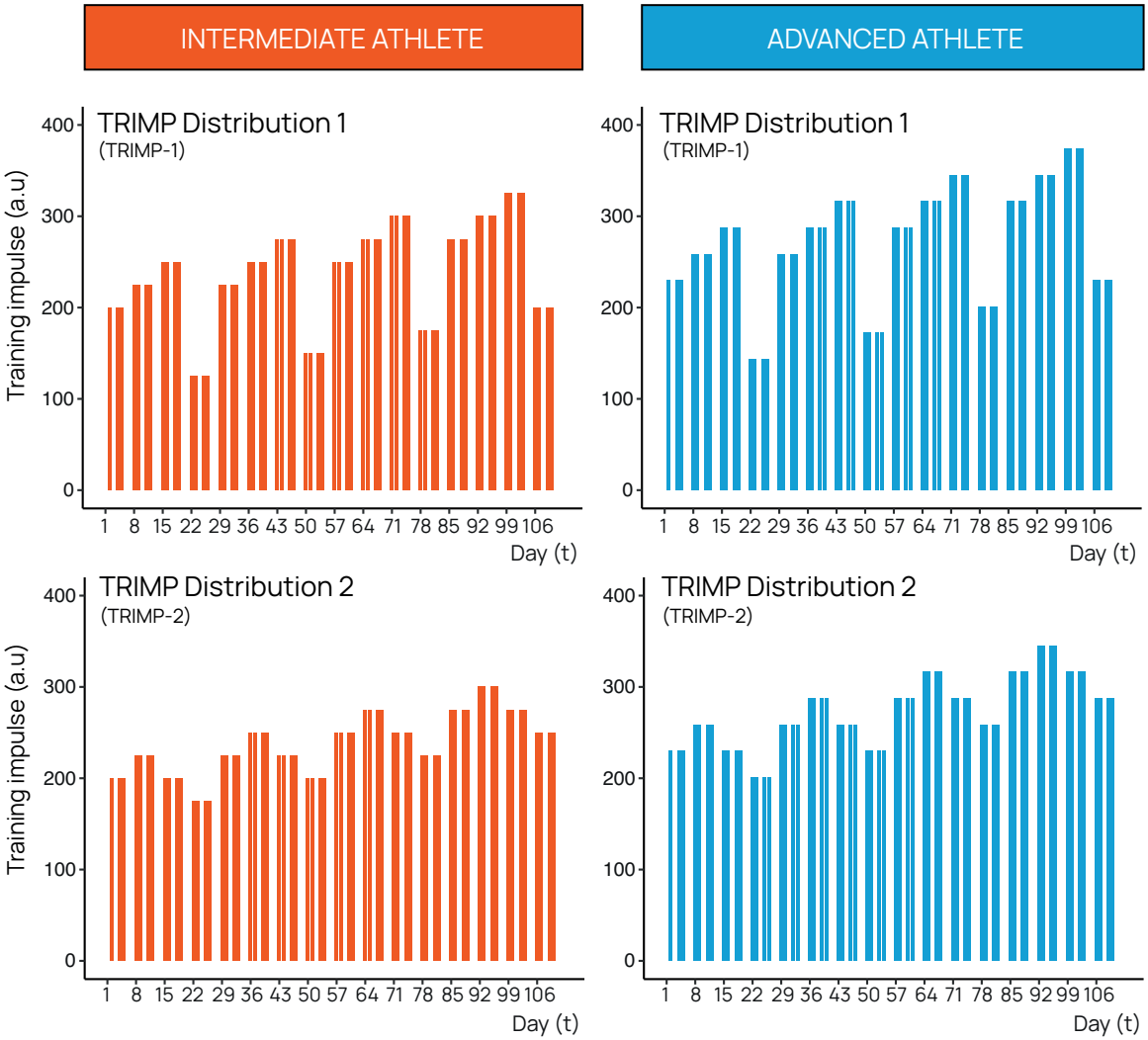


Figure 4.3: Distributions of scaled TRIMP values for the two hypothetical athletes (intermediate and advanced) over 16 weeks (measured in arbitrary units; a.u)

4.3.4 Development of athlete specific (true) parameters

The standard two-component five parameter fitness-fatigue model (eq. 4.1) was used to fit all models in the simulation study. With parameter p^* fixed a priori.

$$\hat{p}(t) = p^* + k_g \sum_{i=0}^{t-1} e^{\frac{-(t-i)}{\tau_g}} \omega_i - k_h \sum_{i=0}^{t-1} e^{\frac{-(t-i)}{\tau_h}} \omega_i, \qquad \omega_0 = 0 \tag{4.1}$$

Where $\hat{p}(t)$ represents model performance on day t and parameters k_g, k_h are weighting factors that translate the units of training load to the fitness and fatigue effects of the performance measure (power measured in Watts), respectively; τ_g, τ_h are the decay constants controlling exponential decay profile of fitness and fatigue effects, respectively; and ω_t is the daily TRIMP value. Athlete-specific *true* parameters were obtained through a process of systematic parameter-space exploration. Briefly, desired end-performance values following 16 weeks of raining were calculated for each athlete, and an interval of ± 75W was constructed to provide an initial screening threshold. Model simulations were run with 3.8×10^6 different parameter sets $\{k_g, k_h, \tau_g, \tau_h\}$ constructed by incrementing values in a grid-like fashion. Approximately 2000 potential parameter sets were obtained for each athlete in which the end-performance value resided within the threshold set. These parameter sets were then plotted and visually investigated for realistic developments over the 16-weeks. This process reduced the number of sets to approximately 10 for each athlete (Table 4.1). Parameter ranges for the intermediate athlete were $k_g: 0.5 - 2.5,\ k_h: 1.0 - 4.0,\ \tau_g: 14 - 37,\ \tau_h: 5 - 19$; And parameter ranges for the advanced athlete were $k_g: 0.5 - 4.5,\ k_h: 1.0 - 5.0,\ \tau_g: 6 - 25,\ \tau_h: 5 - 15$. A final selection was made (Table 4.2) based on further visual comparison to create upward trends with plateau, and ensuring parameter values and their ratios (k_h/k_g and τ_g/τ_h) were consistent with previous research (Pfeiffer, 2008).

Table 4.1: Athlete-specific parameter sets (k_g, τ_g, k_h, τ_h) creating realistic improvements. Top row (highlighted) for each athlete includes the parameter set used for the computer experiments. INT: Intermediate; ADV: advanced athlete.

Athlete	k_g	τ_g	k_h	τ_h	Athlete	k_g	τ_g	k_h	τ_h
INT	0.5	18	1	5	ADV	4.5	8	5	7
INT	2.5	14	3.5	9	ADV	3.5	6	4	5
INT	2.5	19	4	11	ADV	0.5	12	1	15
INT	0.5	37	1.5	9	ADV	1.4	10	2	7
INT	1	22	2	9	ADV	2.5	10	3	8
INT	1	19	1.5	10	ADV	0.5	20	1	9
INT	0.5	31	1	11	ADV	0.5	25	1	11
INT	1.5	26	2	17	ADV	1	19	1.5	12
INT	2.5	25	3	19	ADV	2.5	16	3	13

Table 4.2: Athlete-specific (true) parameters (k_g, τ_g, k_h, τ_h) and initial starting values (p^*) and final performance values $p(112)$. **INT**: Intermediate; **ADV**: Advanced athlete.

Athlete	Change (%)	Baseline performance	Final performance	True parameters			
		$\boldsymbol{p(0)}$	$\boldsymbol{p(112)}$	$\boldsymbol{k_g}$	$\boldsymbol{\tau_g}$	$\boldsymbol{k_h}$	$\boldsymbol{\tau_h}$
INT	~ 15%	4500	~ 5175	0.501	18	1.002	5
ADV	~ 5%	5250	~ 5500	4.501	8	5.002	7

4.3.5 Implementation

Power values were generated for each athlete with the training loads and parameters described above by simulating the Fitness-Fatigue model in equation 4.1 to represent true performance (Figure 4.4). Repeated simulations were then used to investigate the effects of error magnitude and testing frequency under parameters fitted to augmented true data. Measurement error was added to each true power value (in the initial eight-week model training block) to replicate testing in a real-world setting. Errors were added by random draws from a Gaussian distribution with mean zero and standard deviation representative of that obtained during a vertical jump. A review of literature identified that power is frequently measured via a force platform or linear position transducer (Cronin, Hing and McNair, 2004; Cormack *et al.*, 2008). The former measurement tool calculates power via force and velocity values obtained through integration, and the latter calculates power via force and velocity values obtained from differentiation of displacement data. Reliability studies have reported coefficients of variation (CoV) ranging from approximately 2 to 10%, with superior reliability obtained when using a force platform (Cronin, Hing and McNair, 2004; Cormack *et al.*, 2008). As a result, standard deviations for Gaussian errors were derived for both hypothetical athletes by multiplying each CoV value (2, 4, 6, 8, 10%) by the athlete's initial baseline power value p^* and dividing by 100 (Table 4.3). The advanced hypothetical athlete case was investigated further by repeating the experiment while incorporating the same absolute error used in the intermediate case, to facilitate controlled comparisons. To imitate different testing frequency states, a proportion of power values were isolated to recreate the real-world setting of measuring performance from once per week to each day (in unit increments of 1 day). For example, every seventh power value with error was selected from the model training block when imitating the once per week testing state. A total of 210 scenarios were investigated with each comprising 10^4 iterations per scenario (2.1×10^6 total iterations across all three cases: intermediate, advanced athlete, advanced athlete with intermediate errors). Model fitting was performed within a parallel computing framework, with parameter estimates obtained at each iteration via a nonlinear least-squares regression approach applying a limited-memory modification of the BFGS quasi-Newton algorithm (Byrd *et al.*, 1995a), from the optimisation package *Optim* (a part of the R *stats* package included with the standard kernel (R Core Team, 2020), v.3.4.4). Parameter

estimates from each fitting iteration were combined with the corresponding TRIMPs across the entire 16-week block (Ludwig and Asteroth, 2016), with predictions for the hold-out data used to obtain prediction errors for subsequent analysis.

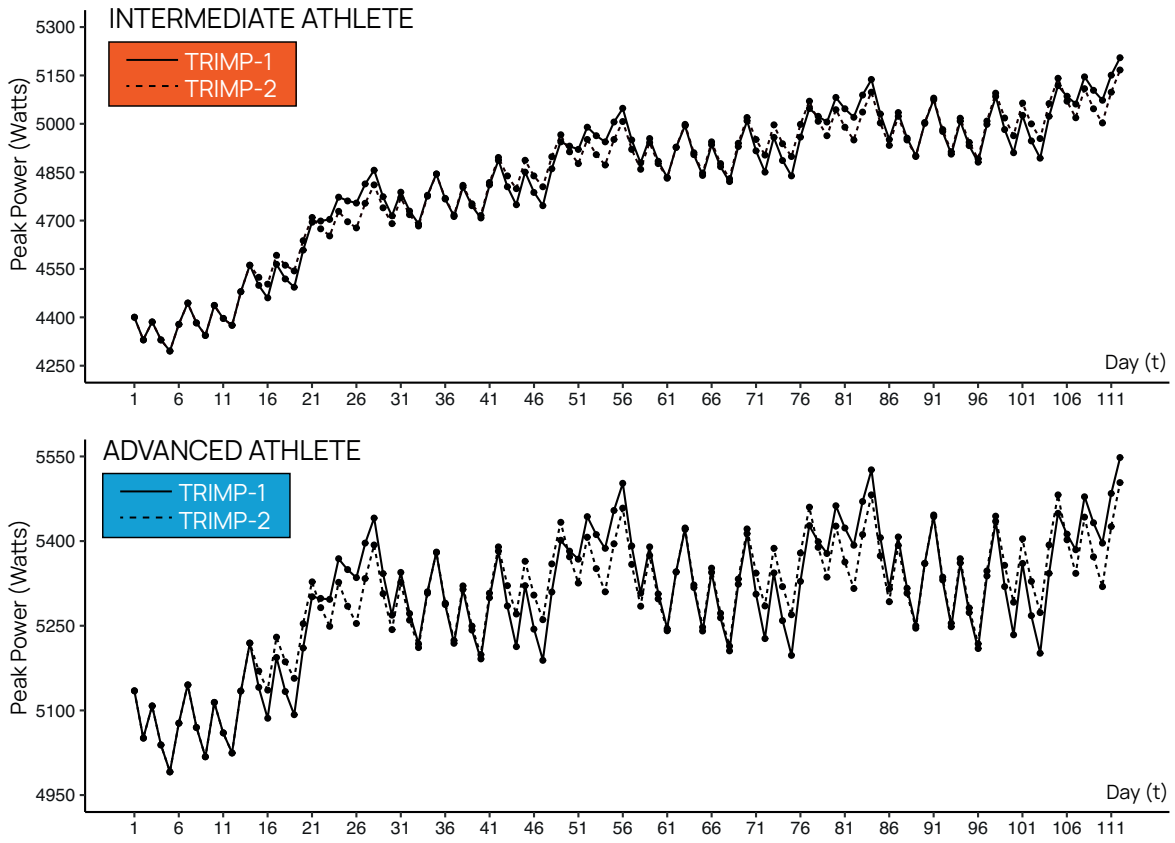


Figure 4.4: True vertical jump power values simulated across 16 weeks with two training load distributions (TRIMP-1, TRIMP-2) for the intermediate and the advanced athlete

Table 4.3: Standard deviation (W) of the Gaussian error distribution with mean 0, from which random measurement errors were drawn and applied to each known true value in the simulations

	SD of Gaussian error distributions for CoV % states				
Athlete	**2%**	**4%**	**6%**	**8%**	**10%**
Intermediate	90 W	180 W	270 W	360 W	450 W
Advanced	105 W	210 W	315 W	420 W	525 W

4.3.6 Statistical analyses

For each set of 10^4 fitting iterations, prediction errors were transformed into summary statistics representing distributional centrality DPE_m and spread DPE_s by calculating the median value and the distance between the 0.16 and 0.84 quantiles, respectively. Relationships between dependent variables (centrality: DPE_m, spread: DPE_s) and centered independent variables (measurement error and testing frequency) were quantified by multiple linear regression. Initially, the linear combination of measurement error and testing frequency expressed as continuous variables were entered into regression models. A second series of models featuring the linear combination and product of measurement error and testing frequency (interaction effect) were then included. Fit and suitability of each linear model was assessed with adjusted R^2 and residual analysis, respectively. Distributions of parameter estimates were described using descriptive statistics and ill-conditioning assessed via calculation of Pearson correlation coefficients (Hellard *et al.*, 2006; Pfeiffer, 2008).

4.3.7 Quality control

Systematic examination of the source code (formal code review) was performed pre- and post-simulation deployment to detect inaccuracies that would prevent successful implementation or cause erroneous results. Modules performing vital functions within the code were typically also isolated and unit-tested at the point of development within the main source code. A basic sensitivity analysis was conducted to assess the effects of different initial values on the non-linear least squares' optimisation function. The sensitivity analysis comprised fitting the least-squares algorithm with 100 different starting values across the parameter space, and no substantive changes on average were noted from code featuring a single set of starting values comprising the true parameters, although there was divergence for some starting parameters with poor objective value and toward the bounds. Optimisation convergence was set using a tolerance of 10^{-8} in reduction of the objective and found to be successful for approximately 99% of total parameters estimated within the experiment.

4.4 Results

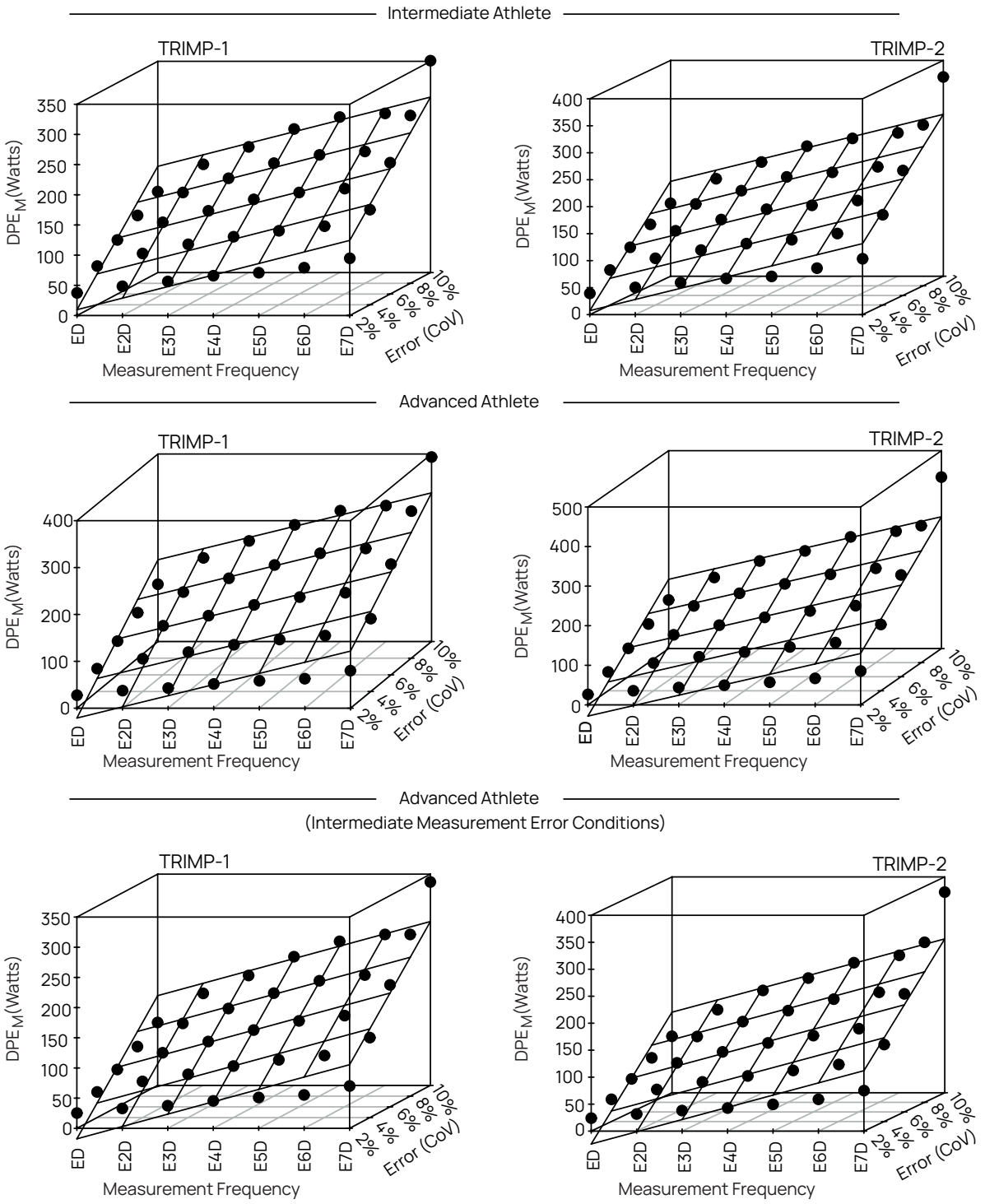
4.4.1 Prediction errors

All analyses revealed positive associations between dependent variables (prediction error centrality and spread) and the independent variables (measurement error and testing frequency). DPE_m was well explained by the linear combination of the two independent variables (Adjusted $R^2 = 0.89 - 0.94$) across all six athlete-TRIMP groupings (Figure 4.5) (Table 4.4). Regression coefficients for testing frequency ($\beta_1 = 19.1 - 26.4$) and measurement error ($\beta_2 = 20.8 - 25.4$) were shown to be significant ($P < 0.001$) for each model assessed. The inclusion of an interaction term significantly ($P < 0.001$) improved the fit of each model (Adjusted $R^2 = 0.96 - 0.98$) and demonstrated that the deleterious effects of increased measurement error on DPE_m was increased at lower testing frequencies. Similar results were obtained for DPE_s (Figure 4.6, Table 4.4), with strong linear relationships obtained with testing frequency and measurement error (Adjusted $R^2 = 0.98 - 0.91$). Again, each regression coefficient was found to be significant ($P < 0.001$), and all models were improved ($P < 0.001$) with an interaction effect demonstrating greater deleterious effects of measurement error at low testing frequencies. Comparisons of prediction errors between the advanced and intermediate athlete demonstrated a dependence on testing frequency. For high measurement frequencies, the advanced athlete experiment for both TRIMP distributions demonstrated systematically lower prediction errors compared to the intermediate athlete. This finding was obtained for both DPE_m (mean $\pm$ sd = 90 ± 47 vs. 103 ± 93 W, respectively) and DPE_s (mean $\pm$ sd = 168 ± 110 vs. 193 ± 93 W, respectively), despite larger absolute measurement error values inputted to advanced athlete experiment. However, when testing frequency was low, prediction errors were similar for both athletes, and in some cases, slightly larger for the advanced athlete. When the experiment was repeated using the same absolute error magnitude for both athletes, centrality and spread of prediction error were consistently lower for the advanced athlete across all conditions.

Table 4.4: Regression coefficient estimates and standard error (SE), residual standard error (RSE, 32 degrees of freedom), and adjusted R^2 values, for the centred independent variables (measurement error and testing frequency) with and without an interaction term, on the response variables (centrality DPE_m and spread DPE_s).

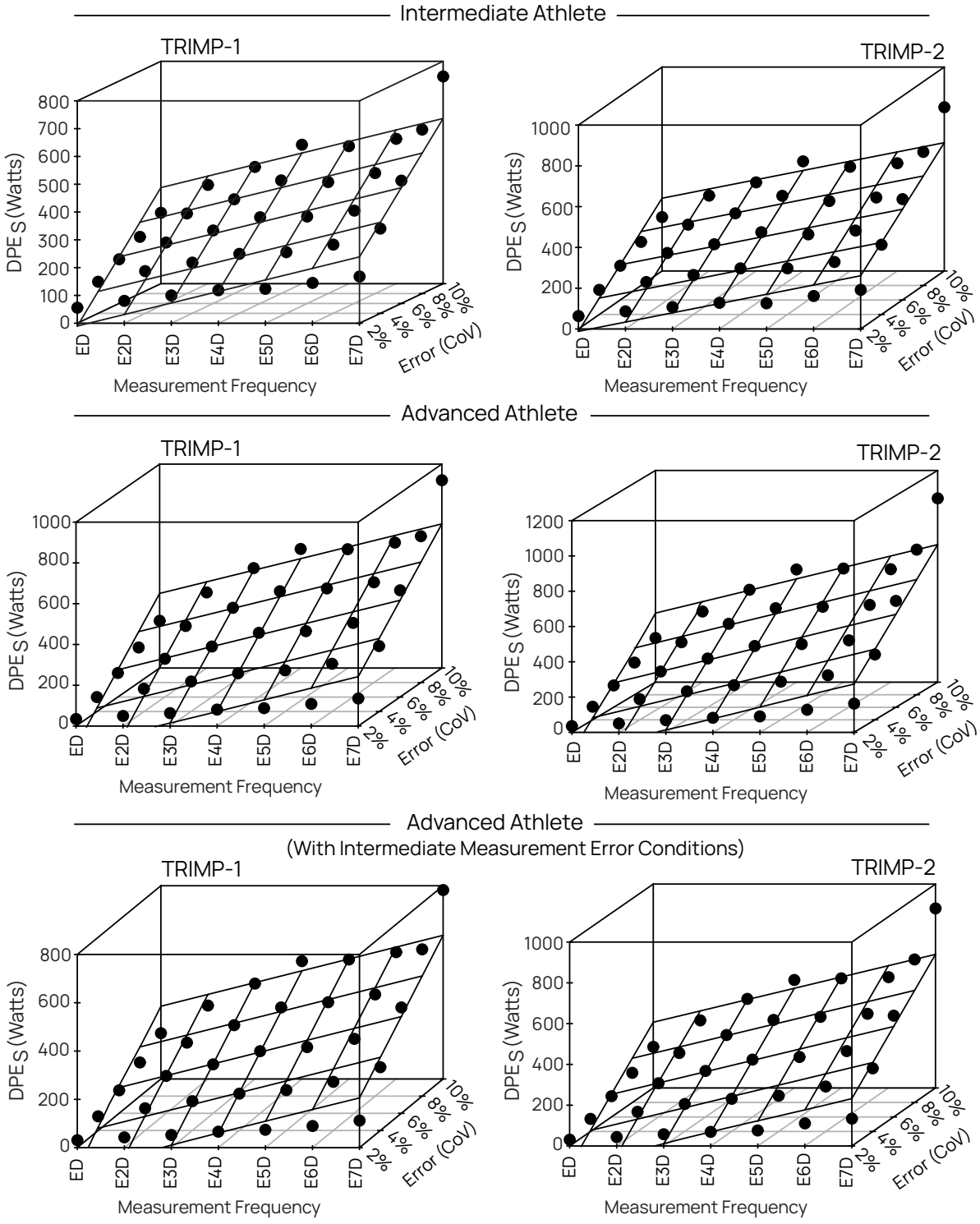
			Intermediate athlete		Advanced athlete		Advanced athlete +	
			T1	T2	T1	T2	T1	T2
$\boldsymbol{DPE_m}$ **(centrality)**	**Coefficient Estimates**	β_0	150.2	153.7	148.1	152.4	127.1	130.6
		β_1 (Freq)	19.1	20.7	23.8	26.4	20.5	22.6
		β_2 (Error)	20.8	21.1	24.4	25.4	20.8	21.6
	Standard Error	β_0 (Intercept)	3.1	3.6	4.4	5.3	3.8	4.5
		β_1 (Freq)	1.5	1.8	2.2	2.7	1.9	2.3
		β_2 (Error)	1.1	1.3	1.6	1.9	1.3	1.6
	Fit Statistics	R^2_{adj}	0.94	0.92	0.91	0.89	0.91	0.89
		R^2_{adj} interaction	0.98	0.97	0.98	0.96	0.98	0.96
		RSE	18.2	21.5	26.2	31.4	22.5	26.9
$\boldsymbol{DPE_s}$ **(spread)**	**Coefficient Estimates**	β_0 (Intercept)	292.1	308.8	304.1	332.5	253.5	277.4
		β_1 (Freq)	41.5	45.4	56.8	64.7	49	55.5
		β_2 (Error)	44.5	46	58.1	63.3	48.6	52.9
	Standard Error	β_0	8	9.6	12	14.4	10.4	12.3
		β_1 (Freq)	4	4.8	6	7.2	5.2	6.2
		β_2 (Error)	2.8	3.4	4.3	5.1	3.7	4.4
	Fit Statistics	R^2_{adj}	0.91	0.89	0.89	0.87	0.89	0.87
		R^2_{adj} interaction	0.96	0.94	0.96	0.94	0.97	0.95
		RSE	47.2	56.6	71.4	85.4	61.5	73

Notes: All fit statistics $P < 0.001$, 32 degrees of freedom. Advanced athlete + refers to the advanced athlete experiment that was performed with intermediate athlete measurement conditions (CoV %). T1 (TRIMPs-1), T2 (TRIMPs-2). R^2_{adj} is the adjusted coefficient of determination. (interaction) refers to the regression model which accounts for interactive effects between error and frequency.



DPE_M: Distributional centrality of absolute prediction error; **CoV**: Coefficient-of-Variation (%); **W:** Watts (Peak Power output); EnD: Every n Days, $n \in \{1,2,3,4,5,6,7\}$

Figure 4.5: Regression planes illustrating relationships between prediction errors (centrality of distribution) and independent variables (measurement error and testing frequency)



DPE$_S$: Distributional spread of absolute prediction error; **CoV**: Coefficient-of-Variation (%); **W:** Watts (Peak Power output); EnD: Every n Days, $n \in \{1,2,3,4,5,6,7\}$

Figure 4.6: Regression planes illustrating relationships between prediction errors (spread of distribution) and independent variables (measurement error and testing frequency)

4.4.2 Model parameter estimates

In general, model parameter estimates $(k_g, \tau_g, k_h, \tau_h)$ displayed a range of different distributions across scenarios. Estimates for the gain parameters (k_g, k_h) were unstable for both athletes, with boundary values frequently obtained when testing frequency was low. This effect was magnified when low testing frequency was combined with high measurement error. Distribution of the decay parameters (τ_g, τ_h) became increasingly right-skewed for both athletes as measurement error increased. This effect became more pronounced when larger measurement errors were combined with low testing frequency. Correlations between parameter estimates ($N = 350{,}000$ per athlete-TRIMP grouping; Table 4.5) revealed strong associations between gain parameters (k_g, k_h: $r_{mean} = 0.978 - 0.99$) for both athletes, thereby demonstrating ill-conditioning. Low to moderate strength negative correlations were also obtained between k_g and τ_g ($r_{mean} = -0.57\ to\ -0.38$), and k_h and τ_g ($r_{mean} = -0.58\ to\ -0.39$) for both athletes.

Table 4.5: Correlations between estimated model parameters for scenarios within athlete-TRIMP groupings (*N* = 10000 parameter sets, per scenario). Highlighted is the main finding indicating ill-conditioning of the model relating to non-independence of the scaling factors.

	Correlation Coefficient							
	INT: TRIMPS-1		**INT: TRIMPS-2**		**ADV: TRIMPS-1**		**ADV: TRIMPS-2**	
Model Parameter	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
$k_g - k_h$	0.99	0.009	0.990	0.013	0.98	0.018	0.978	0.025
$k_g - \tau_g$	-0.57	0.031	-0.560	0.036	-0.38	0.057	-0.382	0.056
$k_g - \tau_h$	0.28	0.231	0.237	0.221	-0.05	0.112	-0.084	0.089
$k_h - \tau_g$	-0.58	0.033	-0.576	0.036	-0.39	0.049	-0.398	0.050
$k_h - \tau_h$	0.23	0.253	0.186	0.244	-0.11	0.125	-0.151	0.100
$\tau_g - \tau_h$	0.16	0.354	0.217	0.348	0.63	0.278	0.689	0.219
$\tau_g - k_g/k_h$	-0.09	0.174	-0.073	0.165	-0.13	0.233	-0.049	0.195
$\tau_h - k_h/k_g$	0.06	0.225	0.039	0.213	0.19	0.114	0.150	0.104

INT: Intermediate | **ADV**: Advanced | **SD:** Standard Deviation

4.5 Discussion

The present study comprised a unique and efficient design to investigate prediction accuracy of the standard fitness-fatigue model in a strength and conditioning context. The *in silico* approach provided an effective method to systematically assess the effects of two key challenges in athlete training modelling, namely, controlling measurement error and identifying appropriate measurement frequencies (Pfeiffer, 2008; Clarke and Skiba, 2013; Bourdon *et al.*, 2017). Whilst it is unrealistic to expect an athlete will respond deterministically to a series of training loads, the approach and underlying assumptions adopted in the present study provide important general information and informative lower bound cases for researchers and practitioners to consider. That is, the approach can identify and rule out specific practices that have no potential to be successful in real-world settings (but not rule in other practices). The key findings of this study indicate that increased measurement error and reduced testing frequency across standard ranges encountered in practice meaningfully increase prediction errors. Additionally, variation in prediction errors were well explained by the simple linear combination of measurement error and testing frequency (Adjusted R^2 = 0.87-0.94). Regression coefficients showed that for every 1% increase in CoV, the distribution of prediction errors increased by approximately 21-25 W in centrality (DPE_m) and 45-63 W in spread (DPE_s). Similarly, models showed that for a single day reduction in testing frequency, the distribution of prediction errors increased by approximately 19-26 W in centrality (DPE_m) and 42-65 W in spread (DPE_s). Theoretically, the standard fitness-fatigue model and traditional non-linear least squares methods used to obtain parameter estimates should demonstrate poor performance with high measurement error. The results of the experiments across the three hypothetical athletes (intermediate, advanced, advanced +) support this notion and show that if observed scores in a given performance test comprise error of more than 2-5% of an athlete's baseline score, they are unlikely to be suitable for use with the fitness-fatigue model due to unacceptable prediction accuracy even in this most optimistic scenario where performance is directly specified by the model. The results demonstrate that when the model is fit to moderately inaccurate data (comprising error more than 5% CoV) predictive errors become unacceptably high across all frequency conditions. For example, if measurements comprised ~6% CoV, the results of this study suggest that even under high testing frequencies, prediction errors of +150W should be expected for intermediate/advanced athletes. For further context, an error of 150 W is equal to approximately 3% of the baseline scores, with total improvement across the entire training phase set at 5 and 15% for the advanced and intermediate athlete, respectively. Furthermore, these computer experiments under deterministic assumptions represent a lower bound case, where additional real-world factors will further increase prediction errors.

4.6 Summary, conclusions, and practical recommendations

Whilst the results of the computer experiment presented here provide novel insights into the effects of measurement error and testing frequency on fitness-fatigue model prediction accuracy, there are a range of complex additional factors that would be expected to influence model prediction of an athlete's response to training. The primary aim of this study was to generate lower bound estimates, where even in the absence of these additional factors, certain measurement errors and testing frequencies conditions would be considered ineffective to warrant use in practice or research. Collectively, the results of this study indicate that practitioners and researchers should focus on relevant performance tests that generate highly reliable data ($\leq$ 4% CoV). Additional processes including taking the average of multiple trials and filtering techniques can also increase reliability and should be considered. Even under high frequency conditions, the results of this study demonstrate that accurate predictions are not likely if measurement error is not minimised. Prior to investing time in data collection, it is recommended that practitioners and researchers adopt an approach similar to the one applied here, where various measurement error and testing frequencies can be applied to training loads and adaptive rates realistic to each athlete being studied. Finally, it is recommended that future research investigating the use of fitness-fatigue models report prediction accuracy using cross-validation to appropriately evaluate the utility of the model to practitioners within the field of sport science.

Chapter 5: Suitability of a quasi-Newton algorithm for estimating FFMs: Sensitivity, troublesome local optima, and implications for future research (An *in silico* experimental design)

5.1 Preface

The typical experimental approach in the study of fitness-fatigue models across prior research has involved collecting laboratory or field data, using this data to fit unknown FFM parameters via nonlinear least-squares regression (NLS), and then testing ensuing model predictions on in-sample data to derive metrics of goodness-of-fit. Explicitly, data spanning one or more training periods is used which comprises: 1) training logs containing key information (e.g., volume, intensity) to quantify model input (training load) via a nominated method (e.g., Banister's TRIMP); and 2) measured values of physical performance collected at semi-regular or irregular intervals over the time-series. Although this type of applied research is a necessary component of model evaluation, previous experimental designs have frequently overlooked both evident and probable issues with applied methods, introducing a noticeable pattern of unconscious replication across research (see Table B-2, appendix B). Most significantly, repetition of issues pertaining to model fitting and evaluation of predictions. A balance should be struck between the volume of applied modelling work, and the theoretical examination of experimental design and operational aspects of FFM study that should inform applied research. Criticisms of the singular testing of fitted models on in-sample data, as opposed to inclusion of some form of out-of-sample testing, have already been outlined in this thesis (chapters 2 and 4) and recent research (Stephens Hemingway *et al.*, 2019, 2021; Imbach *et al.*, 2020). In Chapter 6, this problem is solved at a practical level by provision of a time-series cross-validation approach implemented in the programming language R that can be used by researchers to test and fit FFMs. In addition, very limited direct attention has been given to the suitability of common approaches used to fit FFMs, as well as issues that researchers are likely to encounter when attempting to solve the data-fitting problem, such as the presence of many local minima within the search space.

The optimisation problem of fitting FFMs to data may be split broadly into two steps: 1) problem formulation, i.e., selection of an appropriate objective function; and 2) problem solving, i.e., selection of an appropriate algorithm to find the best minimiser or maximiser of the objective under the inputs (e.g., parameters). Problem formulation for fitting FFMs has typically used the sum-of-squares error function as the objective (abbreviated RSS or SSE) within a nonlinear least-squares (NLS) approach, and more recently the log-likelihood function within a maximum likelihood estimation (MLE) approach (Shrahili, 2014; Busso, 2017; Scarf *et al.*, 2019; Swinton *et al.*, 2021). RSS appears to be a common choice as an objective function for general model fitting. However, less obvious is which - if any - prospective algorithm represents a robust and reliable method to solve an NLS problem in the

case of fitting an FFM to data. With the exception of a recent study by Connor and O'Neill (2020), no direct investigation of algorithms to solve the NLS problem when fitting FFMs to data has been conducted. In their recent preprint paper, Connor and O'Neill (2020) evaluated the performance of an evolutionary algorithm (differential evolution) as an alternative to a typical first order algorithm used previously for fitting FFMs under NLS (Connor and O'Neill, 2020). Specifically, model prediction errors (RMSE) over a hold-out dataset were quantified and compared from fitted solutions found by two algorithms, differential evolution and the quasi-Newton algorithm L-BFGS-B (a first-order method which attempts to mimic a second-order method via a secant method). Data for their study were drawn from the dataset published in Clarke and Skiba (2013). Use of real data represents the biggest limitation of Connor and O'Neill's approach, as although the performance of the two algorithms is comparable based on obtained solutions (i.e., associated objective function value and prediction errors), the global minimum of the objective remains unknown and so the approach cannot be used to identify the reliability of any particular algorithm to globally minimise the NLS problem. This is because the RSS associated with fitted solutions are almost always non-zero, primarily due to model misspecification, and therefore no reference value exists for comparison in the real-world. That is, the absolute minimum (best solution) of the objective function remains unknown, and as such it cannot be determined that a given solution represents a global minimum in the case of the FFM. The caveat to this is that with increasing numbers of free parameters in the model, the closer the fitting algorithm may be able to get to a zero-residual solution for real data, due to larger degrees of freedom. In contrast, an *in silico* design may offer an innovative approach for directly studying the optimisation process, including the suitability of different algorithms to globally minimise the objective under best-case conditions, and identification of complexities in the search space that may affect algorithm behaviour and therefore have implications for choice of solver in real world experiments. From the deterministic assumption of an FFM as the truth function of athlete response, synthetic performance values can be generated via simulation of the model under a pre-defined set of true parameters and mock training load values (model inputs), without additional measurement error. In this manner, a global minimum is known for the model (at the true parameters) associated with the simulated data, which represents a hypothetical "measured performance" profile. This simulated data can then be fed into a fitting process to test (in the most basic case of zero noise) the ability for a particular algorithm to recover the true parameters under an approach such as NLS. For some algorithms, such as first and second-order methods that are by definition, local optimisers, this process would be repeated from multiple starting points to derive an estimate of the algorithm's sensitivity to starting point in the parameter space (i.e., the frequency with which it can converge to the true parameters). For evolutionary algorithms that are stochastic by nature, the equivalent is repetition of the fitting process from many different random starting populations.

This type of experimental approach, contained within an extensive computational framework comprising multiple iterations, is unique for studying the estimation process as it would otherwise be impossible to estimate properties such as starting point sensitivity in the same manner when using real data, due to a lack of a known global minimum for the data. Although there are limitations to this type of simulation approach, including the knowledge that the FFM does not fully specify the training response and therefore no zero-residual solution will exist under real-world data, the benefits of this perspective such as the capability to study the fitting process under a high number of iterations and extensive range of conditions provide strong argument for its uptake as one part of the theoretical picture. In addition, it is a cost-effective approach in comparison to laboratory experiments and/or conducting and monitoring training interventions. It may be also used prospectively to provide an indication of how to appropriately tune evolutionary algorithms specifically for the FFM fitting problem, beyond standard rules of thumb.

In the study to follow, it is hypothesised that for the standard and fitness-delay FFMs, there is a low likelihood of finding a global minimiser under NLS from a "one-shot" optimisation run of a first or second-order algorithm that requires starting values, due to the presence of many local extrema (minima and/or saddle points) that cause these types of algorithms to fall short of the best solution depending on starting point. Correspondingly, it is suggested that the data fitting problem is more challenging than previous research has assumed or recognised, and that this has gone unnoticed across the previous experimental literature where estimates have been derived seemingly from one-shot runs of local optimisers. It is argued that had previous research performed multiple searches (initiated from different starting points) when fitting FFMs under these types of algorithms the problems suggested may have been spotted earlier and questions raised surrounding starting-point sensitivity and subsequent suitability of typical optimisation algorithms adopted. In the spreadsheet of Clarke and Skiba (2013), researchers are advised to "Run the solver again a few times to ensure convergence", suggesting that the authors felt this was sufficient to avoid local sensitivity of a common search algorithm. In addition, they do not provide reasoning for this statement or discussion of the wider implications of this type of sensitivity, nor a suggestion or justification of which particular algorithm to use.

The aim of this chapter, and the associated research paper [5] is to investigate the suitability of a typical first-order algorithm (with second-order approximation) to solve the FFM fitting problem via NLS. The simulation methods used are similar to those developed in chapter 4 and described so far in this preface. Of particular focus is assessment of the hypothesis described above, via study of starting point

[5] Stephens Hemingway, B., Swinton, P. A., & Ogorek, B. (2021). The suitability of a quasi-Newton algorithm for estimating fitness-fatigue models: Sensitivity, troublesome local optima, and implications for future research (An *in silico* experimental design). *SportRxiv (Preprint)*. 10.31236/osf.io/dx7gm

sensitivity of the L-BFGS-B algorithm under best case conditions (i.e., deterministic assumption and no noise), existence of local optima in the search space, and the subsequent implications of any findings on the interpretation of estimates in prior research and considerations for future work.

5.2 Introduction

The standard FFM emerges from a linear system of first-order ordinary differential equations (Banister *et al.*, 1975). When solved, this ODE system yields a nonlinear function in the unknown model parameters (Busso *et al.*, 1990; Morton, Fitz-clarke and Banister, 1990; Clarke and Skiba, 2013). Other FFMs arise from similar ODE systems, although some involve non-linear system dynamics (Turner *et al.*, 2017), higher derivatives (Calvert *et al.*, 1976) (see appendix A), and recursion (Matabuena and Rodríguez-López, 2016, 2019). Therefore, fitting an FFM constitutes a nonlinear optimisation problem in its model parameters. FFMs dependent on time-invariant parameters (Banister *et al.*, 1975), or in special cases time-varying parameters (Busso *et al.*, 1997; Kolossa *et al.*, 2017) that cannot be inferred from observation and must instead be estimated from quantified training load and measured performance data (Stephens Hemingway *et al.*, 2021). The fitting process takes as input a time-series of measured performances (denoted p) and training load values (denoted ω) and provides as output model parameter estimates ($\theta \in \mathbb{R}^{+}$) that give good or preferably the best possible agreement between iteratively computed model values (denoted $\hat{p}$) and measured data (p). Essentially, to fit an FFM, a researcher or practitioner requires a series of suitable training load and performance measurement data, and a method to alter the parameters to best match these through an optimisation perspective.

The most common optimisation approach for fitting FFMs has been NLS (eq. 5.1) (Hellard *et al.*, 2006; Pfeiffer, 2008; Clarke and Skiba, 2013; Proshin and Solodyannikov, 2018; Connor and O'Neill, 2020), or a maximum likelihood perspective (Busso, 2017; Scarf *et al.*, 2019). Least-squares and maximum likelihood estimation coincide under the assumption of independent and identically distributed Gaussian model errors. Of the two, NLS has represented the most accessible approach across prior research and involves minimising the sum of squared deviations (also called errors) between modelled and measured performance (eq. 5.1) (a twice differentiable function) using some iterative algorithm.

$$\min \sum_{i=1}^{m} (\hat{p}_i - p_i)^2 \tag{5.1}$$

Where in (eq. 5.1) i is an index over a set of m of data points $\{(p_1, \hat{p}_1), (p_2, \hat{p}_2), \dots, (p_m, \hat{p}_m)\}$ that represent measured (p) and modelled $(\hat{p})$ criterion performance values at specific integer time points $t_i \in \mathbb{N}$. The term $\hat{p}_i$ is determined by the FFM function $f\left(t_i,\ \theta,\ \{\omega_1, \dots, \omega_{t_i}\}\right)$ that not only

depends on the time-step input (i.e., $\Delta_t = 1$) up to time t_i (i.e. training load series) $\{\omega_1, \omega_2, \dots, \omega_{t_i}\}$ but also on n model parameters (θ) with $m \geq n$. For example, with the standard FFM (eq. 5.2) the parameters θ comprise the set $\{p^*, k_g, \tau_g, k_h, \tau_h\}$, where p^* is an additive term representing baseline performance, τ_g, τ_h are the decay time constants on fitness and fatigue, respectively, and k_g, k_h are the associated scaling factors. NLS regression problems are typically solved using general minimisation methods, where the algorithm evaluates the cost function (eq. 5.1) and uses specific update and stopping criteria to travel the available parameter space to search for the best possible set (i.e., the absolute minimum of the function).

Fitting an FFM via NLS in practice assumes that a unique optimal solution exists and can be found by the algorithm applied. However, this idealistic scenario may not hold for two reasons: 1) the absolute minimum may not be unique; and 2) local minima, saddle points, and/or plateau features may exist that cause problems for certain algorithms. The FFM in basic form is a model in five dimensions (Banister *et al.*, 1975), or six if a delay on fitness is also included (Calvert *et al.*, 1976). Therefore, the parameter surface cannot be plotted or visually inspected via standard techniques to assess convexity. If there exist different parameter sets in the domain that share the same global minimum under standard NLS, then there is a situation where parameters aren't uniquely identified without additional constraints or regularisation terms. However, more likely is that problems with the typical FFM fitting process will stem from the existence of local minima, saddles, or plateau features that cause the algorithm to converge to a solution not equal to the global minimum, or become lost (Philippe *et al.*, 2018). Local optima can provoke sensitivities in the fitting process for first and second-order algorithms that are by definition local optimisers. This manifests as sensitivity to initial parameter estimates (i.e., the starting point the algorithm initialises the search from). The extent of starting point sensitivity is largely unknown in the context of FFMs for common algorithms adopted and has not been studied directly. Given this concern, research reporting a single model solution derived from 'one shot' minimisation of NLS via typical first and second-order algorithms is fundamentally limited by possible uncertainty as to the suitability of fitted estimates as global minimisers. Therefore, the primary aim of the experiment was to study the sensitivity of a quasi-Newton algorithm to selection of initial estimates, and the existence of local optima, when fitting an FFM under an NLS perspective. A secondary aim was to examine the implications of any findings in relation to previous research as well as considerations for future investigations. The aims of the experiment were addressed through an *in silico* (computer experiment) approach that adopted a deterministic assumption that the FFM completely specified athlete response. Under this assumption, two FFMs (standard, and fitness-delay model) were simulated under a set of hypothetical model inputs and manually selected parameter values (for each model), generating in the process a set of synthetic performance data. The parameter values represented *true* values for the model under the deterministic assumption, associated with the

synthetic performance data. The two FFMs were refitted to the synthetic performance data without any noise by a quasi-Newton algorithm, under the same training load inputs, in a repetitive fashion starting from multiple points in the parameter space. This allowed starting point sensitivity of the algorithm to be assessed under best case conditions (no noise), and identification of the presence of local optima in the search space.

5.3 Materials and methods

5.3.1 Experimental approach to the problem

An *in silico* approach was developed, employing a first-order search algorithm (with second-order approximation) to fit two FFMs (eq's. 5.2, 5.3) from multiple starting values to associated synthetic performance data, in an iterative fashion. The performance data (a set of) was generated for each model under pre-defined true parameters and training loads (model inputs) via model simulation. At each iteration in the experiment, the algorithm was initialised from a different starting point (selected sequentially from a large grid of pre-determined values) and the appropriate model fit to the associated performance data via successive minimisation of the NLS objective function (eq. 5.1). The goal of the optimisation algorithm at each iteration was to try and recover the true parameters (global minimum) in the case where no additional noise exists. A *scenario* was defined as the combination of the model involved, and the proportion of simulated data used in the fitting process, with this latter factor described shortly. The total number of iterations in each scenario was equal to the total number of starting sets in the grid, which was therefore also equivalent to the total number of fitted estimates obtained. The synthetic model input values (i.e., daily training loads) used in the experiment were manually constructed to exhibit a realistic distribution (with regard to pattern, shape, and relative magnitude). Additionally, the true parameter values were selected such that the simulated performance values represented realistic performance change and variation over time.

Although the assumption of a completely deterministic model of athlete response is unrealistic due to simplification by design within the modelling process, this experimental approach is believed to be reasonable in a research context to enable lower-bound study of the fitting process in a manner not possible with real data. Furthermore, the simulated performance data were not unreasonable with regard to change in the performance profile, and true parameters (such as the decay constants) were chosen as to be interpretable with regard to model dynamics (Stephens Hemingway *et al.*, 2021). In the real world, fitting FFMs involves non-zero (possibly large) residual solutions that make it impossible to be sure that the fitted estimates represent a unique global minimum. In contrast, the approach developed in this experiment allows the convergence to the true parameters representing a global minimum to be reliably assessed for different initial estimates (starting values) used by the

algorithm, and when fitting to different proportions of the data (i.e., a lower measurement frequency). In the experiment, this second factor in each scenario (proportion of fitting data used) was contextually referred to as the *measurement frequency* based on its correspondence with the availability of data in practice. A reduction in measurement frequency was reflected by repeating the process described above for each model whilst fitting to a reduced subset of the simulated data. Three frequencies were studied: 1) *Every day* (ED) equivalent to 100% of the data; 2) *Every 2 days* (E2D) equivalent to approximately 50% of the data; and 3) *Every 3 days* (E3D) equivalent to approximately 33% of the data. At the heart of the research is to determine whether the fitting algorithm adopted is suitable for use in practice based on its ability (or lack of) to consistently recover the true parameters regardless of starting point or a decrease in volume of data supplied (given that no additional noise is incorporated). A flowchart detailing the computational process is presented in Figure 5.1, and the algorithm for the experimental flow stated in Appendix C-1.

The standard model (Banister *et al.*, 1975)

$$\hat{p}(t) = p^* + \underbrace{k_g \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_{g_1}}}}_{\text{fitness component}} - \underbrace{k_h \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_h}}}_{\text{fatigue component}} \tag{5.2}$$

The fitness-delay model (Calvert *et al.*, 1976)

$$\hat{p}(t) = p^* + \underbrace{k_g \sum_{i=1}^{t-1} \omega_i \cdot (e^{\overbrace{\frac{-(t-i)}{\tau_{g_1}}}^{effect}} - e^{\overbrace{\frac{-(t-i)}{\tau_{g_2}}}^{delay}})}_{\text{fitness component}} - \underbrace{k_h \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_h}}}_{\text{fatigue component}} \tag{5.3}$$

Where for both models, $g(0) = h(0) = \omega_0 = 0$.

5.3.2 Development of the synthetic model inputs (training loads)

The series of model input (training loads) used in the experiment were constructed based on concepts of daily variation and a subtle wave-like profile (Baker, 1998). Pre-simulation scaling of the load series was set to a maximum of 5% of the true baseline performance (additive term p^*), to avoid excessively small values of the scaling parameters (k_g, k_h), but the values were otherwise arbitrary. Primary emphasis was placed on the shape of the distribution (Baker, 1998) and reasonable relative differences in magnitude (i.e., between session), rather than absolute scale. Figure 5.2 provides a visual plot of the training loads developed over a 150-day time-series.

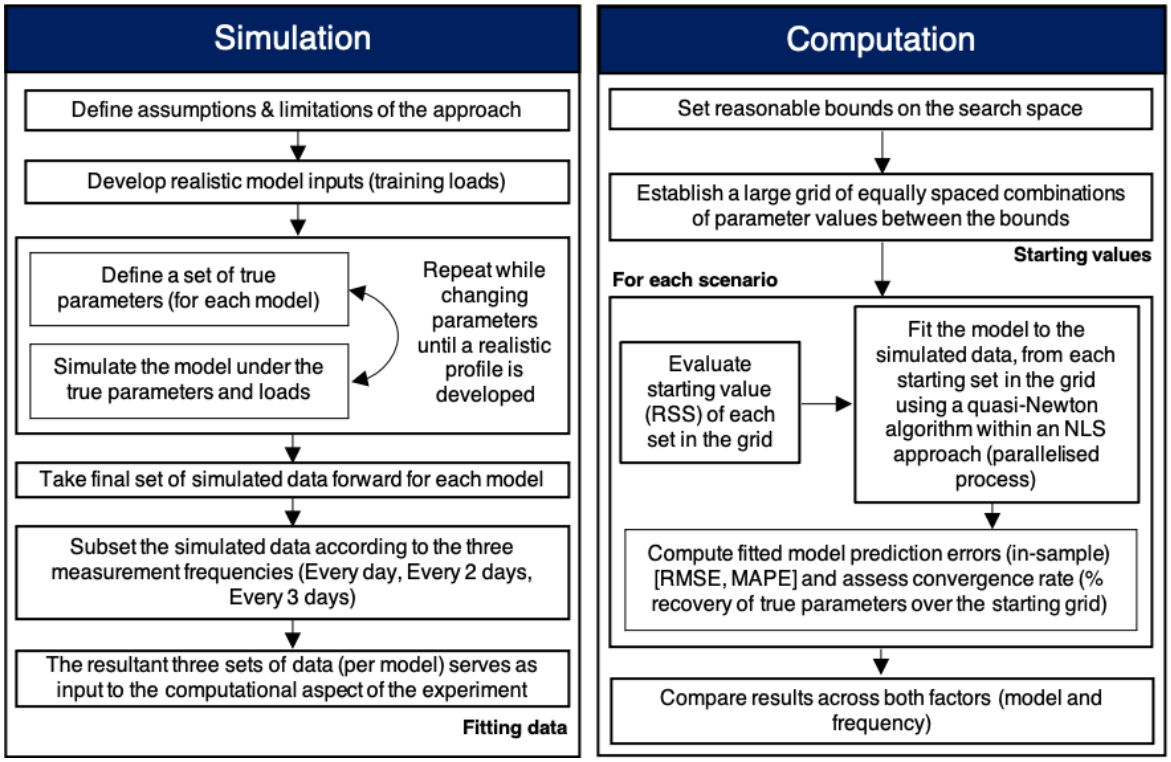


Figure 5.1: Flowchart describing the experimental approach to the problem (for both models)

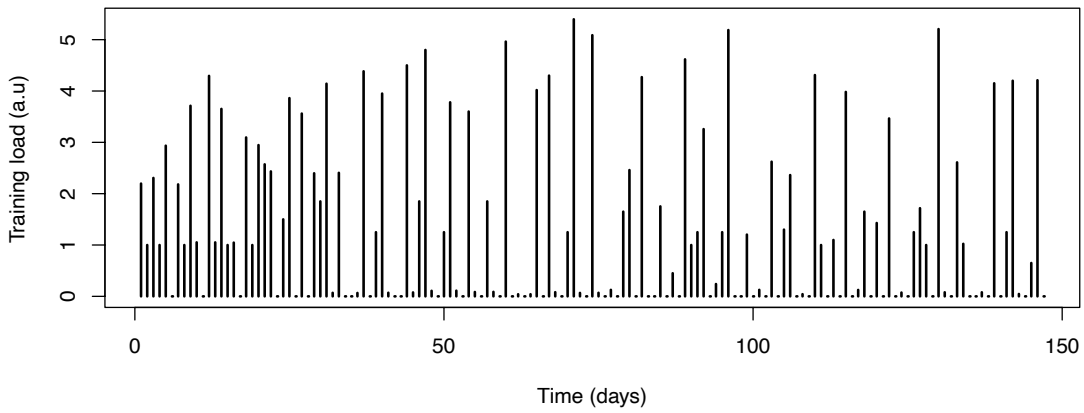


Figure 5.2: Hypothetical training load values for the experiment with realistic variation and wave-like profile

5.3.3 Simulated performance data

As described, the set of simulated daily performance data for each model were developed by first selecting a set of 'true parameters' associated with each model (Table 5.1) that produced a realistic profile with regard to magnitude and shape of performance change under the training load series. To assess magnitude, improvement in the maximum bench press (kg) for a moderately trained athlete over

a 150-day period was used as a guide. Although the introduction of context is useful, it is also recognised that is it not necessary here and that these values could be left undefined without affecting validity of the experiment. The data used to fit (train) the model were developed by isolating a proportion of the model-generated performance values according to the measurement frequency condition in the scenario. In contrast to the first experiment in this research project (Stephens Hemingway *et al.*, 2019) (chapter 4), no noise was introduced to the training data as this would disrupt the presence and purpose of establishing a known global minimum. The total proportion of values isolated from the simulated data and used to fit the model was defined by three measurement frequency conditions described previously (ED, E2D, E3D) and reflected by subsets of 100%, ~50% and ~33% respectively. Figure 5.3 shows the set of simulated performance data for each model, and figure 5.4 illustrates the underlying component profile.

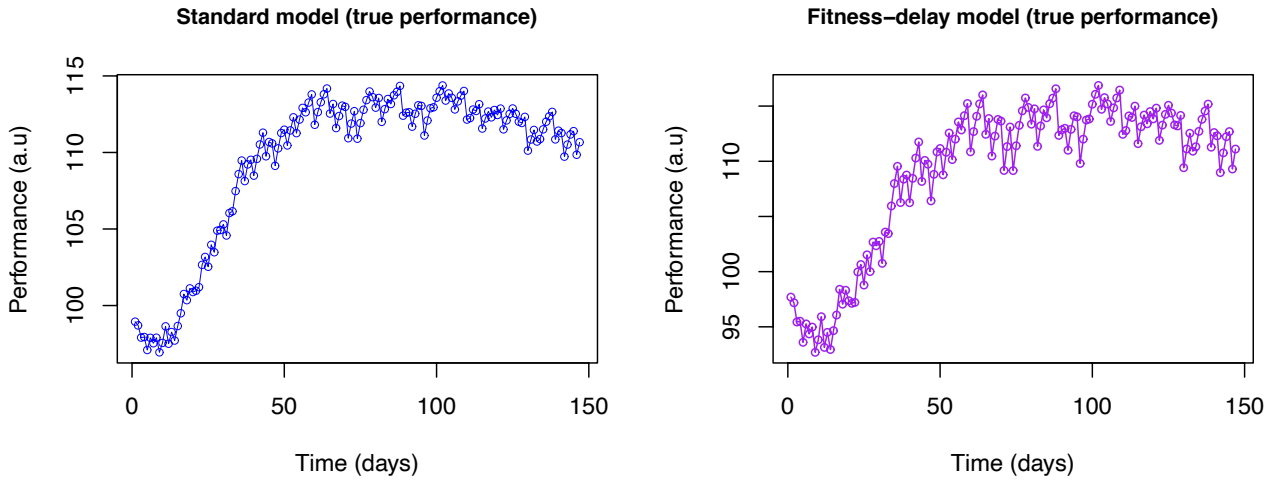


Figure 5.3: Simulated performance data generated for each model: Standard model (left) and fitness-delay model (right)

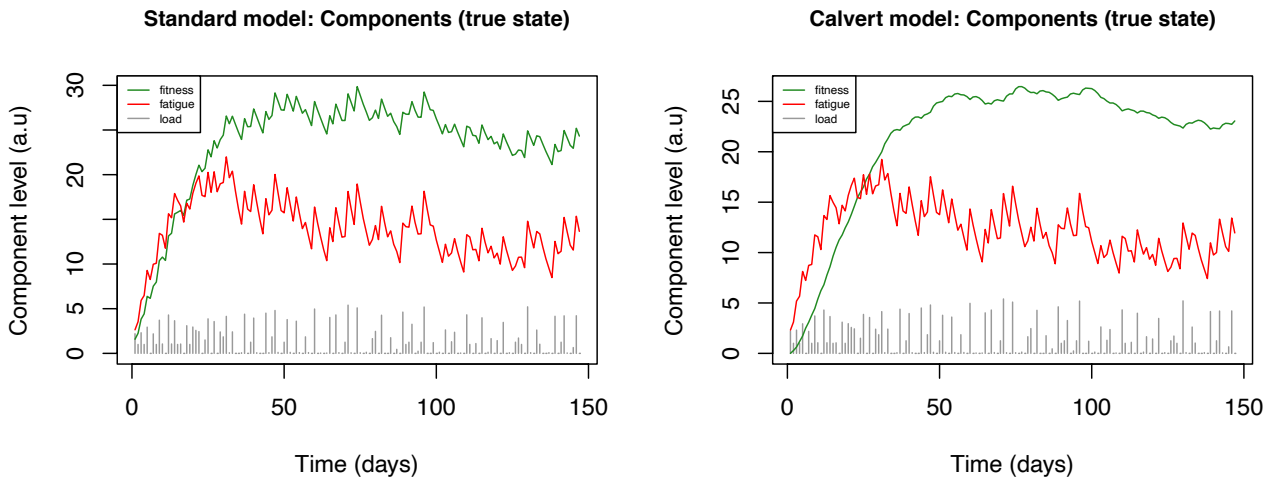


Figure 5.4: Simulated model component states: standard model (left) and fitness-delay model (right); fitness component (green), fatigue component (red), training load (grey)

Table 5.1: True parameters used in the experimental (simulated data) for each model

	p^*	k_g	k_h	τ_h	τ_{g_1}	τ_{g_2}
Standard model	100	0.72	1.2	8.6	28.5	-
Fitness-delay model	100	0.72	1.05	8.6	32.5	4.3

5.3.4 Computational framework

A total of 6 scenarios were investigated comprising conditions of two factors (model [2] × measurement frequency [3]), and the following process was employed for each: 1) reasonable bounds on the parameter space were established on each parameter (see Table 5.2, equivalent across scenarios); 2) between the bounds, a discrete grid of feasible parameter combinations was constructed, with equally spaced step changes in each parameter (See Figure 5.5); 3) the value of each parameter set in the discrete grid was evaluated via the objective function (eq. 5.1) representing the 'fitness' (cost) of each starting point prior to fitting; 4) in an iterative and parallelised model fitting process, each parameter set in the grid was applied as the starting point of the quasi-Newton algorithm; that takes as input the training load series and simulated criterion performance values (fitting data, see 5.3.3), and returns as output a set of fitted parameter estimates via minimisation of NLS; 5) fitted parameter estimates were combined with the training load series to generate fitted model predictions (project performance) and these were transformed into in-sample goodness-of-fit statistics: the root mean percentage error (RMSE), and mean absolute percentage error (MAPE).

Bounds on the parameter space were imposed in the form of box-constraints and these were chosen to not be too tight nor to close to the true parameters, but also not too large as to be physiologically non-interpretable (in particular for decay constants and additive term) or more than five times the magnitude of the difference between the performance values and the training load values (for scaling factors). For the scenarios comprising the standard model (eq. 5.2), each grid of starting values comprised a total of 10^5 parameter sets, and for the fitness-delay model scenarios (eq. 5.3) each grid comprised a total of 7^6 sets. A slightly larger size of grid for the fitness-delay model allowed for a reasonable step-size between changes in parameter values over the bounds to be maintained. A toy example of this method used to construct the grid is illustrated in Figure 5.5.

Table 5.2: Bounds on the parameter space and starting grid for each model

	Standard model (eq. 5.2)		Fitness-delay model (eq. 5.3)	
Parameter	**Lower Bound**	**Upper Bound**	**Lower Bound**	**Upper Bound**
p^*	60	140	60	140
k_g	0.01	5	0.01	5
k_h	0.01	5	0.01	5
τ_h	1	50	1	50
τ_{g_1}	1	50	1	50
τ_{g_2}	-	-	1	50

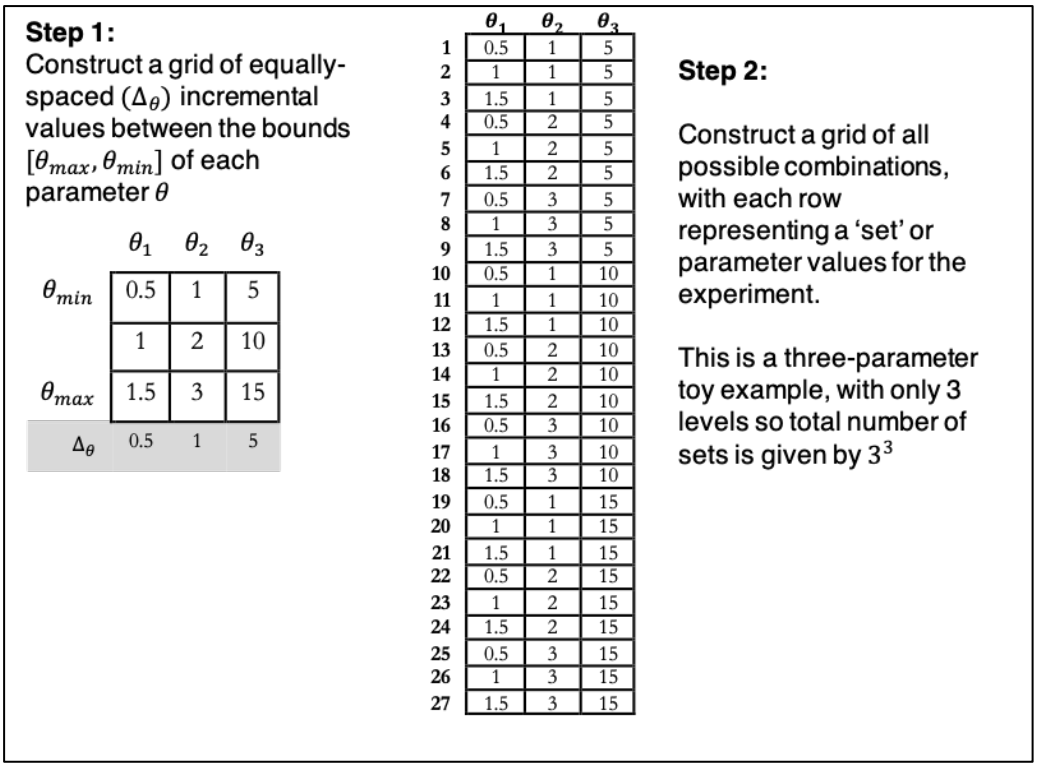


Figure 5.5: An illustration of the method used to construct the grids (generalised example)

5.3.5 The quasi-Newton search algorithm

Most iterative algorithms to find minimisers of smooth NLS problems require computation and storage of first or second-order derivatives of the objective function (Mohammad and Waziri, 2019). For example, Newton-type methods require the exact Hessian, Gauss-Newton (GN) and Levenberg-Marquardt (LM) methods make use of the first-derivative information and ignore the second-order part of the Hessian, and quasi-Newton (QN) methods approximate the Hessian rather than having to

iteratively compute and store it at each step (Mohammad and Waziri, 2019). However, each of these methods have associated limitations. For example, the exact second-order derivatives of the objective function are not normally available at a reasonable cost, and the analytic expression is often intractable for non-linear problems (Dennis Jr and Schnabel, 1996; Sun and Yuan, 2006). Thus, exact Newton-type methods that use on the exact Hessian (reflecting a description of the curvature of the function) are not typically suitable. GN and LM methods are expected to perform well with zero-residual problems, however when solving large-residual problems these methods can perform poorly and may not be suitable (Dennis Jr and Schnabel, 1996; Mohammad and Waziri, 2019). Quasi-Newton methods are a class of methods similar to the full Newton method but instead these approximate the Hessian, with approximations generally improving at each step. The reader is directed to chapter 6, section 6.2.4, which deals with the basic theory behind these algorithms with regard to derivatives and direction of the search process. This is one of the few times in the thesis the reader is directed forward, however, this avoids the need for excessive repetition of basic theory within the thesis. Quasi-Newton methods are typically computationally cheap, for example the Broydon-Fletcher-Goldfarb-Shannon (BFGS) algorithm (Byrd *et al.*, 1995a) requires $\mathcal{O}(n^2)$ operations per iteration, compared to the full Newton method that requires $\mathcal{O}(n^3)$ (Henao, 2014). Quasi-Newton methods have represented a popular choice for NLS optimisation in data fitting problems and are often available across a multiple of programming languages and mathematical suites (e.g., Mathematica, MATLAB, GNU Octave, R, SciPy). In particular, the BFGS algorithm is a standard tool for the optimisation of smooth functions (Wright and Nocedal, 1999) and includes an exact or inexact line search method to determine step size (Henao, 2014). The algorithm used in this experiment to solve the least-squares problem at each iteration was an implementation of the limited memory modification of the BFGS method (L-BFGS) in R, with the inclusion of a further modification to incorporate box constraints (L-BFGS-B). The limited memory modification variant of the BFGS method uses less computer memory to update the approximation to the inverse of the Hessian, by only storing a record of the last m iterations rather than an $n \times n$ matrix where m is a small number and n is the number of parameters (Henao, 2014). As such, L-BFGS only requires $\mathcal{O}(mn)$ operations per iteration so is well suited to problems where the number of free parameters n is large. The inclusion of bounding however increases the cost of the line search slightly due to extra necessary steps to ensure the algorithm remains in the defined box with each step (Henao, 2014). The algorithm is available as part of the *optim* function included as part of the *stats* library included in the standard R environment (R Core Team, 2020). Analytic gradients were not supplied to the function, for reasons described above, and therefore the algorithm attempts to approximate the gradient using finite differencing, and this increases the possibility that in some instances abnormal (unsuccessful) termination in the line search may occur. Algorithm convergence is reported by the *optim* function as part of the convergence code and message returned following the search (Henao, 2014). Although supplying precise analytic gradient functions may improve the success

of the algorithm, this is a challenging to impossible task and unrealistic step for a sport science researcher or practitioner when fitting FFMs Readers are referred to the works of Henao (2014) and Wright and Nocedal (1999) for more in-depth analysis of the behaviour of this algorithm. The parallelised searches were run on an 8-Core Intel® Xeon® Gold 6230 CPU @ 2.10GHz, with 8.0GB available RAM (80% average usage). Note, successful termination (convergence) refers to achieving an iterative reduction of the objective function that is within a factor (1e7) of the machine tolerance (2.2e-16), giving an approximate tolerance of 2e-9.

5.3.6 Analyses

Given that the procedure was evaluated on a large deterministic grid, it is sufficient to treat the results as a "complete population" given that no stochastic element was introduced and therefore there is no intention to perform inference about a superpopulation. Findings were best communicated by descriptive statistics and visualisations to summarise spread, shape, and centrality of fitted parameter estimates, prediction errors, and the rate of convergence to the true parameters (and other local optima) in each scenario. Local optima were appraised based on the definiteness of the Hessian matrix. For solutions where the associated Hessian was positive semi-definite and objective value (RSS) not equivalent to the known global extremum (i.e., 0 at the true parameters), the critical point was indicated to be a local minimum. Similarly, if the Hessian at a given solution was indefinite this indicated the point was a saddle. Appendix D provides further distributional summary tables of solutions at a resolution of each parameter. An online repository [6] is also available which contains the raw results data, analysis, and the code associated with the experimental implementation.

5.4 Results

5.4.1 Parameter estimates (convergence)

Within each scenario (i.e., model [standard, fitness-delay] × proportion of data [100%, ~50%, ~33%]), the L-BFGS-B algorithm terminated successfully during 99.71-99.97% of the iterations. The estimates from searches that terminated successfully are referred to as "solutions", although this term does not imply whether the estimates reached the true parameters. For all models, a reduction in the amount of data did not appear to influence the number of solutions that reached the true parameters (i.e., the number of fitting iterations that recovered the true global minimum). Within the standard model scenarios, 69.1-70.3% of solutions found were the true parameters. In contrast, within the fitness-delay model scenarios, 17.6-17.9% of solutions found were the true parameters. Within the standard model

[6] Associated experiment source code, analysis code, and results data can be found at: **github.com/bsh2/thesis/c5**

scenarios, the remaining non-true solutions resolved to other critical points including saddles (27.6-28.8%) and a small number of local minima (2-2.1%). Within the fitness-delay model scenarios, the remaining non-true solutions resolved to predominantly local minima (76.1-78.3%) and a small number of saddle points (4.0-5.9%). Table 5.3 provides comparison between the scenarios with respect to the results outlined so far. The parameter distributions of the solutions that did not find the true parameters were similar between the three standard model scenarios (i.e., 100%, 50%, 33% of fitting data) (see Figure 5.6). This was also the case for the fitness-delay model scenarios (figure 5.6). Tables of summary statistics describing the fitted parameter estimate distributions are provided in Appendix D parts D-1 and D-2. Tables of the highest frequency (non-true) solutions found for each scenario (model × data proportion) are given in Appendix D parts D-3 and D-4.

Table 5.3: Convergence rates of the solutions found by the L-BFGS-B algorithm (to critical points in the parameter space)

Scenario			**Totals**			**Convergence rates (Critical points)**		
Model	**Data (%)**	**Data points**	**Iterations (total sets)**	**Successful termination**	**Abnormal termination**	**True parameters**	**Other local minima**	**Saddle points**
Standard	**100 %**	147	10^5	99967 (99.97%)	33 (0.03%)	69204 (69.2%)	2047 (2.1%)	28716 (28.7%)
Standard	**50 %**	74	10^5	99968 (99.97%)	32 (0.03%)	69145 (69.1%)	1995 (2.0%)	28828 (28.8%)
Standard	**33 %**	49	10^5	99960 (99.96%)	40 (0.04%)	70284 (70.3%)	2056 (2.1%)	27620 (27.6%)
Fitness-delay	**100 %**	147	7^6	117305 (99.71%)	344 (0.29%)	20588 (17.6%)	91909 (78.1%)	4808 (4.1%)
Fitness-delay	**50 %**	74	7^6	117492 (99.87%)	157 (0.13%)	20651 (17.6%)	92127 (78.3%)	4714 (4.0%)
Fitness-delay	**33 %**	49	7^6	117551 (99.92%)	98 (0.08%)	21065 (17.9%)	89518 (76.1%)	6968 (5.9%)

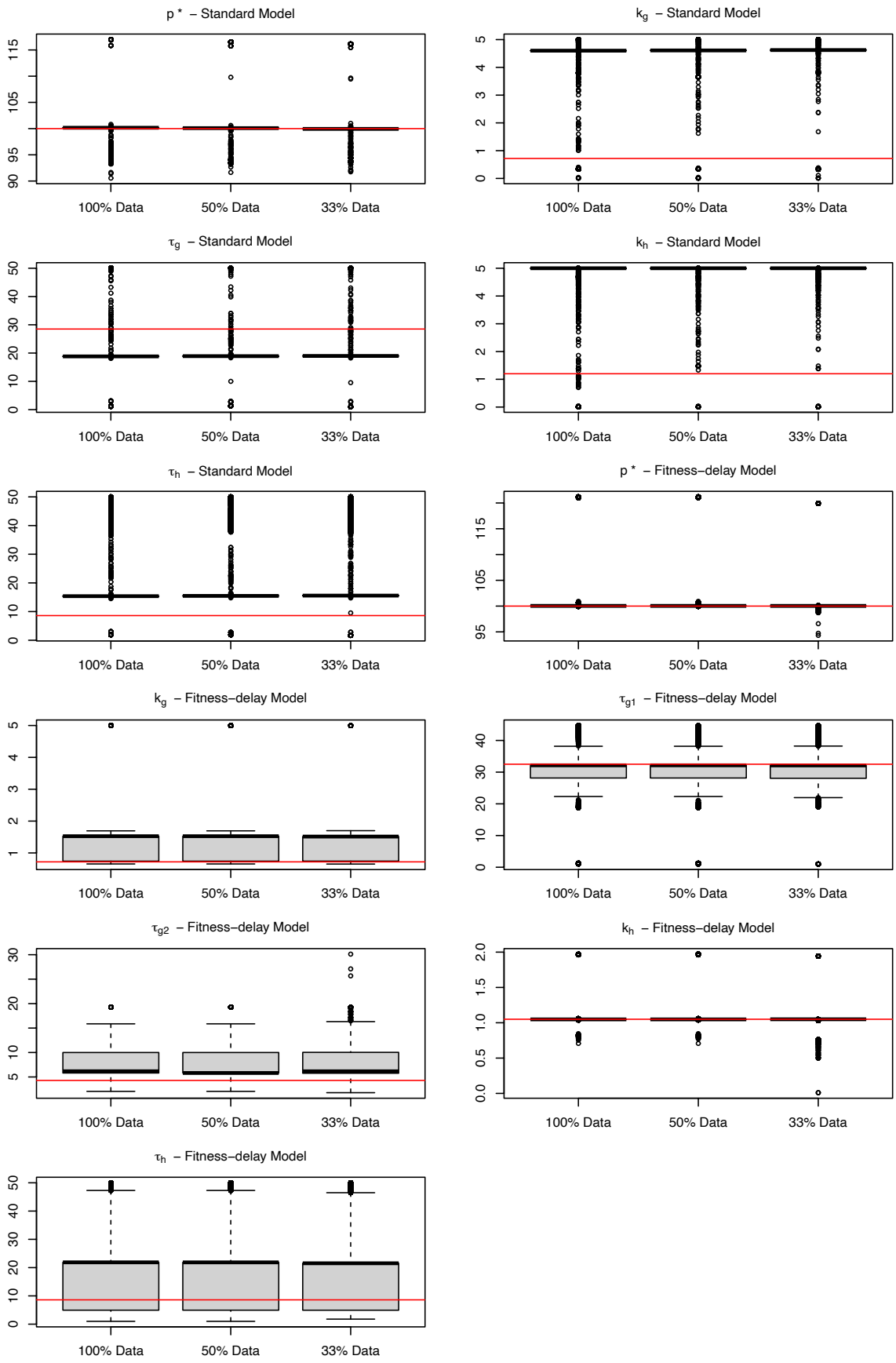


Figure 5.6: Parameter estimate distributions from the solutions that did not reach the true values (i.e., global minimum), for the standard and fitness-delay models. The red line indicates the true value.

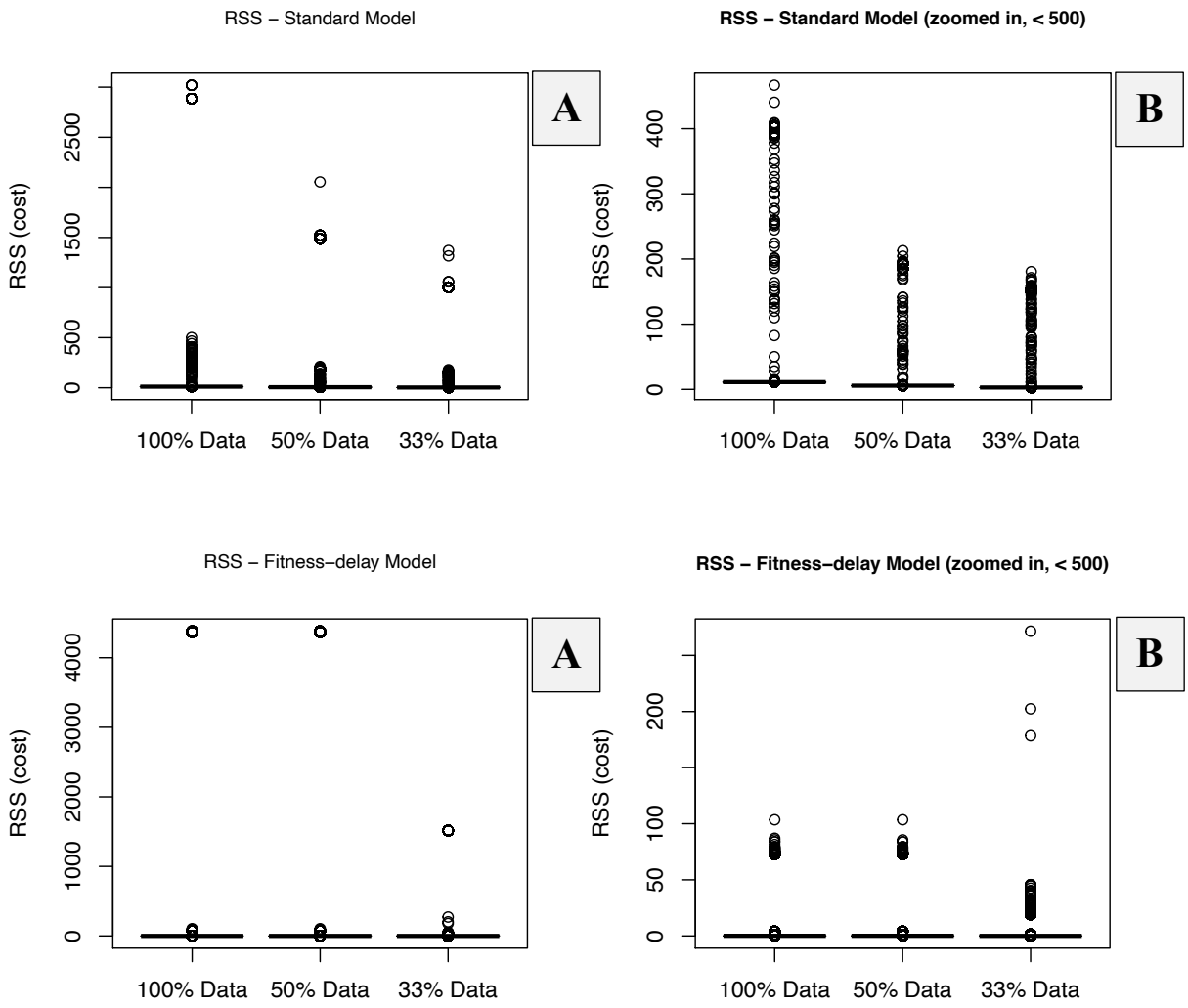


Figure 5.7: Objective function values (RSS) associated with solutions that did not reach the true values (i.e., global minimum), for the standard and fitness-delay models (A). The (B) plots offer a 'zoomed in' picture of the distribution following removal of large values (outliers, RSS < 500).

5.4.2 Prediction errors (model fit)

In-sample median model fit across solutions that did not reach the true parameters were similar and strong in all scenarios and metrics including root-mean-squared error (RMSE) and mean average percentage error (MAPE). For example, $RMSE_{(median)}$ for the standard FFM ranged from 0.24-0.28 and $RMSE_{(median)}$ for the fitness-delay model was 0.03-0.06 across all proportions of fitting data. Median absolute deviation $RMSE_{(m.a.d)}$ was ~0 for the standard model and 0.01-0.03 for the fitness-delay model searches. Although there were a small number of solutions in each scenario that resolved to poor model fit (i.e., RMSE = 5.27 and correspondingly high RSS values) (seen visually as the outliers in the fitted model traces plotted in Figure 5.8 and 5.9). Figures 5.8 and 5.9 also illustrate visually that the range of daily prediction errors (performance profiles) across fitted parameter sets that did not reach the true

parameter values was low, and figures 5.10 and 5.11 provide the associated distributional plots for the errors. In each scenario, parameter sets returned by the algorithm from searches that resulted in abnormal termination also resolved to good model fit (RMSE = 0.01 - 1.77, MAPE = 0.01 - 1.30%, across all scenarios for both models) (Table 5.4) despite differences in the parameter values in these sets relative to the global minimum point (Tables D-1, D-2, appendix D).

Table 5.4: Model fit (in-sample) summary statistics for the fitted solutions

		Descriptive statistic	Standard model		Fitness-delay model	
Data	**Converged**		**RMSE**	**MAPE**	**RMSE**	**MAPE**
100%	**Non-true solutions**	**minimum**	0.28	0.21	0.01	0.01
		maximum	4.53	3.18	5.46	3.87
		median	0.28	0.22	0.03	0.02
		m.a.d	0.00	0.00	0.02	0.02
100%	**Abnormal termination**	**minimum**	0.28	0.22	0.01	0.01
		maximum	0.28	0.22	0.06	0.05
		median	0.28	0.22	0.06	0.04
		m.a.d	0.00	0.00	0.01	0.01
50%	**Non-true solutions**	**minimum**	0.28	0.21	0.00	0.00
		maximum	5.27	3.98	5.46	3.87
		median	0.28	0.21	0.03	0.02
		m.a.d	0.00	0.00	0.03	0.02
50%	**Abnormal termination**	**minimum**	0.28	0.21	0.01	0.01
		maximum	0.28	0.21	0.71	0.54
		median	0.28	0.21	0.05	0.04
		m.a.d	0.00	0.00	0.02	0.02
33%	**Non-true solutions**	**minimum**	0.24	0.19	0.01	0.01
		maximum	5.29	4.06	5.56	3.94
		median	0.24	0.19	0.03	0.02
		m.a.d	0.00	0.00	0.03	0.02
33%	**Abnormal termination**	**minimum**	0.24	0.19	0.01	0.01
		maximum	1.77	1.30	0.07	0.05
		median	0.24	0.19	0.06	0.04
		m.a.d	0.00	0.00	0.01	0.01

m.a.d : median absolute deviation

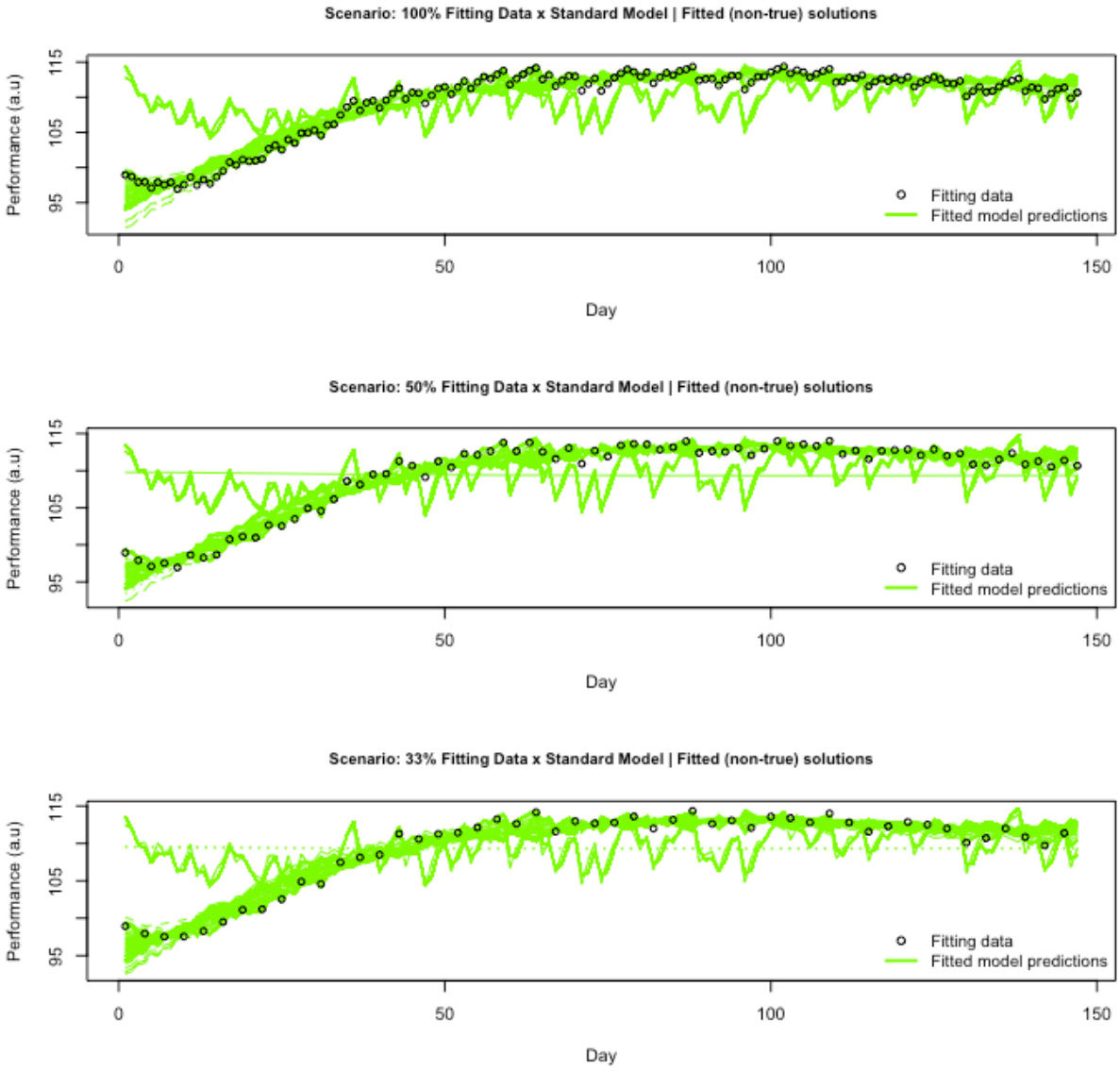


Figure 5.8: Fitted model predictions (in-sample) reflecting the range of performance profiles generated by the non-true solutions – Standard model scenarios (green)

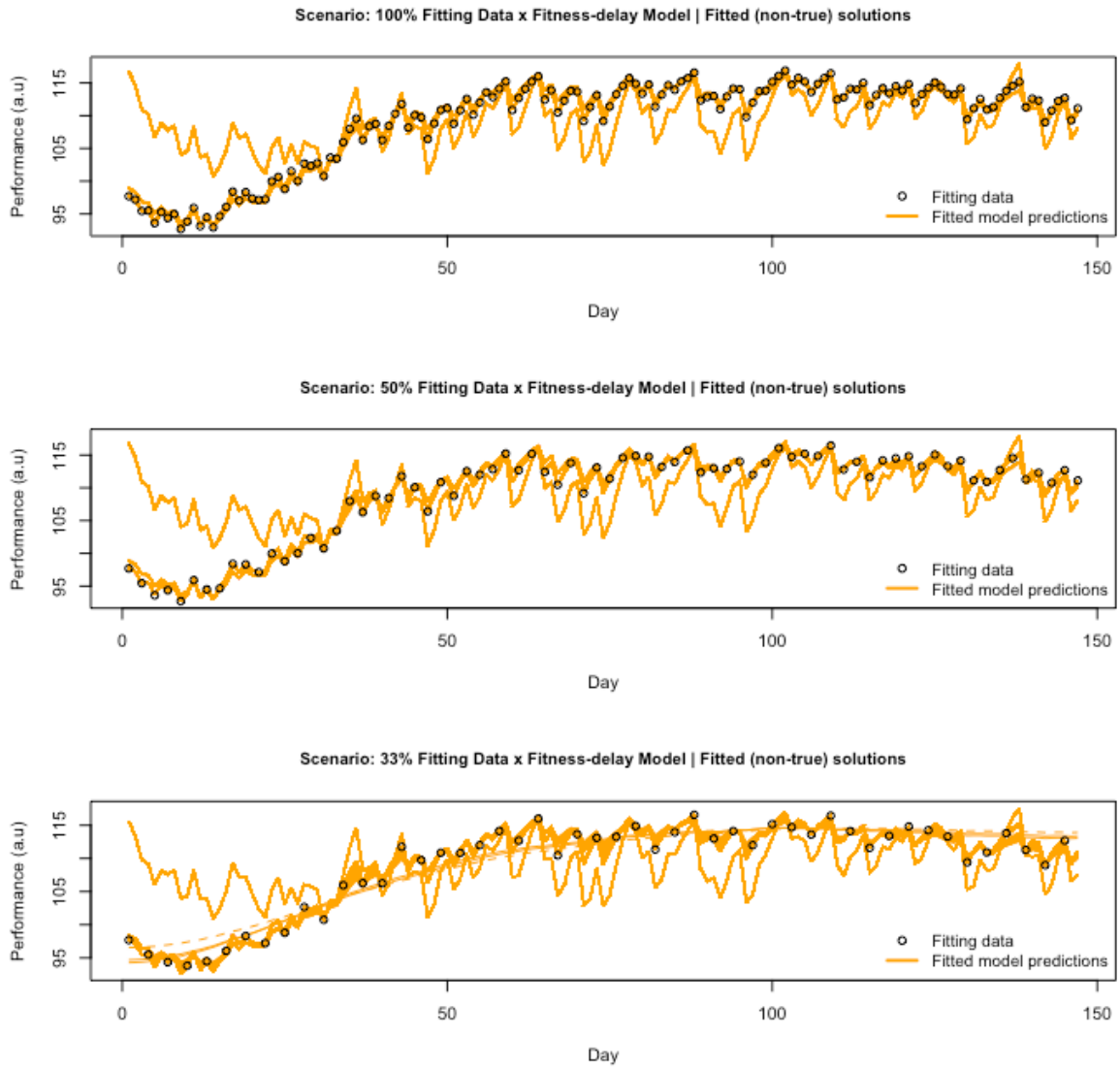
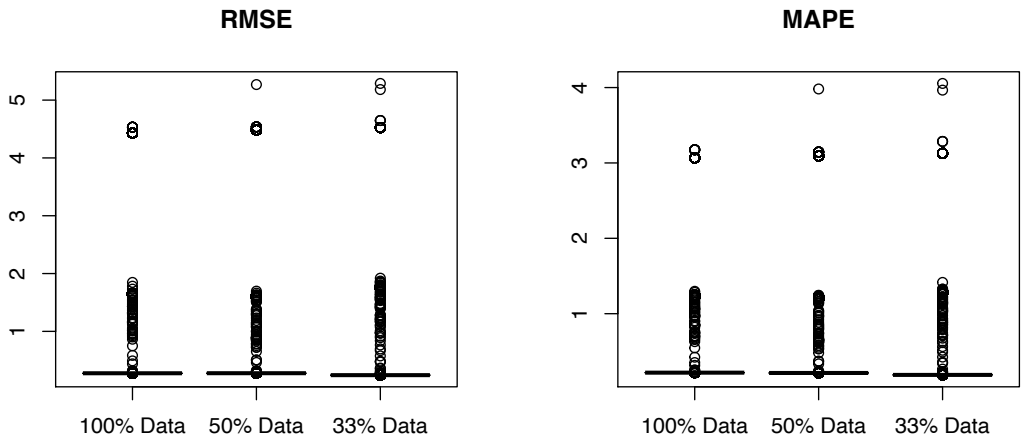
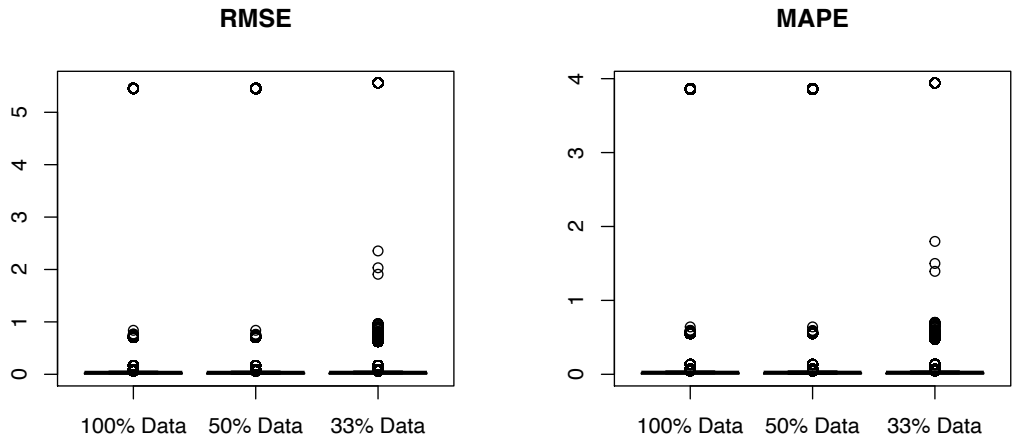


Figure 5.9: Fitted model predictions (in-sample) reflecting the range of performance profiles generated by the non-true solutions – Fitness-delay model scenarios (blue)



Standard Model

Figure 5.10: Comparison of in-sample goodness-of-fit (RMSE, MAPE) for non-true solutions, obtained for the standard model searches across the three proportions of fitting data



Fitness-delay model

Figure 5.11: Comparison of in-sample goodness-of-fit (RMSE, MAPE) for non-true solutions, obtained for the fitness-delay model searches across the three proportions of fitting data trialled.

5.4.3 Runtime

There were large differences in the fitting time between the two models, and within each model when fitting to additional data. An increase in data for both models (%) resulted in an approximately directly proportional increase (%) in fitting time per search. The fitness-delay model was also substantially slower to fit per search (approximately 2.8-2.96 times slower) in comparison to the standard model, likely due to the presence of the additional free parameter. These results are summarised in table 5.5 at the top of the following page.

Table 5.5: Fitting runtime across all scenarios for L-BFGS-B algorithm

Model	Data points	Iterations (N)	Runtime (Total)	Mean runtime (per search)
Standard	100% (N=147)	100,000	74 Hours	2.7 Seconds
Standard	50% (N=74)	100,000	38 Hours	1.4 Seconds
Standard	33% (N=49)	100,000	28 Hours	1.0 Seconds
Fitness-delay	100% (N=147)	117,649	263 Hours	8.0 Seconds
Fitness-delay	50% (N=74)	117,649	133 Hours	4.1 Seconds
Fitness-delay	33% (N=49)	117,649	90 Hours	2.8 Seconds

Runtime (total) given to the approximate hour.

5.5 Discussion

This study investigated the typical model fitting process for two common fitness-fatigue models under a well-known quasi-Newton algorithm (BFGS), with limited memory modification and bounding (L-BFGS-B). To enable direct study of the effectiveness of the optimisation algorithm, each model was assumed to fully specify the training response and simulated under pre-selected true parameters and synthetic load inputs to derive performance data without additional error. It was argued that for this common algorithm to be robust for FFM fitting problems, it should as a basic capability be able to find the true minimum (zero residual) known (only) to exist in this simulation framework. It was known prior to the study that the convexity of the objective function over its domain would influence the effectiveness of a local optimisation algorithm such as BFGS, but the existence of local optima had previously never been acknowledged in prior work and ignored as a possible issue with reported solutions.

When initiated from a wide array of initial points spanning the parameter space in a uniform fashion, the L-BFGS-B algorithm was successful at finding the true solution in approximately 69-70% of the searches in each of the standard model scenarios but was only successful in between 15-18% of searches in the fitness-delay model scenarios (Table 5.3). These results demonstrate concerns with the algorithms ability to obtain suitable fitted estimates for both models even under the idealistic condition of no model error. These concerns were further exacerbated for the fitness-delay model, where the optimiser was only able to find the true values in less than 18% of the searches, with a high frequency of local optima across the search space demonstrated. It is likely that a substantial reduction in the rate of successful convergence to the true solution in this experiment in the fitness-delay model scenarios is due to the added complexity in the search space as a result of the additional fitness parameter (τ_{g_2}) and its relationships with other parameters in the model creating a higher incidence of local optima. There were no discernible patterns in the distribution of initial estimates with respect to starting error (RSS) for the standard (appendix D-3, figure D-3.1, 3.2) and fitness-delay model scenarios (appendix D-4, figure D-4.1, 4.2).

Collectively, these results indicate that the typical NLS fitting approach is a harder optimisation problem for a typical hill-climbing algorithm to solve than previously recognised in the literature, and in particular when fitting the fitness-delay model. However, these results would benefit from further confirmatory experimentation under different input distributions to determine average rates across varying inputs, and alternative standard algorithms (first and second-order methods). It is likely however, that more advanced global optimisation algorithms such as differential evolution, and genetic

algorithms, may be required to adequately solve the NLS data fitting problem within fitness-fatigue modelling.

A reduction in the amount of data used to fit the models did not appear to affect the algorithms convergence rates to the true solution in this experiment, or the distributions of the non-true solutions (parameter values) and associated model fit (Table 5.3, Figure 5.6). As would be expected, a reduction in fitting data did improve fitting time. In isolation, this reduction in fitting time is of little practical interest, as fitting an individual model to data via the method used in this experiment is relatively short (i.e., seconds), and researchers would always be expected to maximise available data even at the expense of runtime. However, fitting time may become important when incorporating robust observational or tuning-based cross-validation approaches to evaluate model validity, and/or when using complex optimisation algorithms (e.g., genetic algorithms). In these instances, researchers are advised to allocate reasonable compute resources and sufficient time to the fitting process and consider the use of parallelisation. Parallelisation in the context of cross-validation is examined at a practical level in chapter 6 (section 6.2.5), with extensive code provided in R.

Many unique (non-true) solutions (to 1.d.p $\{p^*, \tau_g, \tau_h\}$ and 2.d.p $\{k_g, k_h\}$) were found in each of the scenarios involving the standard model (range = 275-353 per scenario) (Appendix D-3). The set of unique solutions collectively spanned most of the search space in all parameters, and the most were saddle points. The frequency of each unique solution was highly variable (for example, a given unique solution appeared between 1 and 25838 times in one standard model scenario). Appendix D-3 demonstrates the top 10 most frequent solutions across each standard model scenario, and supplementary file 1 (SF-1)[7] contains the entire set of unique solutions for the standard model and fitness-delay model scenarios. In contrast to the standard model scenarios, most of the unique solutions found in the fitness-delay model scenarios (range = 383-550 per scenario) (Appendix D-4) were local minima, rather than saddle points. The frequency at which a given unique solution appeared also demonstrated high variation (for example between 1 and 39401 times in one fitness-delay model scenario). Collectively, these results demonstrate that there appear to be many points in the search space at which the algorithm can become stuck, but also that local solutions can be found across the search space.

The implications of these results on prior and future research are that unless efforts were (or are) made to improve the likelihood that a solution found is the absolute minimiser of the NLS problem, then subsequent results relating to prediction accuracy under solutions carried forward may not be

[7] SF-1 can be found at the following repository link:
github.com/bsh2/thesis/c5/analysis/SF-1.xlsx

interpretable or robust generalisations of model error. Although, quite obviously, there is never a known comparator solution (as is the case in a simulation approach) within real world experiments due to model misspecification and noise, an appropriate recommendation arising from the results of this experiment is the re-running of first and second-order algorithms that require starting points from a large grid of stochastically generated points covering most of the parameter space between the bounds. The second recommendation arising from this study is further scientific investigation of global optimisation methods, such as evolutionary strategies for fitting FFMs via NLS, and in particular the prospective use of global methods with integrated random local search (via first and second-order methods). Readers are advised to see chapter 6, section 6.2.3, listing 6.25 for implementation of a genetic algorithm with random local search in R.

Another notable result of this study was that strong in-sample model fit was observed across most solutions that did not converge to the true solutions for each model (and the associated scenarios) (Table 5.4, Figures 5.8-5.11). This finding would benefit from further confirmatory experimentation under different input distributions. However, it is highly conceivable that in prior applications of FFMs, researchers may have ignored the possibility that solution obtained is not the absolute minimiser, specifically when measures of in-sample model fit are particularly strong (e.g., RMSE, MAPE), and where no estimate of fitted estimate uncertainty or starting point sensitivity has been determined. Therefore, the findings in this experiment add weight to the hypothesis that there exists substantial doubt in reported estimates across prior research; particularly where optimisation procedures have not been stated clearly or have lacked the relevant procedural detail to indicate that these issues have been considered or addressed. The negative implications of this are primarily placed on the interpretation of prior model validity work, and subsequent decisions made by researchers with regard to the collective optimism (or lack thereof) toward further study of FFMs.

The main limitation of this study is that there exists a possibility that results may differ under different input (training load) distributions, and that the exact assumptions required to enable study of the optimisation process via computation are too unrealistic in the real world, due to extensive model misspecification. Specifically, fitting to measured performance data is unlikely to ever be zero residual optimisation problem due to the presence of noise and inherent simplification within the modelling process resulting in model misspecification. Further, it is unclear the role that model misspecification will play in these results, and it is possible that algorithm performance may be considerably worse such that changes in local minima have substantive differences in predictions and model fit. However, it seems unlikely they would be better. It appears clear that further work in the application of optimisation approaches for FFM problems is required, and that alternative algorithmic approaches (e.g., evolutionary or genetic algorithms) should be evaluated, and different perspectives (e.g., the use of priors under a Bayesian approach) represent sensible pathways for future work.

5.6 Summary, conclusions, and practical recommendations

Collectively, this experiment highlights that significant care must be taken in future research and practice to ensure that the optimisation problem is appropriately posed, and that the algorithmic approach selected to fit the selected FFM is sufficient, due to the high likelihood of local optima. In particular, solutions may not be all that they appear following one-shot minimisation using a hill-climbing algorithm such as L-BFGS-B, even in the presence of very good in-sample fit. At a minimum, multiple runs of optimisation should be performed under this approach, starting from many points spanning the breadth of the search space (sensitivity), and include some form of observational cross-validation as discussed to estimate uncertainty. Notably, both fitness-fatigue models demonstrated that different non-true solutions may exhibit the same model behaviour and achieve strong model fit. This creates a series of challenges for researchers in obtaining solutions in real-world experiments via similar approaches, as absolute minimisation cannot be confirmed, and uncertainty only estimated. It also casts doubt across correctness of solutions reported in prior FFM literature. This experiment has highlighted that the search space of the standard model, and in particular the fitness-delay model are more complex and challenging for standard algorithms than previously recognised; and it is likely that we require better algorithmic approaches to solve FFM data fitting problems. Bayesian methods and evolutionary algorithms may offer two possible routes toward improved fitting of FFMs. However, the role of cross-validation (out-of-sample testing) in the model evaluation process can also not be ignored going forward. Out-of-sample testing may also offer a qualitative approach for flagging solutions that are clearly incorrect, or it may also be used in a more modern sense within tuning-based cross-validation frameworks. This work, although extensive, is an *n*-of-1 with respect to the input distribution, and so would benefit from further replication under different training load distributions and performance profiles.

Chapter 6: Pathfinding: Software development and consideration of prospective approaches for future research

6.1 Introduction

The research conducted in chapter 4 has been valuable in identifying key practical issues at the heart of the application of fitness-fatigue modelling, and in particular experimental factors that may affect model accuracy, such as the importance of high measurement frequency and recording performance data with sufficient accuracy (i.e., reduction of measurement error and biological variability). The study in chapter 5 has also highlighted issues previously not studied in the selection of algorithms typically used to fit FFMs. In particular, chapter 5 demonstrated the existence of many local optima that can provoke starting point sensitivity within the fitting process for common first and second order algorithms. These results indicated that further research must be carried out in parameter estimation and highlighted that identification of more effective algorithms to fit FFMs is now required. In particular, evolutionary strategies or hybrid approaches represent reasonable next steps due to their global optimisation properties (Rozendaal, 2017; Connor and O'Neill, 2020). This chapter draws together and, in some sections, expands upon key content from the literature review (e.g., discussion of FFMs, and theory of associated methods for model estimation and evaluation), alongside the development of a set of extensive code resources for fitting and evaluating FFMs. This chapter addresses aims 3 and 4 of this thesis, to provide researchers interested in fitness-fatigue modelling with a comprehensive set of tools to guide practice and promising avenues of future research. Specifically, code was developed in the programming language R and presented in a progressive manner, beginning with tools to fit common FFMs (e.g., standard, fitness-delay, VDR) via nonlinear least-squares and maximum-likelihood estimation approaches, before further treatment of aspects of implementation including employing non-linear ODE systems, optimisation theory and algorithms, model evaluation (cross-validation), and incorporating model uncertainty with respect to advanced methods (e.g., Kalman-filtering). This chapter is set out in in a manner that requires the reader to engage with code presented in each section, to maximise the acquisition of skills and understanding relating to key aspects of model implementation. The approach used here contrasts with the provision of a polished software suite or packaged code, where underlying mechanisms are mostly hidden, and the user is separated from the internals of the program. Instead, it was deemed appropriate in the context of research, and particularly fitness-fatigue modelling, for researchers to be aware of the workings of tools they are using. This also offers a substantive amount of flexibility for the code to be adapted by the reader for their specific use case.

6.1.1 Improving the availability of resources for research

The effective development of data driven modelling approaches to assist with individualised prescription will require greater numbers of researchers to investigate more advanced models than the standard FFM and adopt methods that reflect the need for robust evaluation of model validity (e.g., out-of-sample testing). However, it appears that a lack of uptake of even basic FFMs in research is constrained by very limited resources to assist with learning, fitting, and even basic computation of existing models. Historically, resources are limited to the supplementary spreadsheet in Clarke and Skiba (2013), with this spreadsheet representing one of the only deliberately placed practical resources available across the FFM literature. At the time, the spreadsheet was an educational resource rather than a realistic tool for contemporary research, particularly given that it is limited to the basic model and the built-in solvers available in Microsoft ® Excel. Additionally, no facilities for analysis of the modelling process were provided beyond in-sample metrics of model fit. Nevertheless, since its development in 2013 this resource has been useful as an educational device for sport science researchers interested in this area. However, almost ten years later, and given the current offering of several more appropriate FFMs and methods in the literature, flexible and more powerful resources are required that better serve the demands of future work. Increased uptake of powerful programming languages such as R and Python within sport science also creates an opportunity to extend the capabilities of any resources offered to researchers, beyond what is typically achievable within normal spreadsheet packages. Screenshots of key sheets within the Clarke and Skiba (2013) article are highlighted below for the purposes of this chapter remaining self-contained, but readers may obtain the spreadsheet [8] themselves.

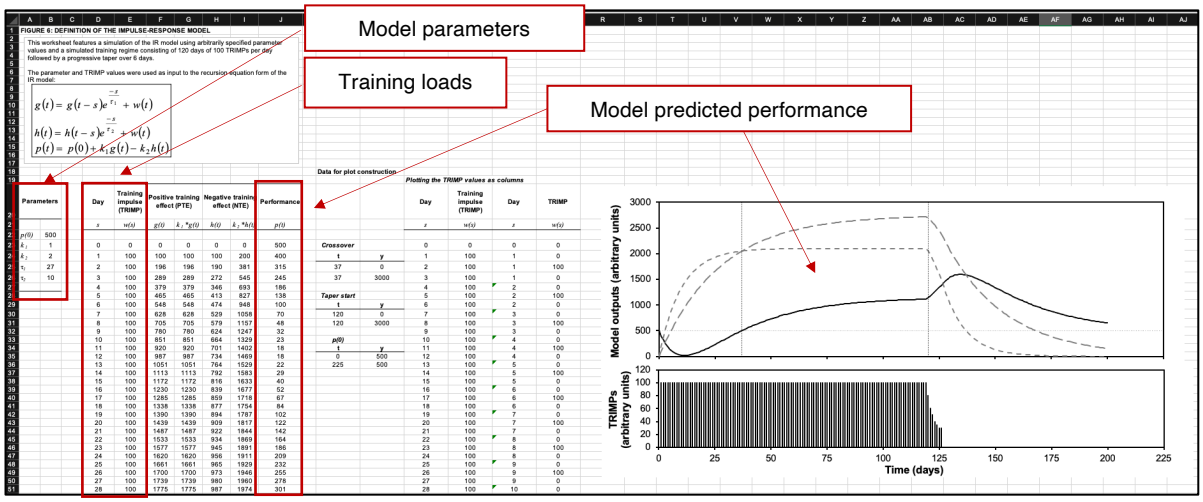


Figure 6.1: Screenshot from the "IR model def" sheet within the Clarke and Skiba (2013) supplementary spreadsheet. Basic FFM simulation for a specified set of parameter values with plots.

[8] https://doi.org/10.1152/advan.00078.2011

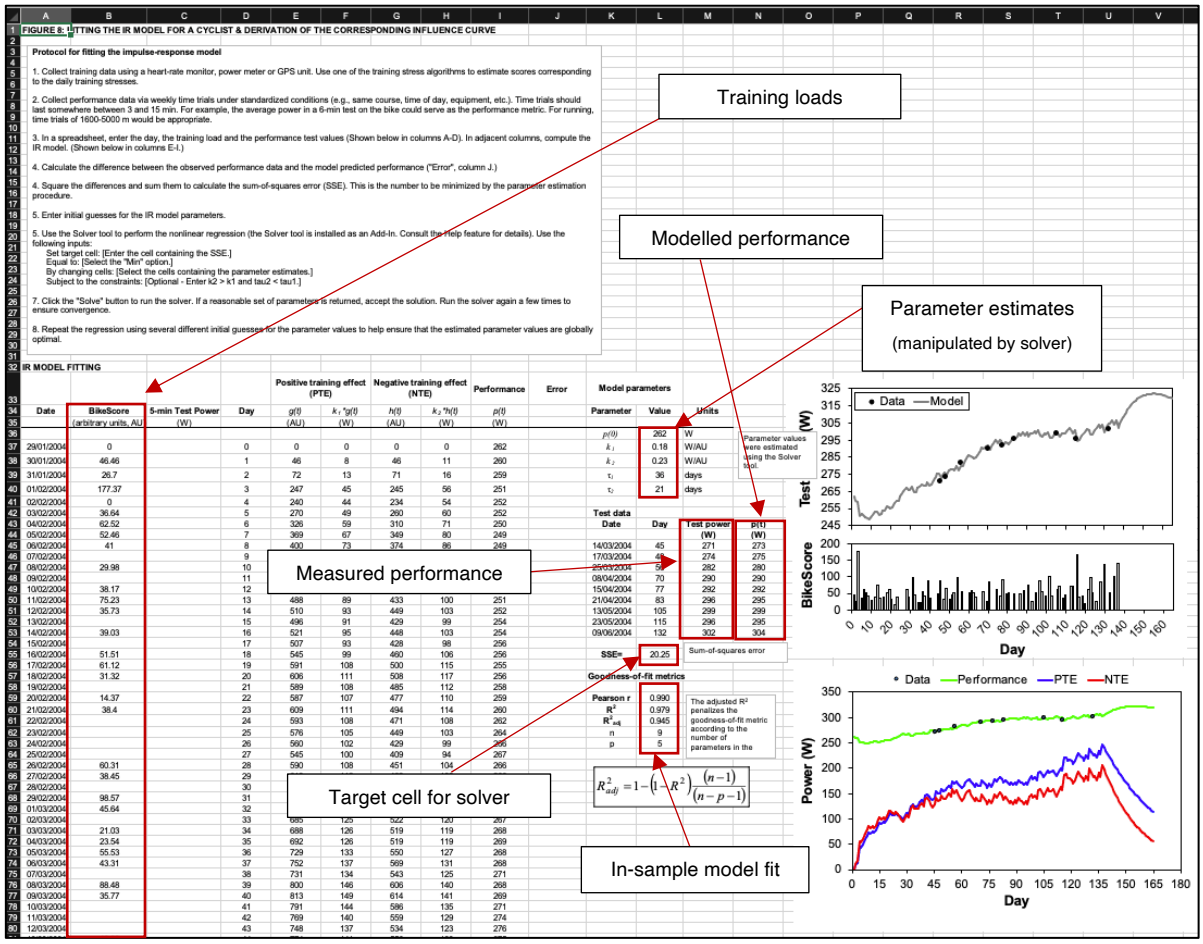


Figure 6.2: Screenshot from the "IR model fit" sheet (1 of 2) within the Clarke and Skiba (2013) supplementary file. The sheet facilitates input of a user's training data and performance observations, and RSS can then be minimised by a generic solver tool which performs nonlinear regression from an initial guess at the parameters. Fitted output is plotted. Limitations include a lack of out-of-sample assessment and no flexibility in the fitting process.

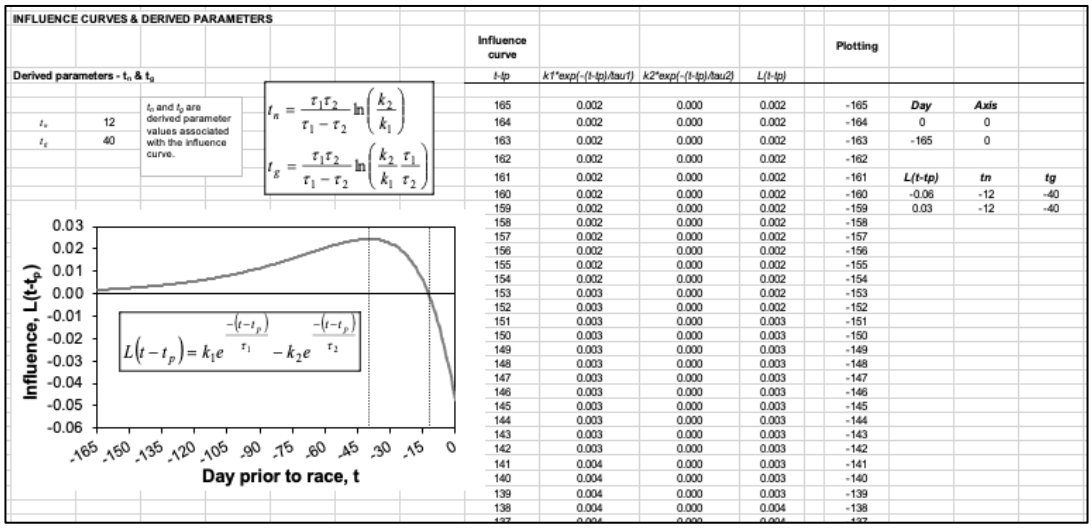


Figure 6.3: Screenshot from the "IR model fit" sheet (2 of 2) within the Clarke and Skiba (2013) supplementary file. Demonstrating influence curve calculation and plotting from fitted parameters.

Toward addressing the current gap in available resources, and in relation to the aims of this research project, an open-source code repository of R tools for fitness-fatigue modelling was created in collaboration with another researcher (Ben Ogorek) and publicised via a dedicated website (fitnessfatigue.com). The codebase is under active development, and researchers new or well known to performance modelling are encouraged to engage with the software via comments, suggestions, and/or contribution of their own code. Collaboration is facilitated via the GitHub ® platform, where development of the codebase also took place. GitHub ® offered a suitable platform for hosting and developing the codebase due to integrated version control and source code management (SCM) functionality provided by git (git-scm.com). Additionally, tools such as feature requests and bug tracking improve the efficiency of the development process. Increased utilisation of these types of platforms in future sport science research involving code and computational designs may help to support open-science principles including open-access, open-source, and open-data (Peng, 2011). Some of the resources contained within the repository were also developed for use as supplementary material to the two-part review series (Stephens Hemingway *et al.*, 2021; Swinton *et al.*, 2021). This chapter focusses on illuminating the development process of models and functionality included in the codebase, such as cross-validation, Kalman-filtering, and optimisation via maximum likelihood estimation. The work in this chapter can be viewed as the documentation and an educational resource aligning with this repository, examining the development of code and associated theory, rather than just demonstrating how the functions are used. Toward this end, the code relating to and presented in this chapter (and specifically each subsection) is segregated from the open-source project in a separate repository, found at **github.com/bsh2/thesis/c6**). Pointers are also given throughout the chapter to relevant files within this separate repository. As previously identified, the aims of this chapter address aims 3 and 4 of this PhD (Table 3.2, chapter 3), by providing resources and skills to support future applications of FFM research, education, and highlighting promising approaches to address existing problems with the standard model.

6.1.2 Structure of the chapter

The chapter is split into 5 sections (6.1 – 6.5) that build progressively on concepts presented and accompanying code. Section 6.2 forms the largest body of work in the chapter, introducing the reader to working with FFMs in R, including discussion of the use of concepts such as vectorisation, loops, and the apply family of base R functions. Functions are developed in subsections 6.2.1 and 6.2.2 for simulation or use as objective functions (sum-of-squares or likelihood) within parameter estimation approaches (nonlinear least-squares or maximum likelihood), for the standard, fitness-delay, and VDR FFMs. Two contrasting approaches: 1) explicit for-loops (6.2.1); and 2) the apply family of base R functions (6.2.2); are presented for developing these functions (simulation, objective). Self-contained examples are then provided at the end of 6.2.1 and 6.2.2 for fitting the models to synthetic data, via

nonlinear least-squares and maximum likelihood estimation, respectively. In section 6.2.3, code is presented for numerically solving the original linear system of ODE's (Banister *et al.*, 1975) and fitting the model parameters. This approach is then adapted to enable the fitting approach required for the nonlinear ODE system proposed by Turner *et al.* (2017) to capture the effects of overtraining in model behaviour. In subsection 6.2.4, the theory behind several common optimisation algorithms is examined from gradient descent to evolutionary strategies, and the availability of several algorithms already available in R discussed. Minimum recommendations are discussed for fitting FFMs via first and second order (local) methods. The last subsection, 6.2.5, of section 6.2, develops a cross-validation approach for fitting and testing the VDR model, that is theoretically adaptable for any FFM or optimisation approach. In section 6.3, the threshold saturation function (Hellard *et al.*, 2005) is briefly revisited, and in section 6.4 a from-scratch implementation of a Kalman filter in R is demonstrated under a state-space reformulation of the standard model, as first presented in the literature by Kolossa *et al.* (2017). Finally, section 6.5 presents a brief summary of the chapter.

Although extensive in text, this chapter attempts to present a large volume of information both essential and detailed to fit with the diverse backgrounds and expertise of readers that may use this material. Code is presented inline alongside text and figures, as opposed to in appendices of supplementary file to provide a better pedagogical approach for maximising practical understanding, anecdotally supported by the structure of the highly popular Q&A site for programmers, Stack Overflow (stackoverflow.com). Although 'code heavy' accompanying explanation attempts to concisely describe concepts where annotation alone is not sufficient understanding (such as when applying certain control flows). However, in many cases the reader is expected to be able to follow the code and annotation to grasp essential elements within a given implementation.

6.1.3 The R environment

The R programming environment (R Core Team, 2020) was viewed as the most appropriate language for developing the codebase and working with FFMs, due to perceived increased popularity amongst sport scientists seeking more powerful alternatives to spreadsheet packages and limited data visualisation tools. The primary benefit (and intrinsic challenge) of using R is that it is highly expressive, and therefore flexible and robust solutions to various modelling and statistics problems can be constructed from first principles or via dedicated packages. In turn, the programmer is likely to become more involved and interested in aspects of model implementation than they may otherwise have been if using point-and-click software. Writing code necessary to fit an FFM requires understanding of the optimisation problem formulation, and associated methods for solving it. This can naturally reveal gaps or raise questions in a researcher's understanding, that otherwise might have been missed working in simpler hands-off environments (e.g., spreadsheet software). In addition,

advanced methods of cross-validation, automated multi-start optimisation, and parallelisation are not realistic nor sensible tasks to attempt in a spreadsheet package when more efficient and suitable alternatives exist.

Crucially, this chapter does not represent a first introduction to the language R and shouldn't be treated as such. There is an expectation that the reader already has some familiarity with the language, including the standard syntax, object types and data manipulation, and basic programming structures. Several excellent resources currently exist providing sound introductions to R to make the most of this chapter. For example, the *swirl* package (swirlstats.com) is an interactive learning environment comprising 4 courses and 15 modules reflecting the fundamentals of the R environment. Concepts covered in *swirl* include basic building blocks of R, workspace and files, sequences, vectors, missing values, arrays, logic, functions, dates and times, and graphics (e.g., plotting).

Finally, it is hoped that this chapter may provide helpful resources that can be used creatively within R courses for undergraduate and postgraduate sport science students. It is this authors opinion that early career sport scientists hamper their development by avoiding tools of modern research such as programming languages and should become familiar with their use as early as possible. In the right circumstances, programming languages, and in particular R, offer excellent interactive environments for learning about aspects of modelling, statistical analysis, and research.

6.2 Working with FFMs in R

This section covers the development of code suitable for simulating and fitting the standard, fitness-delay and variable dose-response (VDR) models (Banister *et al.*, 1975; Calvert *et al.*, 1976; Busso, 2003) via two contrasting approaches: 1) explicit for-loops; 2) the apply family of R functions. The two approaches were demonstrated rather than just one to increase the breadth of the readers skillset when working with FFMs. Fitting the models is demonstrated under nonlinear least squares and maximum likelihood estimation perspectives. VDR model behaviour is also examined in greater depth than the literature review with plots provided for different values of τ_{h_2} in the variable-gain term. Processes related to optimisation are then examined and the availability of different algorithms in R highlighted. After this, the expanding-window CV method developed for fitting and testing FFMs. At several key points over the course of the section, reproducible examples are presented that are self-contained in that they do not require external data files due to the generation of synthetic data under the simulation functions developed. These can therefore be run by the reader more easily to demonstrate the processes described and provide an interactive element for learning about aspects of the experimental work within this thesis.

First, the section begins by examining the standard model, described by:

$$\hat{p}(t) = p^* + k_g \cdot \sum_{i=1}^{t-1} \omega_i \cdot e^{-\frac{(t-i)}{\tau_g}} - k_h \cdot \sum_{i=1}^{t-1} \omega_i \cdot e^{-\frac{(t-i)}{\tau_h}} \quad (6.1)$$

Implementation of equation 6.1 in R requires care to avoid small errors. One potential source of confusion when implementing equation 6.1 relates to the index (i) placed on the sum within each component. Consider the case of where a researcher wishes to compute $\hat{p}(t)$ for day $t = 1$, calculation of each component (fitness and fatigue) then involves a sum running from $i = 1$ to 0, which by mathematical convention is zero, and so $\hat{p}(1) = p^*$. However, R does not default to this convention, particularly when a loop is used to compute each component sum. A useful workaround to this is to include the initial condition $\omega_0 = 0$ at the first position in the model inputs (ω), and execute the loop over the index running from $i = 1$ to t, such that for $t = 1$ the model component would simply be $\omega_0 \cdot e^{-\frac{0}{\tau_g}} = 0$, yielding $\hat{p}(1) = p^*$, implicitly retaining the $(t-1)$ condition on the upper sum index. This approach will become less abstract as code is presented shortly but is described first as it is important to highlight that implementation of models can accidently run into 'off-by-1' type errors, shifting the computation by one day in either direction. Over the long run such rounding errors are unlikely to be of practical significance but can cause confusion when comparing solutions. Additionally, the component's run from $i = 1$ to $t - 1$, rather than to t in equation 6.1 to reflect the notion that training performed on day t is not considered within model predicted performance $\hat{p}(t)$. This convention is often referred to as a 'pre-loaded' model (i.e., performance occurs prior to load on a given day t) (Rasche and Pfeiffer, 2019). Conceptually, in most circumstances this matches practice as physical performance testing is likely to take place prior to any heavy training session or period of physical exertion that may influence the measurement.

Lastly, recall that initial fitness and fatigue effects can be included in the model formulation via the use of two further terms with parameters q_g, q_h, as shown in equation 6.2 (Busso, Carasso and Lacour, 1991)

$$\hat{p}(t) = p^* + \underbrace{q_g \cdot e^{-\frac{t}{\tau_g}}}_{initial\ fitness} - \underbrace{q_h \cdot e^{-\frac{t}{\tau_g}}}_{initial\ fatigue} + k_g \sum_{i=1}^{t-1} \omega(i) \cdot e^{-\frac{t-i}{\tau_g}} - k_h \sum_{i=1}^{t-1} \omega(i) \cdot e^{-\frac{t-i}{\tau_h}} \quad (6.2)$$

Where, residual and fatigue effects are assumed to dissipate at the same rate as future response (τ_g, τ_h). In this section, functions for the standard and VDR models have been developed to incorporate these initial components as optional 'add-ons' that can be estimated or included in model simulation.

6.2.1 An explicit loop approach – Simulating the standard, fitness-delay, and VDR FFM and fitting via nonlinear least-squares

Two types of user-defined R functions are presented here for three FFMs (standard, fitness-delay, VDR): 1) a sum-of-squared residuals (RSS) objective function for calculation of $\sum(modelled - measured)^2$ over the set of measured performances; 2) a simulation function for computing model predictions under a set of inputs and specified model parameters. Note, the term *function* is mainly used in this chapter to refer to a module of code to accomplish a specific task, rather than the mathematical context of mapping domains to a range. The term 'user defined' is included in some cases to help distinguish this meaning. The objective functions for RSS will take as input:

1. A series of consecutive training loads, over a period $[0, T]$, where $\omega_0 = 0$
2. A vector of numeric parameter values (of length and order dependent upon the model involved)
3. A set of n measured performance values $(p_1, \ldots, p_n)$ collected within the period $[1, T]$

And return as output the sum-of-squared residuals (errors) between modelled and measured performance values, i.e., $\sum_{i=1}^{n}(\hat{p}_i - p_i)^2$. When FFMs are fitted to data, minimising the value returned from this function under different parameters becomes the objective of an optimisation algorithm, hence its name.

The model simulation functions (also sometimes called the 'prediction function' or 'computation function') require as input only a series of training load values and suitable parameter values, returning as output model simulated performance values over the length of the input series, along with the associated fitness and fatigue traces. This type of function is most useful for generating modelled performance values once suitable parameters have already been obtained/identified (e.g., via fitting to data), or for exploring model behaviour. Implementations presented in this section make use of one of the most basic and powerful programming structures, the loop. In the next subsection (6.2.2) that examines maximum likelihood estimation, an alternative approach to developing objective and simulation functions is demonstrated utilising the 'sapply' function from the apply family of base R functions. The apply functions are a form of 'loop hiding' that can result in "cleaner" and thereby more efficient code. Throughout subsections 6.2.1 and 6.2.2, inputs passed to functions described are consistently structured as described in Table 6.1. Readers are encouraged to be hands on with the code throughout this chapter, and code related to this subsection is available to download from **github.com/bsh2/thesis/c6/nls.R**.

Table 6.1: Function arguments

Argument	Description	Structure	Example
`loads`	Model prediction functions: Start at day 0 ($\omega_0 = 0$), of any length, 1-day time steps, use of a zero value on a given day denotes the absence of training on that day. For the objective functions: Start at day 0 ($\omega_0 = 0$), only need to run to some day T (where T is the same day as the final performance measurement within the model training set), 1-day time steps, use of a zero value on a given day denotes the absence of training on that day.	Data frame Observations of 2 variables: day, load	`data.frame("day" = c(0, 1, 2, 3, 4, 5), "load" = c(0, 100, 0, 100, 0, 100))`
`perfVals`	Measured physical performance values p_t from within the period $t = 1$ to T, these values are used to fit the model via the objective function.	Data frame Observations of 2 variables: day, performance	`data.frame("day" = c(1, 2, 5), "performance" = c(500, 495, 505))`

`pars`	Model parameter values of a specific length and order (specified as required in the relevant code).	Vector (numeric)	`pars <- c(100, 1, 25, 1.2, 6)`
			`c(p*, k_g, T_g, k_h, T_h)`
			for the standard model

`initialPars`	**For the simulation functions only:** Initial model parameter values q_g, q_h	Vector (numeric)	`initialPars <- c(qg,qh)`
`initial`	**For the objective function only:** A logical value indicating if the initial model parameter values q_g, q_h are to be estimated. The vector pars would also be updated to reflect this.	Logical `(TRUE / FALSE)`	`initial = FALSE` `(default)`
`maximise`	**For the objective function only:** Adjusts the objective function (whether it be NLS or MLE) to account for optimisation algorithms that maximise (`maximise = TRUE`). If an algorithm minimises, then `maximise = FALSE`.	Logical `(TRUE / FALSE)`	`maximise = FALSE` `(default)`

Presented below in listing 6.1 is an implementation of a for-loop based approach to develop a sum-of-squares objective function for the standard model. Although loops written in high-level languages such as R are not always efficient, they remain a powerful programming structure and in many cases are quick to write and simple to debug. If required, the user can manually step through each iteration to identify where operations are breaking down. However, when nested loops are used, control flows can quickly become complicated and challenging to follow. In such instances, alternative approaches such as the apply family of base R functions (which provide a form of 'loop hiding') (Burns, 2011) may be simpler to work with and reflect better programming practice (Wickham, 2019). It is hoped that by

presenting and discussing both approaches in R, this material will serve to expedite the skill acquisition process for researchers who may be considering these same issues. It also offers appropriate educational material for those involved in teaching R to developing sport scientists, providing both context and introduction to concepts such as loops, vectorisation, writing functions, and the apply family. Arguably, the FFM reflects a good vehicle for teaching both quantitative modelling in sport (Clarke and Skiba, 2013), and aspects of the R programming language.

Listing 6.1 Objective function (RSS): Standard model – For loop approach

```
standardObjectiveSS <- function(pars, loads, perfVals, initial = FALSE,
                                maximise = FALSE){

  # (pars) NOTES:
  # ------------------------------------------------------------------------
  # Pos.     [1] [2] [3] [4]  [5] [6] [7]
  # pars = c(p*, kg, Tg, kh, Th)          (if initial = FALSE)
  # pars = c(p*, kg, Tg, kh, Th, qg, qh)  (if initial = TRUE)
  # ------------------------------------------------------------------------
  nMeasurements <- length(perfVals$performance)   # Number of performance measurements

  # Zeroed vector of length equal to number of performance measurements
  squaredResiduals <- numeric(length = nMeasurements)

  # For each performance measurement calculate (modeled - measured)^2 under pars
  for (n in 1:nMeasurements){

    dayT <- perfVals$day[n]              # Day of measured performance
    measured <- perfVals$performance[n]  # Measured performance value on dayT

    # Isolate the required load data to compute the model up to dayT
    # Note: 1:dayT rather than 1:(dayT - 1) as the first row in the loads array is w(0)=0
    inputSubset <- loads[1:dayT, ]

    if (initial == TRUE){
      # Include residual effects (initial components)
      initFitness <- pars[6] * exp(-(dayT) / pars[3])
      initFatigue <- pars[7] * exp(-(dayT) / pars[5])
    } else{
      initFitness <- 0
      initFatigue <- 0
    }

    # Compute modelled performance on dayT
    model <- pars[1] + initFitness - initFatigue +
      pars[2] * (sum(inputSubset$load * exp(-(dayT - inputSubset$day) / pars[3]) ) ) -
      pars[4] * (sum(inputSubset$load * exp(-(dayT - inputSubset$day) / pars[5]) ) )

    # Compute the squared residual value (model - measured)^2
    squaredResiduals[n] <- (model - measured)^2
  } # Loop updates until iterated over all available measurements

  # Output
  if(maximise == FALSE){ return(sum(squaredResiduals))}
  if(maximise == TRUE){
    # For optimisation algorithms that maximise by default
    return(-1 * sum(squaredResiduals))
  }
}
```

We now discuss elements of listing 6.1 that are helpful to unpack, and that are used frequently throughout the following sections, such as *vectorisation*. In addition, only once in this subsection will the control flow be described in so much detail, as the remainder of the code presented for the more advanced models (fitness-delay, VDR) follows much the same structure and so only differences need

to be examined to avoid unnecessary repetition. In listing 6.1, the user-defined R function (`standardObjectiveSS`) takes as input a numeric vector of ordered parameter values (`pars`) for the standard model, a dataframe of training load values (`loads`) (with 1-day time-step between rows, from day 0), and a dataframe of performance values (`perfVals`) where the time-step between rows variable and is not necessarily regular (See Table 6.1). Additional arguments include (`initial`) and (`maximise`), that are also described in table 6.1. The order of the elements in (`pars`) depends on `initial = TRUE (default)` such that `pars =` $(p^*, k_g, \tau_g, k_h, \tau_h)$ or `initial = FALSE` such that `pars =` $(p^*, k_g, \tau_g, k_h, \tau_h, q_g, q_h)$. The function returns as output a single value, the RSS between the set of measured performance values and modelled performance values (computed under `loads` and `pars`). At line 10, a scalar object (`nMeasurements`) is assigned to the number of measured performance values in the dataframe (`perfVals`). Subsequently, at line 13, a zeroed vector (`squaredResiduals`) of length equal to (`nMeasurements`) is assigned. This vector is updated at each iteration of the loop (at the position denoted by the current value of the loop index (`i`), i.e., `squaredResiduals[i]`. As the length has been pre-assigned, it does not need to 'grow', which is generally advised against for memory allocation and efficiency reasons (Burns, 2011). The action of the for-loop structure between lines 16 and 41 is to iterate over the set of measured performance values (`perfVals`), and at each measured performance value p_i, calculate modelled performance $\hat{p}(i)$ under (`loads`) and (`pars`), and record the squared difference of the two. This difference is collected, as described, in the (`squaredResiduals`) vector, before the loop updates to the next step. This process repeats (i.e., loops) until the complete sum of squared residuals for all measured values has been calculated, before returning this value as the output of the function.

Next, we briefly discuss the use of vectorisation and how it applies in listing 6.1, specifically between lines 36-37 in computing modelled performance on some day t. R is a high-level language, and the default implementation is 'interpreted'. In practice, this means consideration of concepts such as memory allocation, floating points, and type declarations are largely taken for granted and dealt with internally. This is particularly useful in environments where scientists and practitioners seek to solve problems quickly and prototype, without layers of complexity within the implementation process. These types of issues require investment to understand and be proficient with. A downside of high-level interpreted languages is that they are often much slower at performing certain tasks compared to lower-level compiled languages such as C++, Fortran, and Rust. In these languages, the programmer must consider more closely their code particularly with respect to the issues outlined above. However, many basic functions available in base R are written in compiled languages such as C. In fact, R simply provides a 'wrapper' function (doorway) to this compiled code. Often, an R user will seek to perform the same operation on multiple elements of a similar type (e.g., integers), stored in vector objects. Instead of having to call a function repeatedly for each value of the vector via the use of an explicit

loop structure (as above), the whole vector can often be passed through a 'vectorised' function that will typically also provide access to fast compiled code. In this circumstance the code is almost always faster because the interpretation step happens only once, and the software can execute the computation to each element (so long as it is of the same type). As a basic illustration, consider the case where a user wishes to calculate the natural log of each value of a vector (`z`) using the base R function (`log`). There are two contrasting approaches that could be used:

Approach 1 (explicit loop)	Approach 2 (vectorised)
`for (i in 1:length(z)){ log(z[i]) }`	`log(z)`

Approach 1 is overly verbose, and more importantly creates unnecessary overhead as the R loop must iteratively call the natural log function on each element. In this case, the interpretation step is happening at each iteration (i.e., the computer has to re-check what it is being asked to do). In contrast, the second approach is more efficient. In approach 2 the entire vector (`z`) is passed to the (`log`) function, performing the equivalent operation at a much lower computational cost. This is because the (`log`) function has been written in a lower-level compiled language that accepts an entire vector as input through an R function that acts as the wrapper. Although the whole vector is still processed inside a loop (or similar construct) within this lower-level code, it is faster because as stated the interpretation step happens only once, and also loops are often faster when written in lower-level code. Differences in speed are demonstrated via the code below:

```
# Create a large vector
  z <- seq(1, 10^8, 1)
# Pre-assign output length to detach influence of vector growth in the loop
  outcome <- numeric(length = length(z))
```

The following runtimes emerge:

```
# Approach 1 (explicit R loop)
  system.time(for (i in 1:length(z)){
              outcome[i] <- log(z[i])})
```

user	system	elapsed
7.719	0.036	7.854

```
# Approach 2 (vectorisation)
  system.time(outcome <- log(z))
```

user	system	elapsed
1.104	0.317	1.491

This example demonstrates that the explicit loop is almost 7 times slower than the vectorised approach (for a large vector). When the use of vectorisation is discussed in R, it is not expected that the user will be writing their own functions in low-level compiled code (which would defeat the main advantage of R in the first place). Rather, that they should make use of functions within base R or other packages that provide various 'free' forms of vectorisation via compiled functions. For example, functions such

as: (`sum, log, mean, prod, max, min, range, mean, mode`) all facilitate vectorisation of some form. Arithmetic operators in R (`+, *, -, /`) also provide a cheap way to perform arithmetic operations on elements that are in the same index position in two or more vectors of equal length. For example, consider two vectors:

```
x <- c(1, 2, 3, 4)          y <- c(5, 6, 7, 8)
```

Then the code:

```
z <- numeric(length = 4)
for (i in 1:length(x)){ z[i] <- x[i] + y[i] }
```

Is equivalent (*in outcome*) to `(z <- x + y)`. In both approaches, the following is happening (*could be restated as*):

```
z <- c((1+5),(2+6),(3+7),(4+5))
```

However, the latter approach is faster, marginally in this case because of the small length of the vector, but results in noticeably cleaner code. Returning to listing 6.1, it becomes clearer how vectorisation has been used to compute the FFM by first calculating each component state using a vectorised application of the (`sum`) and (`exp`) functions in base R, alongside correct application of the arithmetic operators (`+, *, -, /`).

This can be seen by breaking down a portion of line 36, that is:

```
(sum(inputSubset$load * exp(-(dayT - inputSubset$day) / pars[3])))
```

Consider an example such that the objects used in the code above posses the following values:

```
inputSubset$load <- c(0, 100, 200)    # Vector
inputSubset$day <- c(0, 1, 2)         # Vector
pars[3] <- 2                          # Scalar
dayT <- 3                             # Scalar
```

If the control flow of the above extract of line 36 (listing 6.1) is looked at more closely, vectorisation as it applies becomes clearer, as follows:

```
dayT - inputSubset$day
        ⇔ c((3-0), (3-1), (3-2))

exp(-(dayT - inputSubset$day) / pars[3])
        ⇔ c(exp(-1 * (3-0)/2 ), exp(-1 * (3-1)/2 ), exp(-1 * (3-2)/2))

inputSubset$load * exp(-(dayT - inputSubset$day) / pars[3])
        ⇔ c((0 * exp(-1 * (3-0)/2)), (100 * exp(-1 * (3-1)/2 )), (200 * (exp(-1* (3-2)/2) )))

sum(inputSubset$load * exp(-(dayT - inputSubset$day)/pars[3]))
        ⇔ (0*exp(-(3-0)/2)) + (100*exp(-(4-2)/2)) + (200*(exp(-(3-2)/2))
```

⇔ (denotes '*equivalent in outcome to / can be restated as*')

The other component in listing 6.1 is computed via the same concepts. Hopefully now the control flow of listing 6.1 is clearer as well as the concepts underpinning its implementation. It should also become clear from listing 6.2 next that the only modification required to implement the fitness-delay model (Calvert *et al.*, 1976) is a change at lines 36-37 of listing 6.1.

Listing 6.2 Modification of listing 6.1 (lines 36-37) to incorporate the fitness-delay model (RSS objective)

```
1  pars[2] * (sum(inputSubset$load *
2                 (exp(-(dayT - inputSubset$day) / pars[3]) -
3                  exp(-(dayT - inputSubset$day) / pars[4])))) -
4  pars[5] * (sum(inputSubset$load * exp(-(dayT - inputSubset$day) / pars[6])))
```

Notes (listing 6.2):

- If `initial = TRUE`, `pars` is now a vector of length 8 with order: `c(p*, k_g, Tau_g1, Tau_g2, k_h, Tau_h, qg, qh)`
- If `initial = FALSE`, `pars` is now a vector of length 6 with order: `c(p*, k_g, Tau_g1, Tau_g2, k_h, Tau_h)`
- This modified function has been presented in full in: **github.com/bsh2/thesis/c6/nls.R**

Via a similar approach to the one shown so far using an outer loop, a simulation function (`simulateStandard`) can be developed to compute performance $\hat{p}(t)$ and isolate the fitness/fatigue traces of the standard FFM for a training load series of any length (under an associated set of model parameters). This function is presented in listing 6.3.

Listing 6.3 Simulation function: Standard model – For loop approach

```
1  simulateStandard <- function(pars, loads, initialPars = c(0,0), returnObject = "all"){
2
3    # pars = c(p*,kg,Tg,kh,Th)
4
5    # Set up zeroed vectors of required length
6    seriesLength <- tail(loads$day, 1)
7    performance <- numeric(length = seriesLength)     # Model performance
8    fitness <- numeric(length = seriesLength)         # Model fitness
9    fatigue <- numeric(length = seriesLength)         # Model fatigue
10   initialFitness <- numeric(length = seriesLength)  # Residual effects (initial component)
11   initialFatigue <- numeric(length = seriesLength)
12
13   # Calculate model fitness g(t), fatigue h(t), and
14   # performance  p(t) for t = 1:seriesLength
15   for (t in 1:seriesLength){
16
17     # Isolate the required load data for calculating p(t)
18     # (i.e., loads from day 0 to day t-1)
19     inputSubset <- loads[loads$day < t, ]
20
21     # Residual effects from initial components at time point t
22     initialFitness[t] <- initialPars[1] * exp(-(t) / pars[3])
23     initialFatigue[t] <- initialPars[2] * exp(-(t) / pars[4])
24
25     # Compute g(t), h(t), p(t) for current (t)
26     fitness[t] <- pars[2] * sum(inputSubset$load * exp(- (t - inputSubset$day)/ pars[3]))
27     fatigue[t] <- pars[4] * sum(inputSubset$load * exp(- (t - inputSubset$day)/ pars[5]))
28     performance[t] <- pars[1] + fitness[t] - fatigue[t] +
29                               initialFitness[t] - initialFatigue[t]
30
31   } # Loop index updates (t <- t+1) until t = seriesLength
32
33   # Output
34   if (returnObject == "performance"){return(performance)}
35   if (returnObject == "fitness"){return(fitness)}
36   if (returnObject == "fatigue"){return(fatigue)}
37   if (returnObject == "all"){
38     return(data.frame("day" = 1:seriesLength,
39                       "initial_fitness" = initialFitness,
40                       "initial_fatigue" = initialFatigue,
41                       "fitness" = fitness, "fatigue" = fatigue,
42                       "performance" = performance))
43   }
44
45 } # End function (closing bracket)
```

In listing 6.3, the `returnObject` argument is added, by default returning all the information (fitness, fatigue, performance, states of the initial effects) as a dataframe, however, vectors of individual model components or performance can be returned by supplying the appropriate character string to this argument, which can be useful for plotting purposes. This argument simply adds more flexibility to the function. Just as in the objective function for the standard model (listing 6.1), the model simulation function above (listing 6.3) can be adapted to accommodate the fitness-delay model, by replacing lines 21-25 as follows (listing 6.4).

Listing 6.4 Modification of listing 6.3 (lines 21-25) to incorporate the fitness-delay model (simulation)

```
1  initialFatigue[t] <- initialPars[2] * exp(-(t) / pars[6])
2  fitness[t] <- pars[2] * sum(inputSubset$load * (exp(-(t - inputSubset$day) / pars[3])  -
3                                                  exp(-(t - inputSubset$day) / pars[4])))
4  fatigue[t] <- pars[5] * sum(inputSubset$load * exp(-(t - inputSubset$day) / pars[6]))
```

Notes (listing 6.2):

- `pars` is now a vector of length 6 with order: `c(p*, k_g, Tau_g1, Tau_g2, k_h, Tau_h)`
- This modified function has been presented in full in: **github.com/bsh2/thesis/c6/nls.R**

Considered next in this section is the conceptually more attractive variable dose-response (VDR) model from Busso (2003). Before developing code for the objective function and simulating this model, key theoretical concepts are revisited. Recall the VDR model was stated mathematically in this thesis as follows:

$$\hat{p}(t) = p^* + k_g \sum_{i=1}^{t-1} \omega_i \cdot e^{-\frac{t-i}{\tau_g}} - k_h \sum_{i=1}^{t-1} k_{h_2}(i) \cdot e^{-\frac{t-i}{\tau_h}} \tag{6.3}$$

Where:

$$k_{h_2}(i) = \sum_{j=1}^{i} \omega_j \cdot e^{-\frac{i-j}{\tau_{h_2}}} \tag{6.4}$$

The variable gain term (equation 6.3) adds an extra level of recursion to the R functions developed so far, and therefore requires some care to implement. Inclusion of this additional term can be achieved by incorporating an additional for-loop, nested within the main loop, that generates a vector of $k_{h_2}(i)$ terms from 1 to some i. In the following subsection it is shown that there is a 'cleaner' way to complete this task, via the use of the apply family of base R functions. But for now, this section will proceed with the loop-based approach developed so far.

In Busso (2003), the model stated is also slightly different compared to equations 6.3 and 6.4. The difference isn't an unintentional error, but rather a purposeful modification. The original Busso (2003) model was described by:

$$\hat{p}(t) = p^* + k_g \sum_{i=1}^{t-1} \omega_i \cdot e^{-\frac{t-i}{\tau_g}} - \sum_{i=1}^{t-1} k_{h_2}(i) \cdot \omega_i \cdot e^{-\frac{t-i}{\tau_h}} \tag{6.5}$$

Where:

$$k_{h_2}(i) = k_h \sum_{j=1}^{i} \omega_j \cdot e^{-\frac{i-j}{\tau_{h_2}}} \tag{6.6}$$

The noticeable differences between the VDR model formulae in equations (6.3, 6.4) vs. (6.5, 6.6) are:

- The training load term (ω) appears twice in the fatigue component in equations (6.5, 6.6), versus once in equations (6.3, 6.4). Note the ω_i term in equation 6.5 and ω_j in equation 6.6.
- The k_h term in equation 6.6 was moved to the outside of the fatigue component sum in equation 6.3. This change generates no computational difference.

The former of these differences, the inclusion of an additional training load term in the original Busso (2003) VDR model, is only significant when approximating the model over a short period of time. This is because FFMs are defined by systems of differential equations whose solutions require integration. The summations presented as FFM equations therefore represent approximations with integration replaced by rectangular summation. Over very short periods modifications to indexing etc, may substantively influence approximations and therefore differences. However, over-longer periods suitable approximations will present differences that are not practically significant. The formula for the VDR model in equations 6.3 and 6.4 (i.e., the form as consistently stated through this thesis) also makes the purpose of the variable gain term clearer. That is, the purpose of the term is to weight the fatigue effect of training based on previous training doses (proximity and magnitude). In essence, the process acts as an exponentially weighted average of previous training loads with more weight applied to more recent loads. Listing 6.5 presents the VDR model objective function, based on its mathematical form as presented in this thesis so far (equations 6.3, 6.4), and listing 6.6 demonstrates a modification to listing 6.5 in order to return to the original formulae (equations 6.5, 6.6).

Listing 6.5 Objective function (RSS): VDR model – For loop approach

```
1  vdrObjectiveSS <- function(pars, loads, perfVals, initial = FALSE, maximise = FALSE){
2
3    # INPUT NOTES:
4    # ----------------------------------------------------------
5    #         [1] [2] [3] [4] [5] [6]  [7] [8]
6    # Pars: c(p*, kg, Tg, kh, Th, Th2)          initial = FALSE
7    # Pars: c(p*, kg, Tg, kh, Th, Th2, qg, qh)  initial = TRUE
8    # ----------------------------------------------------------
9
10   nMeasurements <- length(perfVals$performance)  # Number of performance measurements
11
12   # Zeroed vector of length equal to number of performance measurements
13   squaredResiduals <- numeric(length = nMeasurements)
14
15   # For each performance measurement calculate (modelled - measured)^2 under pars
16   for (n in 1:nMeasurements){
17
18     dayT <- perfVals$day[n]                # Day of measured performance
19     measured <- perfVals$performance[n]    # Measured performance value on dayT
20
21     # Isolate the required load data to compute the model up to dayT
22     inputSubset <- loads[1:dayT, ]
23
24     # Initial components
25     if (initial == TRUE){
26       initFitness <- pars[7] * exp(-(dayT) / pars[3])
27       initFatigue <- pars[8] * exp(-(dayT) / pars[6])
28     } else{
29       initFitness <- 0
30       initFatigue <- 0
31     }
32
33     # Set up a zeroed vector to hold the variable gain term values kh2(i) for i=0 to dayT - 1
34     kh2 <- numeric(length = dayT)  # Variable gain term vector
35
36     # Calculate the variable gain term kh2(i) for i=0,1,2,...,dayT-1 (Recursive)
37     for (i in 1:dayT){
38       kh2[i] <- sum( inputSubset$load[1:i] * exp( -((inputSubset$day[i]-
39                                                      inputSubset$day[1:i])/pars[6])))
40     }
41
42     # Compute modelled performance on dayT under pars
43     model <- pars[1] + initFitness - initFatigue +
44       pars[2] * (sum(inputSubset$load * exp(-(dayT - inputSubset$day) / pars[3]))) -
45       pars[4] * (sum(kh2 * inputSubset$load * exp(-(dayT - inputSubset$day) / pars[5])))
46
47     # Compute the squared residual value (model - measured)^2
48     squaredResiduals[n] <- (model - measured)^2
49
50   }
51
52   # Output
53   if(maximise == FALSE){return(sum(squaredResiduals))}
54   if(maximise == TRUE){return(-1 * sum(squaredResiduals))}
55 }
```

Reflecting the original Busso (2003) model (eq.'s 6.5, 6.6) is straightforward and involves only changing listing 6.5 at line 45, as follows:

Listing 6.6 Modification of listing 6.5 (line 45) to incorporate original VDR formula (RSS)

```
1  pars[4] * (sum(kh2 * inputSubset$load * exp(-(dayT - inputSubset$day) / pars[5])))
```

The last function presented in this subsection is a VDR model prediction function. The code in listing 6.7 follows all of the same concepts discussed so far throughout this work, and so is not explained beyond the annotation itself.

Listing 6.7 Simulation function: VDR model – For loop approach

```
simulateVDR <- function(pars, loads, initialPars = c(0,0), returnObject = "all"){

  # pars = c(p*, k_g, Tau_g, k_h, Tau_h, Tau_h2)

  # Set up zeroed vectors of required length
  seriesLength <- tail(loads$day, 1)
  performance <- numeric(length = seriesLength)     # Model performance
  fitness <- numeric(length = seriesLength)         # Model fitness
  fatigue <- numeric(length = seriesLength)         # Model fatigue
  initialFitness <- numeric(length = seriesLength)  # Residual effects (initial component)
  initialFatigue <- numeric(length = seriesLength)
  kh2Dat <- matrix(data = NaN, nrow = tail(loads$day, 1), ncol = tail(loads$day, 1))

  # Calculate model fitness g(t), fatigue h(t), and
  # performance  p(t) for t = 1:seriesLength
  for (t in 1:seriesLength){

    # Isolate the required load data for calculating p(t), (i.e., loads from day 0 to day t-1)
    inputSubset <- loads[loads$day < t, ]

    # Residual effects from initial components at time point t
    initialFitness[t] <- initialPars[1] * exp(-(t) / pars[3])
    initialFatigue[t] <- initialPars[2] * exp(-(t) / pars[4])

    # Set up a zeroed vector to hold the variable gain term values kh2(i) for i=0 to dayT - 1
    kh2 <- numeric(length = t)  # Variable gain term vector

    # Calculate the variable gain term kh2(i) for i=0,1,2,...,dayT-1 (Recursive)
    for (i in 1:t){
      kh2[i] <- sum(inputSubset$load[1:i] *
                    exp( -((inputSubset$day[i]-inputSubset$day[1:i])/pars[6])))
    }
    # For each iteration t+1 of the outer loop, save kh2(i) values where i = 0 to t-1
    kh2Dat[1:t, t] <- kh2

    # Compute g(t), h(t), p(t) for current t
    fitness[t] <- pars[2] * sum(inputSubset$load * exp(- (t - inputSubset$day)/ pars[3]))
    fatigue[t] <- pars[4] * sum( kh2 * exp(- (t - inputSubset$day) / pars[5]) )
    performance[t] <- pars[1] + fitness[t] - fatigue[t] + initialFitness[t] - initialFatigue[t]

  } # Loop index updates (t <- t+1) until t = seriesLength

  # Output
  if (returnObject == "performance"){return(performance)}
  if (returnObject == "fitness"){return(fitness)}
  if (returnObject == "fatigue"){return(fatigue)}
  if (returnObject == "all"){
    return(data.frame("day" = 1:seriesLength,
                      "initial_fitness" = initialFitness,
                      "initial_fatigue" = initialFatigue,
                      "fitness" = fitness, "fatigue" = fatigue,
                      "kh2_t" = kh2Dat[, tail(loads$day, 1)],
                      "performance" = performance))
  }
} # End function (closing bracket)
```

Next the function (`simulateVDR`) for simulating the VDR model from listing 6.7 is applied to study the behaviour of the fatigue component $h(t)$ and in particular the variable gain term k_{h_2}, via plots where the value of parameter τ_{h_2} is varied under different training load patterns (e.g., constant, binary, increasing, undulating, mixed/variable). This begins with the simple case of constant training load (Figure 6.4).

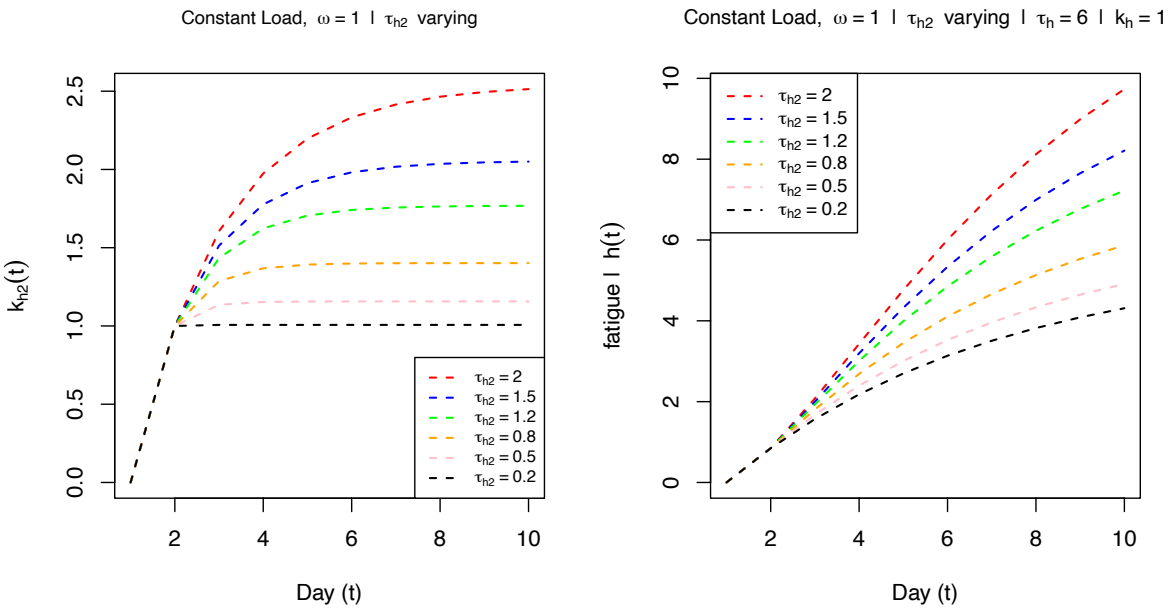


Figure 6.4: Behaviour of the variable gain term and fatigue component $h(t)$ of the VDR model for different values of the parameter τ_{h_2} (constant load: $\omega = 1$, parameters fixed: $k_h = 1, \tau_h = 6$).

Note: $k_{h_2}(t) = \sum_{j=0}^{t} \omega(j)\, e^{-\left(\frac{t-j}{\tau_{h_2}}\right)}$

Code associated with figure 6.4 is contained in **github.com/bsh2/thesis/c6/nls.R**., with the plots in figure 6.7 generated in a similar fashion. The plots in figure 6.4 illustrate that for lower values of the parameter τ_{h_2}, the VDR limited interaction in the fatigue component with previous training sessions. This is reflected in the variable gain term in response to a new load. To see this, examine the left hand plot for $\tau_{h_2} = 0.2$ (black line), compared to $\tau_{h_2} = 2$ (red line). Note that as t increases from day 1 to 10, the variable gain term ($\tau_{h_2} = 0.2$, black line) stays approximately constant. This tells us that on any day t, the loads on days $(1,2,\ldots,t-1)$ have very little scaling effect on the response to a new load within the fatigue component (for small values of τ_{h_2}). In contrast, for the red line, the effect of previous training doses spans further back and is higher in magnitude. Larger values of τ_{h_2} model greater interaction with previous training sessions in the response to a new load. For the example above, the response in the fatigue component to a new load on day t for $\tau_{h_2} = 0.2$ could be approximated by just $e^{-\frac{1}{\tau_h}}$. To illustrate this further, Figure 6.5 below shows the difference between these same values of τ_{h_2} by the associated contribution of previous loads to the variable gain term $k_{h_2}(t)$ on $t = 10$, under constant load $\omega = 1$ [a.u].

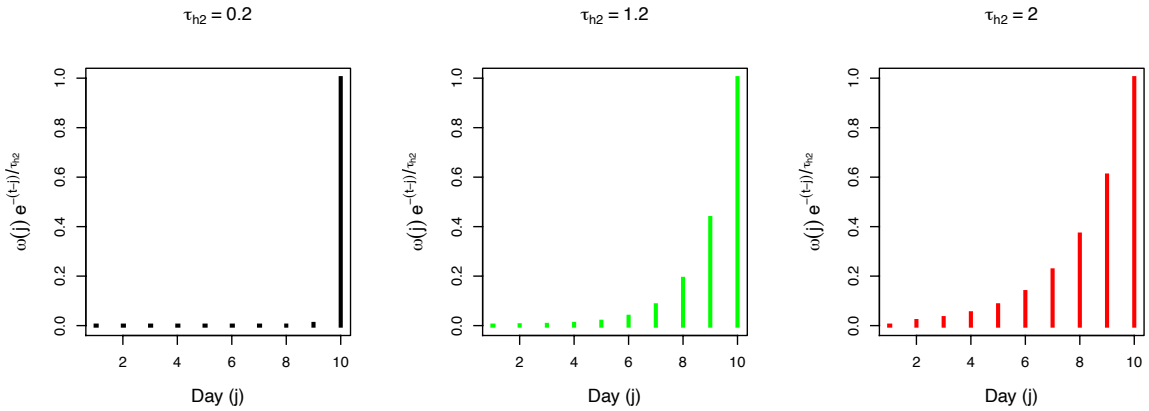


Figure 6.5: Differences between τ_{h_2} parameter values (0.2, left, black; 1.2, middle, green; 2, right, red) reflected in previous loads contribution to the variable gain term $k_{h_2}(t)$, for $t = 10$, under $\omega = 1$ (constant).

Several further training inputs (Figure 6.6) are now plotted (Figure 6.7) via similar means to further exhibit the behaviour of the variable gain term $k_{h_2}(t)$ and fatigue $h(t)$ component in the VDR model.

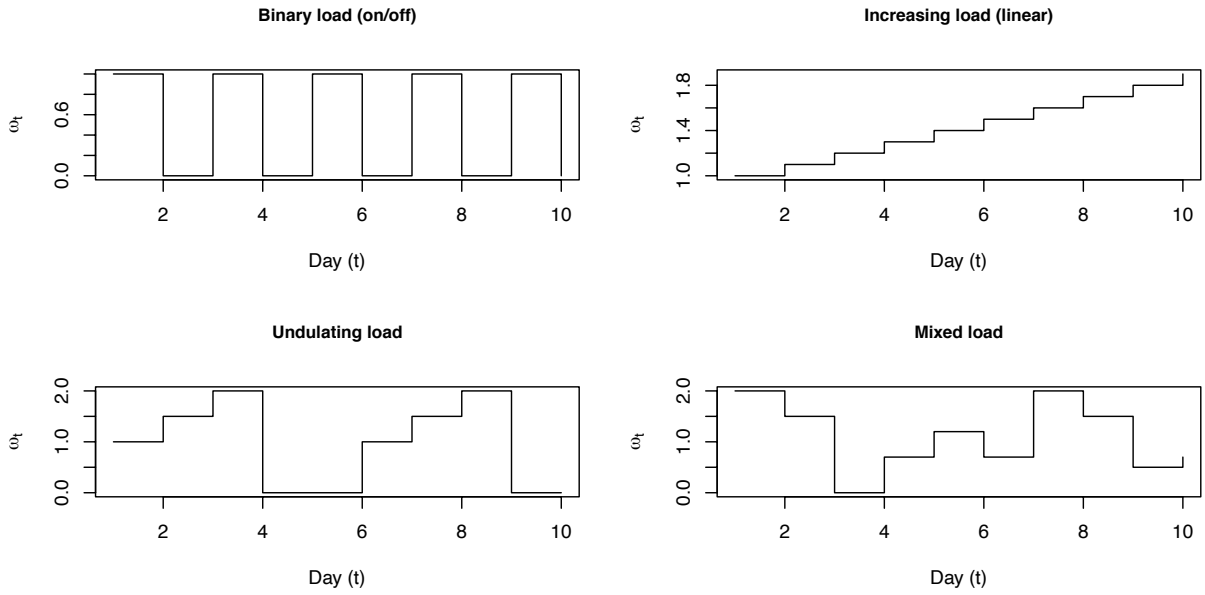


Figure 6.6: Training inputs applied to illustrate the behaviour of the VDR model

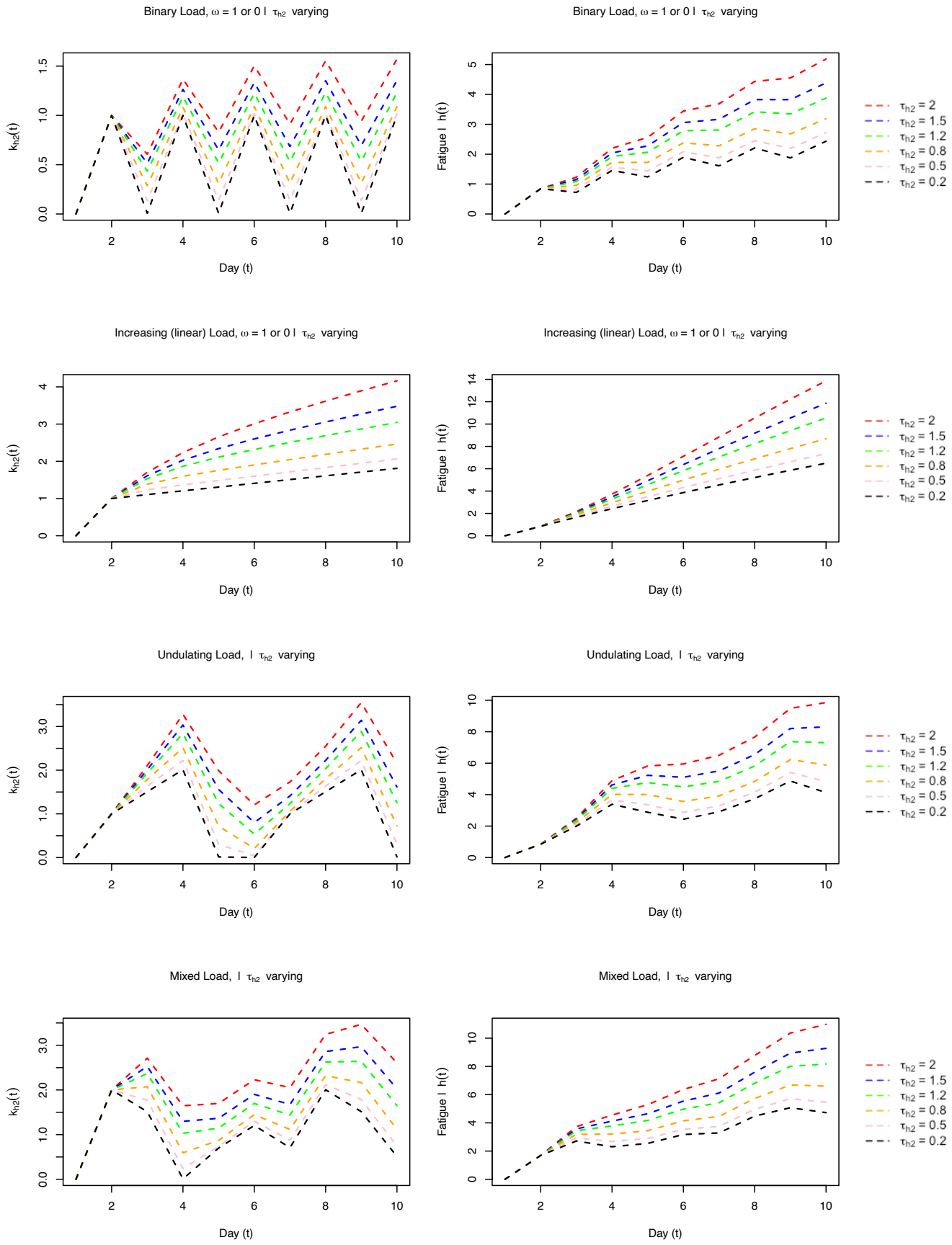


Figure 6.7: VDR behaviour ($k_{h_2}(t)$, $h(t)$ for varied τ_{h_2}, $k_1 = 1, \tau_h = 6$ fixed) under different loads

Fitting an FFM to data via nonlinear least-squares (A self contained example)

Closing this subsection, a reproducible example is presented, demonstrating how the RSS objective functions developed for the three FFMs can be applied in a nonlinear least-squares fitting process to estimate FFM parameters from data. The L-BFGS-B algorithm from the popular general purpose optimisation package *optimx* (Nash et al., 2020) available on the comprehensive R archive network (CRAN, cran.r-project.org) is used in the example. However, appropriate choice of optimisation algorithm and arguably better alternatives are discussed in section 6.2.4. As demonstrated in section 6.2.4, applying different algorithms available from various packages in R is largely a plug and play task, and so choice of algorithm here matters very little for demonstrating the overall picture. In other words, focus here is not on the suitability of algorithm used, parameters obtained, synthetic data developed, or model predictions. Instead, the focus is on providing an introduction to the approach for fitting an FFM in R. Despite being self-contained, users may wish to download the code rather than copy-paste to run this themselves, and this can be found at **github.com/bsh2/thesis/c6/nls.R**.

The *optimx* package must be installed and loaded into R from CRAN by running the following code

```
install.packages("optimx")
library(optimx)
```

Using the objective functions developed so far with real data requires the user to import data into R and aligning it with the format described in Table 6.1. For the purposes of this example and to remain self-contained, synthetic data was developed via simulation (listing 6.8, Figure 6.8), including a set of training loads, and simulated performance data (function `simulateStandard`) under a set of manually specified parameters.

Listing 6.8 Synthetic data: Simulated performance values

```
# Specify some arbitrary loads
loads <- data.frame("day" = 0:100,
                    "load" = c(0, rep(c(1,1.2,0.5,1.8,2,0.25,0.7,0.9,0,0.5,1,
                                        0.8,1.2,1.3,0.9,0,0,2,1.1,0.5), 5)))

# Simulate the standard model to build some synthetic performance data
mockParameters <- c(100, 1, 22.5, 1.2, 8)       # c(p*, kg, Tg, kh, Th)

mockPerformances <- data.frame("day" = 1:100, "performance" =
                    simulateStandard(mockParameters, loads, returnObject = "performance"))

# Subset to reduce no. of datapoints
mockPerformances <- mockPerformances[seq(1, 100, 3),]
```

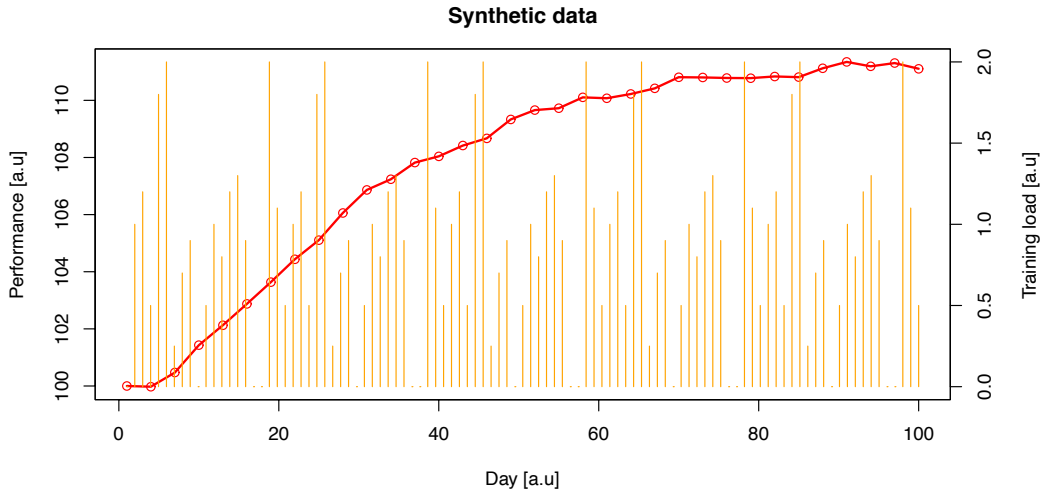


Figure 6.8: Synthetic data: Training load series (orange) and simulated performance values (red) (listing 6.8) for the reproducible example in section 6.2.1

Fitting the standard and VDR models to the synthetic data (listing 6.8, figure 6.8) is then completed via a simple function call to the 'wrapper' function (`optimx`) found in the optimisation package *optimx*. This function is a 'wrapper' in that it provides access to multiple algorithms (functions) in the package via the argument (`method`), shown next. Algorithms available in *optimx* include Nelder-Mead (a downhill simplex method), BFGS (a quasi-Newton method with optional contraints), a Newton-type algorithm, and a conjugate gradient method. These algorithms are discussed in section 6.2.4. Listing 6.9 below makes use of the objective functions developed (listings 6.1, 6.5) to fit the standard and VDR models to the simulated data via a NLS regression approach using the BFGS algorithm (Result plotted in figure 6.9).

Listing 6.9 Fitting the simulated data to the standard and VDR models under NLS via BFGS

```
library(optimx)                                          # Load the required package

# Start values for the optimisation algorithm
startValsStandard <- c(90, 0.8, 26, 1.5, 11)           # c(p*, kg, Tg, kh, Th)
startValsVDR <- c(90, 0.8, 26, 1.5, 11, 0.65)          # c(p*, kg, Tg, kh, Th, Th2)

# Fit the standard model via Nonlinear least-squares
standardFitted <-  optimx::optimx(par = startValsStandard,       # Starting values
                                  fn = standardObjectiveSS,        # RSS
                                  method = "BFGS",               # Method (see ?optimx)
                                  loads = loads,                 # Passed to fn (inputs)
                                  perfVals = mockPerformances)   # Passed to fn (target)

# Fit the VDR model via Nonlinear least-squares
vdrFitted <-  optimx::optimx(par = startValsVDR,          # Starting values
                             fn = vdrObjectiveSS,           # Objective function (RSS)
                             method = "BFGS",             # Method (see ?optimx)
                             loads = loads,               # Passed to fn (inputs)
                             perfVals = mockPerformances) # Passed to fn (target)
# Inspect results
standardFitted
vdrFitted
```

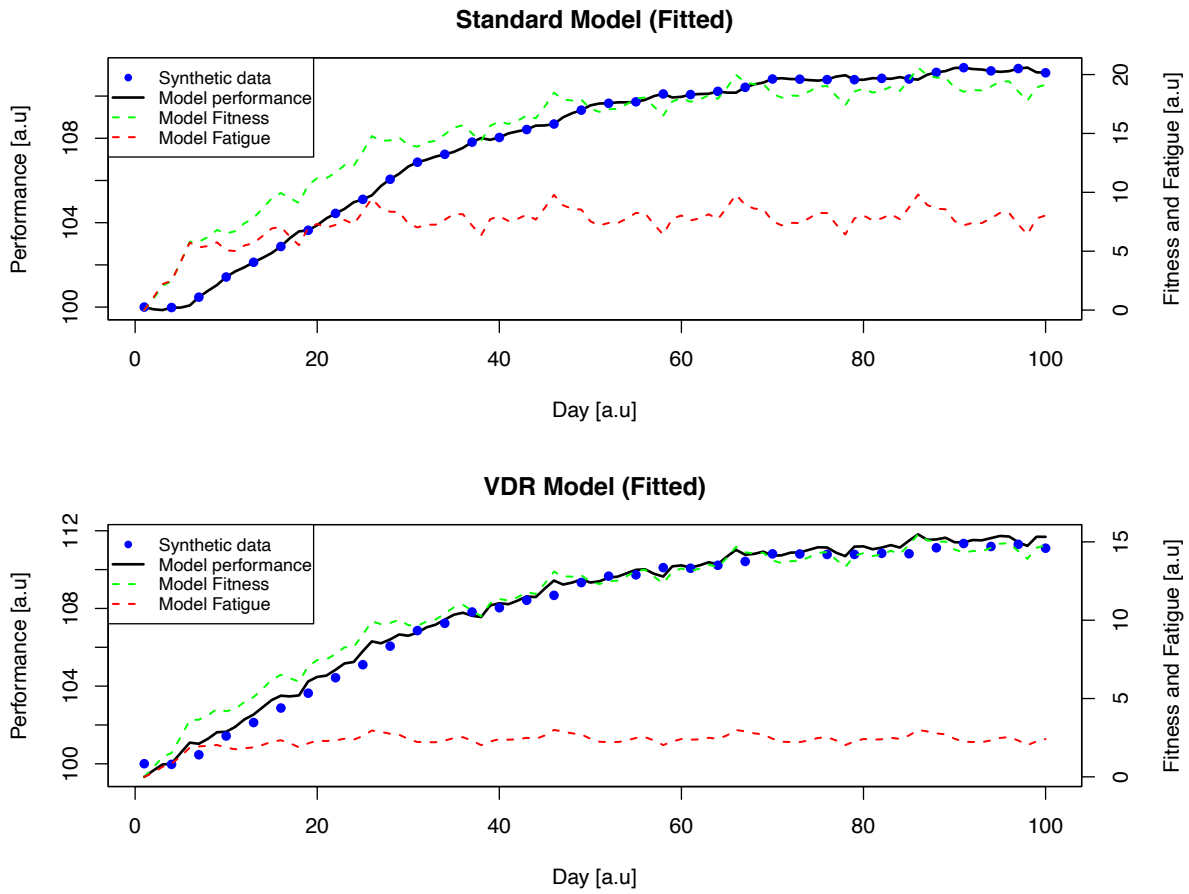


Figure 6.9: Standard and VDR FFMs fitted to the synthetic data via NLS under L-BFGS-B (listing 6.9). Code for the plots found in github.com/bsh2/thesis/c6/nls.R

6.2.2 A hidden loop approach (sapply) – Simulating the standard, fitness-delay, and VDR FFM and fitting via maximum likelihood estimation

This subsection begins by working toward developing an alternative simulation function for the standard model by means of a base R function from the *apply* family, equipping the reader with a contrasting approach to the explicit loop structures used in the previous subsection (6.2.1). The apply family of functions enables the user to perform repetitive action on data stored in various objects including arrays (e.g., data frames, matrices) and lists. These actions can range from simple calculations (e.g., multiplication), to complicated operations processed by user-defined R functions. If used correctly, the utilisation of the apply family of functions can improve the readability of code (particularly when nested loops are involved), facilitate parallelisation, and can be well suited to recursion (Wickham, 2019). However, it must be noted that the apply family is not the same as vectorisation, it is effectively loop-hiding with a for-loop in its definition, and has execution times roughly equal to an explicit for-loop (Burns, 2011). The primary benefit of the apply family of functions is improvement in the readability of code. In general, it is important that the control flow of any code is as simple as possible to achieve the task (within the bounds of not trying to overoptimise the code), and so that debugging is easier. Therefore, presentation of the apply approach in this subsection may be useful for some readers who are still struggling with the terse nature of the loop structures presented in the previous subsection. Inputs for this subsection remain the same as those described in section 6.2.1 (see table 6.1). The fitness-delay model adaptations are not demonstrated to avoid unnecessary repetition, as it is assumed readers are to be now aware of the ease with which the objective or prediction functions developed can be modified for use with the fitness-delay model. However, they are included in the associated code file for this subsection:

github.com/bsh2/thesis/c6/maximum_likelihood.R

The secondary aim of this chapter is to discuss the theory of and demonstrate maximum likelihood estimation for fitting FFMs to data. This section can represent a steeper learning curve to become acquainted with the control flow of the code associated with the theory of maximum likelihood compared to section 6.2.1 where a conceptually simpler NLS approach was examined.

The R function `sapply` is one function within the family of apply functions in base R. The function `sapply` takes as input a vector of values, and each element (of that vector) is then passed through another function (often with other inputs), returning the result from each iteration in as simple a data structure as possible (typically another vector or a data frame).

To begin to understand how the sapply function can be used in lieu of an explicit for-loop to implement FFMs, recall the general form of the FFM components (Busso, Candau and Lacour, 1994):

$$\sum_{i=1}^{t-1} \omega(i) \cdot e^{-\frac{t-i}{\tau}} \tag{6.7}$$

Next, considering what may be required to implement equation 6.7 in R, if a user sought to repeatedly compute this expression for iteratively increasing values of t (say, where t goes from $t = 1$ to T in time steps of 1 day). Based on the description given above for the sapply function it should be apparent that it this function a well placed approach for completing this task. Following this further, we first define an R function to compute equation 6.7 for any given value of t (listing 6.10).

Listing 6.10 General component convolution function

```
convolveTraining <- function(loads, tau){

  # Value of t relevant to (eq 6.9)
  dayt <- length(loads)

  # Note that loads[1:dayt] will yield c(w(0), w(1), ... , w(t-1))
  return(sum(loads[1:dayt] * exp(-(dayt:1 / tau))))
}
```

The function in listing 6.10 (convolveTraining) takes as input a vector of load values (loads) running from day 0 to day $t - 1$ and the relevant time decay constant (τ) for the component, and returns the sum expressed in equation 6.7 for a value of t implied by the length of the load vector supplied. Note that as before, the load vector is supplied with the first element as ω_0, and so for some day t we do not have to specify $t - 1$ at any point in the code as we naturally retain this condition. This function can now be used in conjunction with sapply to iteratively compute the component values for changing (increasing) t, as shown in the example in listing 6.11.

Listing 6.11 An example demonstrating how sapply can be used to iteratively compute eq. 6.7 (general component) for increasing t via the convolveTraining function in listing 6.10

```
base::sapply(1:T, function(t) convolveTraining(loads[1:t], tau))
```

In listing 6.11, sapply takes a numeric vector of length T, defined by c(1:T). In an iterative process, the value of t is updated to match the value of each element in this vector, and at each iteration of changing t the convolveTraining function is called with the input vector of load values cut to length t-1 (recall that ω_0 is the first element in the loads vector so loads[1:t,] naturally yields loads from day 0 to $t - 1$). Returned back to the sapply function at each iteration is the result of equation 6.7 for increasing t, and these results are then compiled and returned to the user as an ordered vector.

Now that this concept has been introduced, it should become clearer how a simulation function to compute equation 6.6 could be developed for both the standard and VDR models, that is now devoid of explicit loops (recall that loops are still used, they are just hidden by `sapply`). These functions are presented next, beginning with a simulation function for the standard model that uses an sapply approach.

Listing 6.12 Simulation function: Standard model – sapply approach

```
simulateStandard2 <- function(pars, loads, initialPars = c(0,0)){

  # Parameters supplied as: pars <- c(p*, k_g, Tau_g, k_h, Tau_h)

  # Ancillary function (required)
  # ------------------------------------------------------------------------
  convolveTraining <- function(loads, tau){

    # Value of t relevant to (eq 6.9)
    dayt <- length(loads)

    # Note that loads[1:dayt] will yield c(w(0), w(1), ... , w(t-1))
    return(sum(loads[1:dayt] * exp(-(dayt:1 / tau))))
  }
  # ------------------------------------------------------------------------

  # Length of the training load series (final day)
  T <- tail(loads$day, 1)

  # If initial parameters q_g and q_h are supplied (otherwise evaluates to 0)
  initialFitness <- initialPars[1] * exp(-(1:T) / pars[3])
  initialFatigue <- initialPars[2] * exp(-(1:T) / pars[4])

  # Calculate the fitness and fatigue effects (Utilizing sapply function)
  fitness <- pars[2] *
    base::sapply(1:T, function(t) convolveTraining(loads$load[1:t], pars[3]))

  fatigue <- pars[4] *
    base::sapply(1:T, function(t) convolveTraining(loads$load[1:t], pars[5]))

  performance <- pars[1] + initialFitness - initialFatigue + fitness - fatigue

  # Return model predicted performance, fitness, and fatigue
  return(data.frame(day = 1:T,
                    performance = performance,
                    fitness = fitness,
                    fatigue = fatigue,
                    load = loads$load[2:(T + 1)]))
}
```

Adapting the `standardPredict` function in listing 6.12 to accommodate the VDR model involves the requirement for an additional function (`kh2Compute`) (listing 6.13), to compute the additional variable gain term $k_{h_2}(i) = \sum_{j=1}^{i} \omega(j) \cdot e^{-(i-j)/\tau_{h_2}}$ within the fatigue component (see equations 6.3 and 6.4 from section 6.2.1). The `kh2Compute` function will be used later in conjunction with an additional `sapply` function call to compute $k_{h_2}(i)$ for changing values of i.

Listing 6.13	Variable gain term function $k_{h_2}(i)$ for the VDR model

```
1  kh2Compute <- function(loads, tau){
2    day_i <- length(loads)
3    return(sum(loads[1:day_i] * exp(-((day_i - 1):0 / tau))))
4  }
```

In listing 6.13, for any value of `day_i`, the computation `(day_i-1):0` will yield the resultant vector:

`((day_i-1), (day_i-2), …, 0)`

Which is equivalent to repeatedly computing $(i - j)$ for $j = (1, 2, \dots, i)$.

And so, in the listing 6.13 (line 3), the RHS of the expression:

`exp(-((day_i-1):0 / tau))`

yields the vector $(e^{-\frac{i-1}{\tau}}, e^{-\frac{i-2}{\tau}}, \dots, e^{-\frac{(i-i)}{\tau}})$.

Also recall that vector operations in R are performed on paired elements (by position in the vector) and so if performing the multiplication of two vectors, say `c(1,2,3) * c(2,3,4)`, the resultant will be the vector `c((1*2), (2*3), (3*4))`. It follows that the vector multiplication within listing 6.13 above will yield a vector representing repeated calculation of the expression:

$$\left(\omega(j) \cdot e^{-\frac{i-j}{\tau_{h_2}}}\right) \tag{6.8}$$

for values of $j = (1, \dots, i)$.

Therefore $k_{h_2}(i)$ is simply found by taking the `sum` of the elements in this vector. With these aspects in mind, the VDR model prediction function can be developed as follows in listing 6.14 using the functions from listing 6.13 (`kh2Compute`) and 6.10 (`convolveTraining`).

Listing 6.14 Simulation function: VDR model – sapply approach

```
1  simulateVDR2 <- function(pars, loads, initialPars = c(0,0)){
2
3    # Parameters supplied as: pars <- c(p*, k_g, Tau_g, k_h, Tau_h, Tau_h2)
4
5    # Ancillary functions (required)
6    # ------------------------------------------------------------------------
7    # Listing 6.10
8    convolveTraining <- function(loads, tau){
9
10     # Value of t relevant to (eq 6.9)
11     dayt <- length(loads)
12
13     # Note that loads[1:dayt] will yield c(w(0), w(1), ... , w(t-1))
14     return(sum(loads[1:dayt] * exp(-(dayt:1 / tau))))
15   }
16
17   # Listing 6.13
18   kh2Compute <- function(loads, tau){
19     day_i <- length(loads)
20     return(sum(loads[1:day_i] * exp(-((day_i-1):0 / tau))))
21   }
22   # ------------------------------------------------------------------------
23
24   # Length of the training load series supplied = final day (T)
25   T <- tail(loads$day, 1)
26
27   # If initial parameters q_g and q_h are supplied (otherwise evaluates to 0)
28   initialFitness <- initialPars[1] * exp(-(1:T) / pars[3])
29   initialFatigue <- initialPars[2] * exp(-(1:T) / pars[5])
30
31   # Compute modeled fitness
32   fitness <- pars[2] *
33     base::sapply(1:T, function(i) convolveTraining(loads$load[1:i], pars[3]))
34
35   # For each i=1:T, compute k_h_2(i) = sum_(j=0)^(i){w(j)e^(-(i-j)/tau_h_2)}
36   # s.t. k_h_2(t) = sum(k_h_2(i)) for i = 1,2,...,(t-1)
37   kh2 <- base::sapply(1:T, function(i) kh2Compute(loads$load[1:i], pars[6]))
38
39   # Compute modeled fatigue
40   fatigue <- pars[4] *
41     base::sapply(1:T, function(i) convolveTraining(kh2[1:i], pars[5]))
42
43   # Compute modeled performance
44   performance <- pars[1] + initialFitness - initialFatigue + fitness - fatigue
45
46   # Return model predicted performance, fitness, and fatigue
47   return(data.frame(day = 1:T,
48                     performance = performance,
49                     fitness = fitness,
50                     fatigue = fatigue,
51                     kh2 = kh2,
52                     load = loads$load[2:(T + 1)]))
53 }
```

To understand the key concepts in the implementation of the VDR prediction function in listing 6.14, note that line 37 applies the `sapply` function to repeatedly compute the function $k_{h_2}(i)$ for changing values of i from $i = 1$ to $T - 1$ (where T is the final day in the load series). The index i will implicitly run to $T - 1$ rather than T as the load values passed to the function `kh2Compute` by `sapply` begin at day zero ($\omega_0 = 0$). This concept was discussed at the start of section 6.2. Recall that the fatigue component of the VDR model computed for a given day t is also as follows:

$$k_h \sum_{i=1}^{t-1} k_{h_2}(i) \cdot e^{-\frac{t-i}{\tau_h}} \tag{6.9}$$

By the process described for the function in listing 6.13, the sum in equation 6.9 can then be computed by making use of vector multiplication in R. For a given day t, computing eq. 6.8 in R requires us to snip the resultant vector of length T obtained at line 18, to the required length t, giving us the vector of $k_{h_2}(i)$ values from $i = 1$ to $t - 1$. This vector is then multiplied with a vector containing the values of $e^{-\frac{t-i}{\tau_h}}$ for $i = 1$ to $t - 1$, and the sum of the elements in the resultant vector (scaled by k_h) yields equation 6.8 for t. Recall in listing 6.10 that the `convolveTraining` function was introduced, and note that if in that function if the values supplied to the loads argument were replaced with the obtained $k_{h_2}(i)$ values for $i = 1$ to $t - 1$, this would achieve exactly the required process just described. In essence, this represents the recursive aspect of the VDR model. To take this one step further, remember that the prediction function developed is concerned with being able to compute modelled fatigue for each day between days 1 and T in 1-day time-steps, and so the `sapply` function is utilised once again to repeat the above process for changing values of t ($t = 1,2, \dots, T$) (line 41).

Now that several of the most important concepts have been covered to help the reader understand the flow of the functions developed so far, and in particular the use of the `sapply` function, attention is turned to parameter estimation from data and in particular the theory behind a maximum likelihood approach. Similar to section 6.2.1, appropriate objective functions will be developed, and then applied at the end of the section within a reproducible example to fit the models to synthetic data.

First a general introduction to likelihood is given. For a set of observations (x), the probability density is known as the *joint probability density* if the independence assumption can be made, where the joint probability density is the product of the individual densities (Zieffler, 2019), described as:

$$p(x_1, x_2, \dots, x_n) = p(x_1) \cdot (x_2) \cdot \dots \cdot p(x_n) \tag{6.10}$$

The probability distribution of a normal distribution for a value x is defined as:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{6.11}$$

To compute the probability density in R (of x) from a normal distribution with mean μ (`mu`) and standard deviation σ (`sigma`) there is a simple function available in standard R (`dnorm`) that can be applied, as follows:

```
prod_density_x <- dnorm(x, mean = mu, sd = sigma)
```

The likelihood ($\mathcal{L}$) is defined as the probability of a particular set of parameters given the data and assumption that the data are from a particular distribution (Zieffler, 2019), described by:

$$\mathcal{L} = \mathrm{P}(\text{Parameters} \mid \text{Distribution \& Data})$$

For example, given a set of observed data $X = (x_1, x_2, x_3, x_4)$ assumed to come from a normal distribution, a researcher might be interested in answering what the likelihood (probability) is that the mean (of that distribution) is a ($\mu = a$) and standard deviation is b ($\sigma = b$). Computing this likelihood is just the case of computing the joint probability density of the data under the parameters ($\mu = a$, $\sigma = b$). In R this is then achieved as follows:

```
Likelihood_X <- prod(dnorm(c(x1, x2, x3, x4), mean = a, sd = b))
```

Maximum likelihood is concerned with which values of μ and σ are most likely to generate the observed data X, for example "is (a, b) the best, or some other values, say (c, d)?". This is the main concept in maximum likelihood estimation (MLE) (Zieffler, 2019). In MLE, the goal is to find the set of parameters (μ, σ) that maximises the likelihood given the data and a distribution (Zieffler, 2019). This could be done using brute force approaches such as a grid search that compare a handful of different possible values, or via optimisation algorithms that methodically search the parameter space. Often, likelihood values are quite small due to the multiplication of multiple probabilities and so taking the natural logarithm of the likelihood can help alleviate this issue and can become important when applying optimisation algorithms as excessively small values can cause early convergence to the wrong solution (due to changes in the objective function becoming smaller than tolerance). Assuming normality and independence, this is written symbolically as follows:

$$\mathcal{L}(\text{parameters} \mid \text{data}) = p(x_1) \cdot p(x_2) \cdot \ldots \cdot p(x_n) \tag{6.12}$$

Such that:

$$\begin{aligned} \ell(\text{parameters} \mid \text{data}) &= \ln\big(\mathcal{L}(\text{parameters} \mid \text{data})\big) = \ln\big(p(x_1) \cdot p(x_2) \cdot \ldots \cdot p(x_n)\big) \\ &= \ln\big(p(x_1)\big) + \ln\big(p(x_2)\big) + \cdots + \ln\big(p(x_n)\big) \end{aligned} \tag{6.13}$$

From equation 6.13, it is clear that the log-likelihood is the sum of the log-transformed densities. In R the `dnorm` function provides an additional argument for this (`log = TRUE`), such that:

```
Log_likelihood_X <- sum(dnorm(c(x1, x2, x3, x4), mean = a, sd = b, log = TRUE))
```

Where maximisation of log-likelihood yields the same parameters as maximising the likelihood (Zieffler, 2019). When fitting models, it is the residual ($measured - modelled$) values that are of interest to us, and the values on which we place the distributional assumptions (Zieffler, 2019). The goal in fitting FFMs via maximum log-likelihood estimation is to find the set of model parameters θ that maximise the log-likelihood for a set of residuals (ϵ) that come from a normal distribution defined by:

$$\epsilon \text{ are } i.i.d\ N{\sim}(\mu, \sigma^2) \tag{6.14}$$

Where residuals (errors) are assumed to be independent and identically distributed. For the purposes of fitting the standard and VDR models in this section we assume the mean of the normal distribution on the errors to be $\mu = 0$, leaving a sixth parameter σ to be estimated when fitting the standard model, or seventh for the VDR (excluding initial component parameters). Such that the log likelihood is given by:

$$\ell(\theta \mid \text{data}) = \ln\big(p(\epsilon_1) \cdot p(\epsilon_2) \cdot \ldots \cdot p(\epsilon_n)\big) \tag{6.15}$$

Where,

$$\epsilon_i = (p_i - \hat{p}_i), i = (1,2, \ldots, n) \tag{6.16}$$

In R, under equation 6.11, this can be computed as follows by utilising the `dnorm` function (Assuming the two vectors of (`modelled`) and (`measured`) values have already been obtained), as follows:

```
errors <- (modelled - measured)
Log_likelihood_errors <- sum(dnorm(errors, mean = 0, sd = sigma, log = TRUE))
```

Given the concepts described so far for MLE, the reader should now have enough awareness to follow the R functions in listing 6.15 and 6.16, that defines a negative log-likelihood objective function for the standard model (with or without initial components) and VDR model, respectively. This objective function computes the log likelihood of the residuals for the parameters θ, including σ. A data fitting process is demonstrated at the end of this subsection in a self-contained reproducible example. Also to note in listing 6.15 and 6.16, the negative log-likelihood is returned, as the optimisation package *optimx* (and the majority of others in R) minimise the objective function by default, and minimisation of a negative function is equivalent to maximisation (Nash, 2014). However, the logical argument `maximise` is included in the function (false by default) in the case that a maximiser is used in which case if `maximise = TRUE`, the log likelihood is returned (and maximised by the algorithm).

Listing 6.15 | Objective function (log likelihood): Standard model – sapply approach

```
1  standardObjectiveLL <- function(pars, loads, perfVals, initial = FALSE,
2                                  maximise = FALSE){
3
4    # INPUT NOTES:
5    # ------------------------------------------------------------------------
6    #         [1] [2] [3] [4] [5] [6]    [7] [8]
7    # Pars: c(p*, kg, Tg, kh, Th, sigma)                  initial = FALSE
8    # Pars: c(p*, kg, Tg, kh, Th, sigma, qg, qh)          initial = TRUE
9    # ------------------------------------------------------------------------
10
11   # Ancillary function (required)
12   # ------------------------------------------------------------------------
13   convolveTraining <- function(loads, tau){
14
15     # Value of t
16     dayt <- length(loads)
17
18     # Note that loads[1:dayt] will yield c(w(0), w(1), ... , w(t-1))
19     return(sum(loads[1:dayt] * exp(-(dayt:1 / tau))))
20   }
21   # ------------------------------------------------------------------------
22
23   finalMeasurement <- tail(perfVals$day, 1)
24
25   if (initial == TRUE){
26     initFitness <- pars[7] * exp(-(1:finalMeasurement) / pars[3])
27     initFatigue <- pars[8] * exp(-(1:finalMeasurement) / pars[5])
28   }
29
30   # Compute modeled performance from t=1 to t=finalMeasurement
31   fitness <- pars[2] * sapply(1:finalMeasurement,
32                               function(t) convolveTraining(loads$load[1:t], pars[3]))
33   fatigue <- pars[4] * sapply(1:finalMeasurement,
34                               function(t) convolveTraining(loads$load[1:t], pars[5]))
35
36   if (initial == FALSE){
37     performance <- pars[1] + fitness - fatigue
38   }
39
40   if (initial == TRUE){
41     performance <- pars[1] + initFitness - initFatigue + fitness - fatigue
42   }
43
44   # Extract modeled performance values on days where measurement exists
45   performance <- performance[perfVals$day]
46
47   # Compute errors
48   errors <- perfVals$performance - performance
49
50   if (maximise == FALSE){
51     return(-1.0 * sum(dnorm(errors, mean = 0, sd = pars[6], log = TRUE)))
52   }
53   if (maximise == TRUE){
54     return(sum(dnorm(errors, mean = 0, sd = pars[6], log = TRUE)))
55   }
56 }
```

Notes (listing 6.15):

- The length and order of the argument `pars` supplied will differ depending on whether `initial = TRUE` or `initial = FALSE` (default) – See lines 4-9

Listing 6.16 | Objective function (log likelihood): VDR model – sapply approach

```
1  vdrObjectiveLL <- function(pars, loads, perfVals, initial = FALSE,
2                             maximise = FALSE){
3
4    # INPUT NOTES:
5    # ----------------------------------------------------------------
6    #         [1] [2] [3] [4] [5] [6]  [7]    [8] [9]
7    # Pars: c(p*, kg, Tg, kh, Th, Th2, sigma)          initial = FALSE
8    # Pars: c(p*, kg, Tg, kh, Th, Th2, sigma, qg, qh)  initial = TRUE
9    # ----------------------------------------------------------------
10
11   # Ancillary functions (required)
12   # ------------------------------------------------------------------------
13   convolveTraining <- function(loads, tau){
14
15     # Value of t
16     dayt <- length(loads)
17
18     # Note that loads[1:dayt] will yield c(w(0), w(1), ... , w(t-1))
19     return(sum(loads[1:dayt] * exp(-(dayt:1 / tau))))
20   }
21
22   kh2Compute <- function(loads, tau){
23     day_i <- length(loads)
24     return(sum(loads[1:day_i] * exp(-((day_i-1):0 / tau))))
25   }
26   # ------------------------------------------------------------------------
27
28   finalMeasurement <- tail(perfVals$day, 1)
29
30   if (initial == TRUE){
31     initFitness <- pars[8] * exp(-(1:finalMeasurement) / pars[3])
32     initFatigue <- pars[9] * exp(-(1:finalMeasurement) / pars[5])
33   }
34
35   # Compute modeled performance from t=1 to t=finalMeasurement
36   fitness <- pars[2] * sapply(1:finalMeasurement,
37                               function(t) convolveTraining(loads$load[1:t], pars[3]))
38   kh2 <- sapply(1:finalMeasurement, function(i) kh2Compute(loads$load[1:i], pars[6]))
39   fatigue <- pars[4] * sapply(1:finalMeasurement,
40                               function(t) convolveTraining(kh2[1:t], pars[5]))
41
42   if (initial == FALSE){
43     performance <- pars[1] + fitness - fatigue
44   }
45
46   if (initial == TRUE){
47     performance <- pars[1] + initFitness - initFatigue + fitness - fatigue
48   }
49
50   # Extract modeled performance values on days where measurement exists
51   performance <- performance[perfVals$day]
52
53   # Compute errors
54   errors <- perfVals$performance - performance
55
56   if (maximise == FALSE){
57     return(-1.0 * sum(dnorm(errors, mean = 0, sd = pars[7], log = TRUE)))
58   }
59   if (maximise == TRUE){
60     return(sum(dnorm(errors, mean = 0, sd = pars[7], log = TRUE)))
61   }
62 }
```

Notes (listing 6.16):

- The length and order of the argument `pars` supplied will differ depending on whether `initial = TRUE` or `initial = FALSE` (default) – See lines 4-9

Fitting an FFM to data via maximum likelihood estimation (MLE) (A self contained example)

To finish this subsection, a self-contained example is provided for fitting an FFM to simulated performance data similar to that generated at the end of subsection 6.2.1, but the addition of noise (listing 6.17). The transformed data is fitted to both the standard and VDR models, from the functions developed so far in this subsection (listings 6.15, 6.16). As in the previous subsection, the L-BFGS-B algorithm is used to estimate the model parameters, from the package *optimx* (Nash *et al.*, 2020). The reader may find it helpful to work through this example in their own development environment to cement their understanding. This example can also be found within the code at **github.com/bsh2/thesis/c6/maximum_likelihood.R**

Listing 6.17 Synthetic data developed by simulation of the VDR model under hypothetical training loads

```
# Synthetic load data
loads <- data.frame("day" = 0:100,
                    "load" = c(0, rep(c(1, 1.2, 0.5, 1.8, 2, 0.25, 0.7, 0.9, 0, 0.5,
                                        1, 0.8, 1.2, 1.3, 0.9, 0, 0, 2, 1.1, 0.5), 5) ) )

# Using the standard model to build the synthetic performance data
mockParameters <- c(100, 1, 22.5, 1.2, 8, 0.5)              # c(p*, kg, Tg, kh, Th, Th2)
mockPerformances <- simulateVDR2(mockParameters, loads)[,1:2]
mockPerformances_noerror <- mockPerformances

# Add some random error from a Gaussian distribution
sigma <- 1
set.seed(1)
mockPerformances$performance <- mockPerformances$performance + rnorm(100, 0, sigma)

# Subset to reduce number of data points
mockPerformances <- mockPerformances[seq(1, 100, 3),]
```

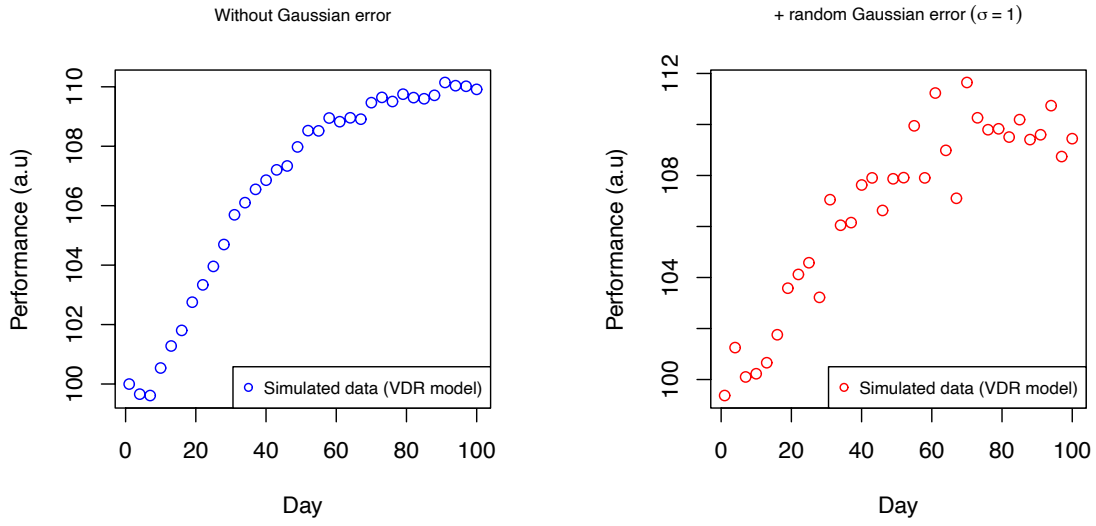


Figure 6.10: Simulated performance values (via VDR model) for the synthetic data (listing 6.17), with and without random Gaussian error added. Parameters ($p^* = 100, k_g = 1, \tau_g = 22.5, k_h = 1.2, \tau_h = 8, \tau_{h_2} = 1.2$)

Lastly in this subsection, code and output is presented for fitting the standard and VDR models to the synthetic data in listing 6.17, including estimation of error variance σ^2. The process again utilises the `optimx` function however this time enforcing bounding on the BFGS algorithm unlike in listing 6.9 (L-BFGS-B algorithm) to compute the parameters under a maximum-likelihood estimation approach (listing 6.18). This algorithm is discussed in depth in subsection 6.2.4. The results of listing 6.18 are presented in Figure 6.11.

Listing 6.18 Fitting the standard and VDR models to the simulated data (listing 6.17) under MLE via L-BFGS-B

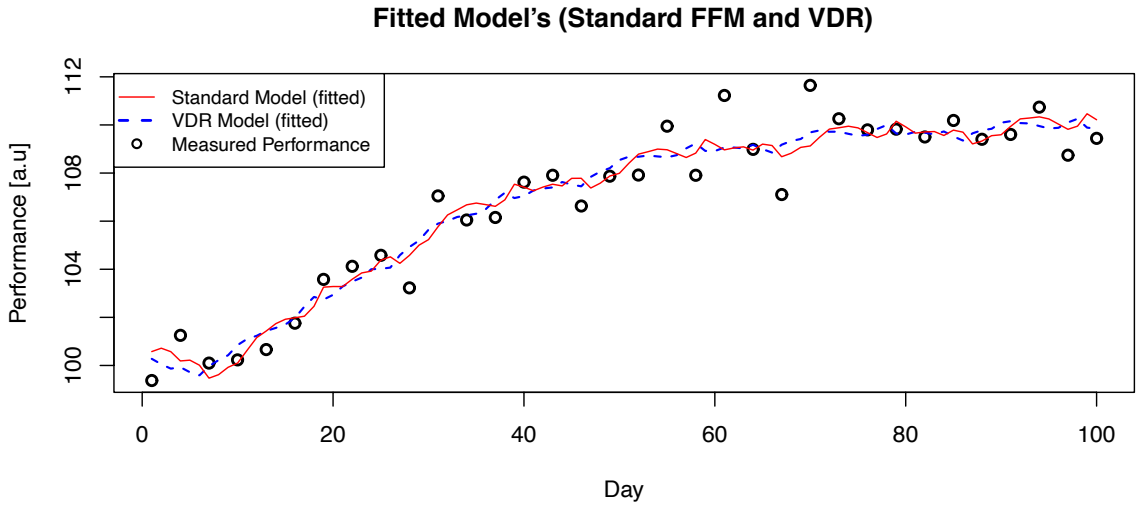
```
# Standard: c(p*, kg, Tg, kh, Th, sigma) | VDR: c(p*, kg, Tg, kh, Th, Th2, sigma)
startPars <- list("standard" = c(95, 0.85, 26, 1.4, 5, 0.6),
                  "vdr" = c(95, 0.85, 26, 1.4, 5, 1, 1.3))

# Fit the standard model
standardFitted <- optimx(par = startPars[["standard"]],
                         fn = standardObjectiveLL,
                         lower = c(60, 0.1, 1, 0.1, 1, 0.1),
                         upper = c(200, 3, 50, 3, 50, 10),
                         method = "L-BFGS-B",
                         loads = loads,
                         perfVals = mockPerformances,
                         control = list(maxit = 10000))

# Fit the VDR model
vdrFitted <- optimx(par = startPars[["vdr"]],
                    fn = vdrObjectiveLL,
                    lower = c(60, 0.1, 1, 0.1, 1, 0.1, 0.1),
                    upper = c(200, 3, 50, 3, 50, 5, 10),
                    method = "L-BFGS-B",
                    loads = loads,
                    perfVals = mockPerformances,
                    control = list(maxit = 10000))
```

Notes (listing 6.18):

- `maxit` is a control parameter for `optimx` for the number of iterations (objective function calls)
- Other option control arguments for the optimiser are available including arguments to scale (e.g., normalise) the parameter values (`parscale`), receive tracing information during the optimisation process (`trace`), and to control the convergence of the "L-BFGS-B" method based on reduction of the objective function (`factr`). See associated help files for further information (run `?optimx` in R).
- The use of the L-BFGS-B algorithm does not reflect a recommendation for research as an appropriate solver for the optimisation problem, and is merely used here as an introduction to optimisation in R. See chapter 5 for discussion of the issues surrounding the use of the L-BFGS-B algorithm with the standard and fitness-delay models. When incorporating an additional parameter σ and the additional parameters of the VDR model and optional initial components, it is expected that this algorithm will perform even worse. Further research is required in this area, and discussion of available optimisation algorithms in R can be found in section 6.2.6 further on in this chapter.
- To estimate initial components, adapt the `optimx` function call as follows:
    - The starting parameter vector supplied to par now needs to be of length and order
        - $c(p^*, k_g, \tau_g, k_h, \tau_h, \sigma, q_g, q_h)$ for the standard model
        - $c(p^*, k_g, \tau_g, k_h, \tau_h, \tau_{h_2}, \sigma, q_g, q_h)$ for the VDR model
    - Supply the additional argument `initial = TRUE` (passed to `fn`)
    - Adapt the lower and upper bound vectors to incorporate the additional parameters, in the order as shown above



	p^*	k_g	τ_g	k_h	τ_h	τ_{h_2}	σ	$-\ell$	ℓ	$\mathcal{L}/n$
Standard	100.3	2.68	17.1	3.00	11.6	-	0.96	47.2	-47.2	0.25
VDR	100.5	0.92	22.3	1.33	1.7	5	0.92	45.3	-45.3	0.26

$-\ell$ (negative log likelihood, i.e., the objective function value at the solution),

ℓ (log likelihood), $\mathcal{L}/n$ (average likelihood)

Figure 6.11: Models estimated from simulated data (listing 6.17) via MLE (listing 6.18)

6.2.3 Numerical approaches for solving the underlying ODE system and fitting to data

This section follows the development of code to numerically solve the standard model system in its ordinary differential equation (ODE) form, and then demonstrates how this process can then be enveloped by a data fitting framework (nonlinear least-squares). A numerical integration approach is useful when working with ODE models where obtaining a solution is difficult or not analytically (symbolically) tractable, such as for the non-linear FFM proposed by Turner *et al.* (2017). As the original FFM (Banister *et al*., 1975) is a first-order system of ordinary differential equations (chapter 2, equations 2.2 - 2.4), it is an initial value problem (IVP) with initial conditions $g(t_0) = g_0,\ h(t_0) = h_0,\ p(t_0) = p_{t_0}$, and can also be solved by numerical methods. When wrapped around by a data-fitting framework these initial conditions are also estimated without the need for further components. Various numerical methods exist for solving first-order IVPs, and in R the package *deSolve* (Soetaert and Petzoldt, 2010) provides access to many of these integrators from both the linear multistep (e.g., Adam's methods, backward differentiation (BDF) methods) and Runge-Kutta (e.g., Euler's method) categories. The package also provides methods from the Livermore family such as the default lsoda algorithm, as used in the implementations later on, that can automatically switch between stiff (BDF) and non-stiff (Adams) methods, dynamically monitoring data and deciding which method to use, meaning the user does not have to determine whether the problem is stiff or not. To engage with the resource the user does not need to know this level of detail however, it is provided for the more

advanced reader. The purpose of this section is to illustrate the process of numerically solving an ODE systems model in R, and fitting the model parameters to data. However, in-depth discussion of integrator choice or numerical methods is an area out with the scope of this thesis, and readers are directed to other works (Lambert, 1991). Recommendations to try from the *deSolve* package therefore include the default lsoda algorithm, or possibly Euler's method for the standard FFM. However, the choice of solver will depend on the problem at hand and is an area of study best approached with help from experts (applied mathematicians). Anecdotally, testing the subsequent standard model implementation indicated little to no differences in solutions obtained between the lsoda and Euler solver.

As a first step, the user must install and load the *deSolve* package from CRAN. Next, the system of differential equations is defined as a seperate R function (`banisterSystem`) that computes the derivatives g', h' in the ODE system (i.e., the model definition) according to the independent variable (e.g., time t) and load ω_t (Soetaert, Petzoldt and Setzer, 2010). This function is later passed as an argument in a main loop that iteratively calls the ODE function from the *deSolve* package for each time t, and as such Soetaert, Petzoldt and Setzer (2010) state that it:

> "*Must be defined as*
>
> `function(t, y, parms, ...)`
>
> *where* (`t`) *is the actual value of the independent variable (e.g., the current time point in the integration),* (`y`) *is the current estimate of the variables in the ODE system,* `parms` *is the parameter vector, and* (…) *can be used to pass additional arguments to the function*"

In listing 6.19 below, the additional argument (…) is used to pass the object ω_t (the current training load value at time t). The return value (output) of the function `banisterSystem` is a list containing a vector `r` that holds the derivatives of the state variables with respect to `t`.

Listing 6.19 Defining the standard model ODE system as an R function

```
1  banisterSystem <- function(t, y, parms, currentLoad){
2    r = c()
3    r[1] <- (parms[2]*currentLoad) - ((1/parms[3]) * y["G"])
4    r[2] <- (parms[4]*currentLoad) - ((1/parms[5]) * y["H"])
5    return(list(r))
6  }
```

In listing 6.20 on the following page, a function `banisterPredict` is defined, that takes as input a set of model parameters and load values, numerically solving the IVP specified by `banisterSystem` via the *deSolve* package. At each iteration of the loop at lines 9-41, the estimates of the state variables y are initialised at the value of the previous iteration y_{t-1} and the model solved numerically. At time $t < 2$, these values are simply initialised at the conditions g_0, h_0 (which also reflect the initial conditions to be fitted as parameters within the model fitting routine covered shortly).

Listing 6.20	Applying a numerical integrator to develop a model simulation function from the original system of ODEs (standard model)

```
banisterSimulate <- function(parms, loads){

  dat <- loads
  dat$G <- c(rep(0, length(loads$day)))
  dat$H <- c(rep(0, length(loads$day)))
  dat$pHat <- c(rep(0, length(loads$day)))

  # Solve model numerically at each time point (j)
  for (j in 1:length(dat$day)){
    currentLoad <- dat$load[j]
    if (j < 3){
      stateInit <- c(G = parms[6],     # Fitness (initial condition)
                     H = parms[7])     # Fatigue (initial condition)
    } else{
      # Initialize based on previous value (j-1)
      stateInit <- c(G = dat$G[j-1],    # Fitness on previous day
                     H = dat$H[j-1])    # Fatigue on previous day
    }

    # Vector of time-steps up to next day (j+1)
    t <- 0:1

    # Solve model for current time point (j)
    out = ode(y = stateInit, times = t, func = banisterSystem, parms = parms,
              method = c("lsoda"), currentLoad = currentLoad)

    # Extract solutions (j)
    if (j < 3){
      dat$G[j] <- unname(out[1,2])
      dat$H[j] <- unname(out[1,3])
    } else{
      dat$G[j] <- unname(out[2,2])
      dat$H[j] <- unname(out[2,3])
    }

    # Modelled performance at time j
    dat$pHat[j] <- parms[1] + dat$G[j] - dat$H[j]

  } # Update loop (j+1) and repeat
  return(dat)
}
```

Notes (listing 6.20):

- The length and order of `parms` supplied in the function call is as follows:
  - c($p^*, k_g, \tau_g, k_h, \tau_h, g_0, h_0$)
- The argument `loads` is as before in Table 6.1

Enveloping the simulation function in 6.20 into an NLS fitting framework is then rather straightforward and can be accomplished as follows:

Listing 6.21	RSS wrapper to facilitate NLS estimation of FFM system parameters and initial conditions

```
banisterObjective <- function(parms, perfVals, loads){
  dat <- banisterSimulate(parms, loads)
  datSubset <- dat[c((perfVals$day) + 1), ]   # +1 due to day zero first row
  RSS <- sum((datSubset$pHat - perfVals$performance)^2)
return(RSS)
}
```

The function `banisterObjective` in listing 6.21 can then be used to solve fit the ODE system in the original FFM to data (from listing 6.17) as shown in listing 6.22. This is a slightly verbose approach given the closed form approximation for the standard model exists and as shown earlier is simple to work with (Busso *et al.*, 1992). However, when it comes to the nonlinear ODE system (Turner *et al.*, 2017), this is the only way to fit the FFM as the analytic solution is intractable. Hence, the purpose of this section so far has been to set the reader up with a basic awareness of the approach necessary to fit an FFM ODE system, before leading into the nonlinear system which we will treat next.

Listing 6.22 Solving the original ODE system by numerical methods and fitting it to data (from listing 6.17) via NLS solver (L-BFGS-B)

```
# Create some starting values for the search c(p*,kg,Tg,kh,Th,g0,h0)
startParameters <- c(100, 0.9, 26, 1.2, 5, 0, 0)

# Bounds <- c(p*,kg,Tg,kh,Th,g0,h0)
fittedModel <- optimx(par = startParameters,
                      fn = banisterObjective,
                      method = "L-BFGS-B",
                      lower = c(50, 0.1, 1, 0.1, 1, 0, 0),   # Lower bounds
                      upper = c(150, 3, 50, 3, 50, 20, 20),  # Upper bounds
                      perfVals = mockPerformances,           # Same data as listing 6.17
                      loads = loads)
```

In 2017, Turner and colleagues proposed a non-linear variant of the original ODE system:

$$p(t) = p^* + k_g \cdot g(t) - k_h \cdot h(t) \tag{6.17}$$

$$g'(t) = \omega(t) - \frac{1}{\tau_g} g(t)^{\alpha} \tag{6.18}$$

$$h'(t) = \omega(t) - \frac{1}{\tau_h} h(t)^{\beta} \tag{6.19}$$

where α, β are power terms that represent the model's nonlinearities. The standard model system can be recovered by $\alpha = \beta = 1$. Adjusting the functions developed so far (listings 6.19-6.21) to numerically solve and fit this nonlinear systems model is straightforward and involves only minor modifications presented on the following page. Code from this section is available to download directly from **github.com/bsh2/thesis/c6/banister_and_turner.R**

Modification 1) At lines 3 and 4 in listing 6.19, with listing 6.23

Listing 6.23 Adapting the `banisterSystem` function (listing 6.19, lines 3-4) for the nonlinear system

```
r[1] = (parms[1]*currentLoad) -  ((1/parms[3]) * y["G"]^(parms[8]))
r[2] = (parms[2]*currentLoad) -  ((1/parms[4]) * y["H"]^(parms[9]))
```

Where `parms[8]` and `parms[9]` are the elements containing the values of α, β respectively

Modification 2) At line 2 in the example data-fitting code (listing 6.22), adjustment needs to made to the starting parameter vector and bounds to incorporate the additional parameters α, β at positions 8 and 9 in the vectors. i.e., `parms <- c(`$p^*, k_g, \tau_g, k_h, \tau_h, g_0, h_0, \alpha, \beta$).

Modification 3) Apart from simple cosmetic changes to the rest of the code, such as renaming of the functions (e.g., swapping `banisterSystem` for `turnerSystem`) and so forth, there are no other functionally important changes required to incorporate the non-linear model system from Turner *et al.* (2017).

Code is now presented to replicate several of the plots from Turner *et al.* (2017), including those that demonstrate saturation of training load effects, and the negative implications of excessive loads (i.e., overtraining). The authors considered the special case of constant daily training load to illustrate these concepts, and analytically obtained an equation for steady state performance:

$$\tilde{p} = p^* + \left(k_g \cdot \tau_g \cdot \omega\right)^{\frac{1}{\alpha}} - \left(k_h \cdot \tau_h \cdot \omega\right)^{\frac{1}{\beta}} \tag{6.20}$$

Turner *et al*. (2017) report two key values of interest that can be derived from this special case: 1) the constant training load that causes maximum performance (denoted $\omega_{optimal}$); and 2) The value at which a training load would result in no improvement in performance (denoted ω_{max}). Where:

$$\omega_{optimal} = \left(\left(\frac{\beta}{\alpha}\right)^{\alpha\beta} \frac{\left(k_g\tau_g\right)^{\beta}}{\left(k_h\tau_h\right)^{\alpha}}\right)^{\frac{1}{\beta-\alpha}} \tag{6.21}$$

$$\omega_{max} = \left(\frac{\left(k_h\tau_h\right)^{\alpha}}{\left(k_g\tau_g\right)^{\beta}}\right)^{\frac{1}{\beta-\alpha}} \tag{6.22}$$

Although equation 6.21 is written above exactly as it appears in Turner *et al.* (2017) it contains minor errors in the first and second fraction. These errors are not reflected in the plots within their study implying a simple reporting error that can be corrected here. The correct formulae for $\omega_{optimal}$ is as follows:

$$\omega_{optimal} = \left(\left(\frac{\alpha}{\beta}\right)^{\alpha\beta} \frac{\left(k_h\tau_h\right)^{\alpha}}{\left(k_g\tau_g\right)^{\beta}}\right)^{\frac{1}{\beta-\alpha}} \tag{6.23}$$

In their study, the authors chose a set of parameters and initial conditions that adequately simulated these concepts, however these were chosen arbitrarily to produce a realistic response, and model parameters are conceptually dependent on an athletes individual characteristics (Turner *et al.*, 2017).

The parameters used by the authors were: $(p^* = 155, k_g = 0.10, k_h = 0.12, \tau_g = 61, \tau_h = 5.5, g_0 = 70.9, h_0 = 24.5, \alpha = 1.16, \beta = 0.85)$. Three R functions in listing 6.24 below can be used for computing the values of these three formulae (equations 6.20, 6.22, 6.23) under the relevant parameter values and constant model input (ω).

Listing 6.24 R functions to compute $\tilde{p}$ (eq. 6.20), $\omega_{optimal}$ (eq. 6.23), and ω_{max} (eq. 6.22) for the special case of the Turner model system under constant load

```
# The training load that causes optimal (max) performance under constant load
w_optimal <- function(alpha, beta, kg, Tau_g, kh, Tau_h){
 out <- (((alpha/beta)^(alpha * beta)) *
         (((kh * Tau_h)^(alpha)) / ((kg * Tau_g)^(beta))))^(1/(beta-alpha))
 return(out)
}

# The training load value that would result in no performance improvement
w_max <- function(kg, Tau_g, kh, Tau_h, alpha, beta){
  return((((kh*Tau_h)^(alpha)) /
            ((kg*Tau_g)^(beta)))^(1/(beta-alpha)))
}

# Steady state performance under constant load
p_tilde <- function(p_0, kg, Tau_g, kh, Tau_h, alpha, beta, w){
  return(p_0 + (kg*Tau_g*w)^(1/alpha) - (kh*Tau_h*w)^(1/beta))
}
```

In figure 1 of Turner *et al.* (2017), steady state performance is plotted for increasing values of constant load ($\omega = 0 - 750$), to demonstrate saturation of load and over-training (whereby larger loads than ω_{max} result in a decrease in $\tilde{p}$. The values of $\omega_{optimal}$ and ω_{max} are also indicated on the plot. In figure 2 of Turner *et al.* (2017), the diminishing returns of load on fitness and compounding effects of fatigue are demonstrated under the steady state model described. Both plots are easily reproduced in R from the functions developed in listing 6.24 (Figures 6.12 and 6.13). The code associated with Figures 6.12 and 6.13 can be found in the code file associated with this chapter.

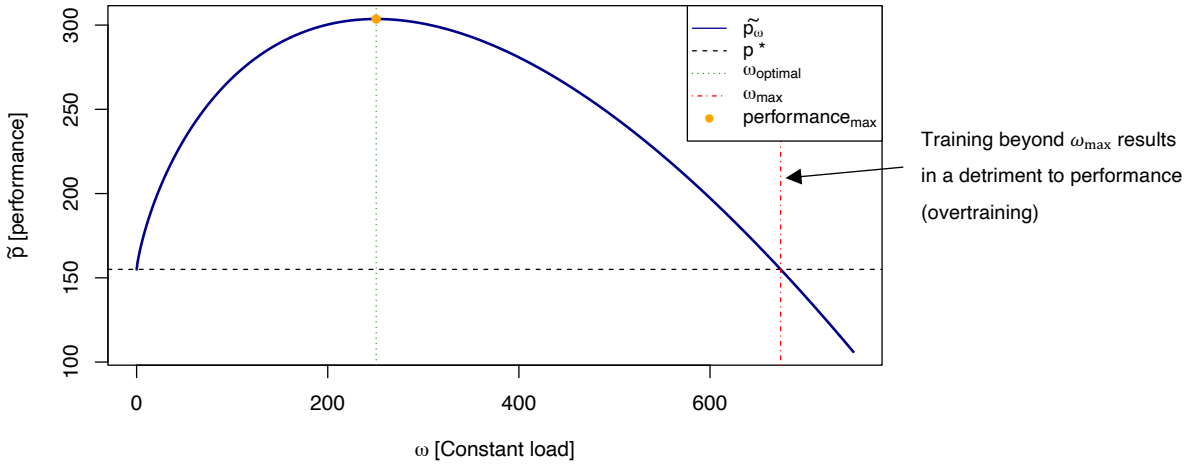


Figure 6.12: Replicating figure 1 in Turner *et al.* (2017), demonstrating saturation and overtraining model behaviour (reflected as increasing constant load on steady state performance)

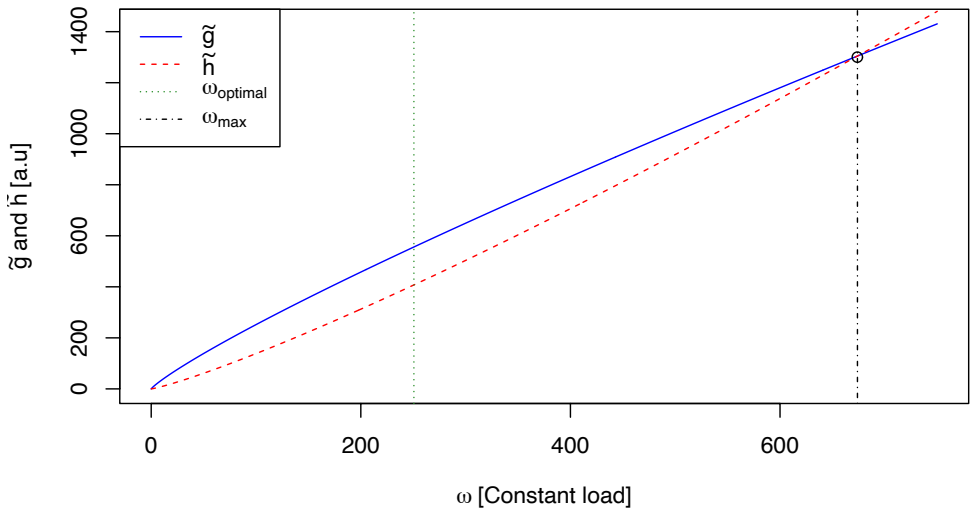


Figure 6.13: Replicating figure 2 in Turner *et al.* (2017), demonstrating the compounding effects of fatigue and diminishing returns of fitness with increasing constant load. Also shown is the point at which fatigue overwhelms fitness under the steady state model.

However, this analysis only considers the unrealistic case of constant daily training load, and even in this steady-state model, small changes in a single model parameter (whilst keeping others constant) can result in large changes in model behaviour with respect to these concepts such as saturation and overtraining. Furthermore, with the inclusion of the power terms (α, β), relationships between model parameters are unclear, and in particular how changes in these power terms (either alone, or together) affect model behaviour when the remainder of parameters are kept constant, and visa-versa. Practically, this is important as it may influence the difficulty of the optimisation problem, reducing the ability to find the appropriate solution (i.e., the global minimum of the objective function). One approach to investigating this in the future is applying a derivative field method and represents an area for future work. Of clear relevance to the purpose of this chapter and in particular sections 6.2.4 and 6.2.5, we will briefly consider the model fitting approach used by Turner *et al.* (2017) as it reflects a helpful start to addressing the problem of insufficient model evaluation processes in prior literature. When fitting the model, Turner *et al.* (2017) used a cross-validation approach to estimate overfitting and generalisation in the experimental portion of the work (i.e., fitting the model to real-world data).

Cross-validation approach from Turner *et al.* (2017)

1. Of the 18 real-world performance measurements (trials), 9 were randomly selected and their model was fitted via a genetic algorithm (*tuning parameters*: tournament selection, BLX-α crossover, Gaussian mutation).
2. This process of random trial selection and fitting was repeated 333 times in total. Giving an estimation of the distribution of all possible sets of 9 unique trials and yielding a total of 333 fitted parameter sets to different subsets of the data.
3. Using the sets of (sets of) parameters found from fitting the randomly selected trials (333 sets), model predictions were generated for each over the entire data period, and these were compared with the full series of performance measurements ($n = 18$) to generate a distribution of model-fit inclusive of out-of-sample prediction accuracy.
4. Results of this process were presented as follows:
    a. The single parameter set that provided the best model fit (out of the $n = 333$) was extracted
    b. The full set of fitted model predictions (i.e., from the $n = 333$ parameter sets) was visualised by a density plot (figure 19 in Turner *et al*. (2017))
    c. Histograms of model parameters obtained over the 333 sets were plotted demonstrating variability in fitted estimates.

The cross-validation approach in Turner *et al.* (2017) is representative of some awareness amongst contemporary researchers working in the experimental FFM literature of the requirements for appropriate model evaluation. However, it may still be improved with incorporation of model testing against out-of-sample data obtained under different inputs (i.e., a future training block or tapering period). In addition, walk-forward rather than random hold-out type approaches offer a more principled approach to mixing past and future data and respecting the temporal order of time-series data (Prado, 2018). This was discussed in the literature review of chapter 2, and is given further attention in subsection 6.2.5, where an implementation of a walk-forward cross-validation approach is developed to guide researchers working in R. Therefore, the exact approach from Turner *et al*. (2017) is not replicated here to avoid confusion on the recommended future approach within this work. However, demonstration of a genetic algorithm from the package *GA* (Scrucca, 2013) to fit the nonlinear model system to the synthetic data from listing 6.17 is demonstrated (listing 6.25). This is in contrast the derivative-based (quasi-Newton) algorithm used in listing 6.22 where the original model system was fitted. Tuning evolutionary algorithms to obtain the best results can often be challenging, and researchers are referred to the literature for discussion of these aspects (Smit and Eiben, 2010), and encouraged to consult with experts in optimisation when utilising evolutionary algorithms within experimental designs. The genetic algorithm function call in listing 6.25 replicates approximately the same tuning parameters used by Turner *et al.* (2017) (tournament selection, BLX-α crossover, Gaussian mutation) but includes the optional addition of constrained local search via the L-BFGS-B algorithm and also parallelises the fitting process via default architecture applied by the GA package. More discussion of the parallelisation architecture available in R is given in subsection 6.2.6 (cross-validation) along with discussion of its relevance to FFM research.

Listing 6.25	Fitting the nonlinear ODE system to synthetic data (listing 6.17) under NLS via a genetic algorithm from the package *GA*, with tuning parameters equivalent to Turner *et al.* (2017). Also includes optional local search (L-BFGS-B), and parallelised fitting process.

```
# Packages required
require(parallel)
require(doParallel)
require(GA)

# Need to bound Turner as optimiser struggles when alpha or beta nears <=0
constraints <- data.frame(lower = c(50, 0.1, 1, 0.1, 1, 0.1, 0.1, 0.4, 0.4),
                          upper = c(200, 10, 50, 10, 50, 20, 20, 3, 3))

fittedModel <- GA::ga(type = "real-valued",         # Type
                      fitness = turnerObjective,   # Objective function
                      perfVals = mockPerformances, # Measured performance values
                      loads = loads,               # Training loads
                      lower = constraints$lower,   # Lower bound
                      upper = constraints$upper,   # Upper bound
                      maxiter = 1000,              # Maximum number of iterations
                      monitor = TRUE,              # Console output during process
                      popSize = 90,                # Population size
                      optim = FALSE,               # Include local search (Y/N)
                      # if (optim = TRUE) the following reflect the local search args
                      # optimArgs = list(method = c("L-BFGS-B"),
                      #     poptim = 0.1, # [0,1] probability of performing a local search
                      #     pressel = 0.5, # [0,1] pressure selection (default 0.5)
                      #     lower = constraints$lower,
                      #     upper = constraints$upper,
                      #     control = list(maxit = 1500)),
                      elitism = 7.5,                    # Survival at each generation (%)
                      selection = gareal_tourSelection, # Tournament selection
                      crossover = gareal_blxCrossover,  # BLX-alpha crossover
                      mutation = gareal_rsMutation,     # Random Gaussian mutation
                      run = 150,                        # Halt value
                      parallel = "multicore",           # Multi-platform parallelisation
                      seed = 12345                      # Seed for replication later
                  ) # End of GA call
```

Notes (listing 6.25):

- The function and package GA maximises by default, and its clear that maximising the RSS is not desired, so it needs to be converted to a maximisation problem by multiplying the objective function by -1. This must be done in the objective function code itself (at the return line), before running the function call in listing 6.25 in order to get adequate (any) results.
- Readers can see (github.com/bsh2/thesis/c6/banister_and_turner.R) where this is demonstrated.
- Bounding is necessary on the alpha, beta parameters in particular as when they tend to zero the fitting process bumps into errors.

6.2.4 Optimisation algorithms and parameter estimation in R

In this chapter so far, fitting FFMs using the objective functions developed (for NLS or maximum likelihood estimation) has been performed almost entirely by the box-constrained limited-memory modification of the quasi-Newton BFGS algorithm (L-BFGS-B), from the general-purpose optimisation package, *optimx* (Nash *et al.*, 2020). However, no attention has been paid to the theory behind this algorithm, or optimisation in general, with the emphasis in previous sections directed toward the structure of code for fitting FFMs in R, rather than any particular algorithm that might be used to solve the optimisation problem. At the end of the previous subsection (6.2.3), a genetic algorithm was shown for fitting the nonlinear model system from Turner *et al.* (2017). This included replication of the same tuning methods as used by the authors, facilitated via the R package *GA* (Scrucca, 2013). However, no theory behind this algorithm was provided. Although model fitting is an area of FFM research in need of investment, it is an area best tackled in a collaborative setting with the involvement of optimisation experts (e.g., applied mathematicians). Nevertheless, it is still important that sport science researchers have a basic understanding of the associated theory, can manipulate models in languages such as R, and have an awareness of the options available for formulating model fitting problems (e.g., NLS, MLE) and available solvers.

More generally, it is this authors opinion and experience within this project that if sport scientists are encouraged more often and particularly during their early training to upskill in quantitative areas, more opportunities are likely to arise for future interdisciplinary collaboration to tackle these types of research problems. Upskilling is also important to avoid common barriers to interdisciplinary research, including an absence of common language and knowledge of other disciplinary traditions and conventions that causes discrepancies between each side's understanding of the task or appropriate methods to be used (Siedlok and Hibbert, 2014). These types of issues may also extend beyond the research process and manifest in the wider communication of methods, results, and practical recommendations arising from collaborative work. In the context of FFM research, previous studies have either offered very limited reporting and communication of technical aspects, or the work has been placed in technical journals and therefore often overwhelmed by mathematics that does not match the focus of a general sport science audience. There is a clear balance to be struck in FFM research as to how and where findings are communicated. The quantitative experts on one side must recognise that the general sport science audience (i.e., the prospective stakeholders) are typically not interested in the particular details of an algorithm or method used to estimate model parameters. Rather, they need to be pointed to the potential impact of the work on practice or implications for research, and its ability (or not) to contribute toward efforts to solve real-world training design problems via modelling. This aspect of information needs to be front and centre wherever possible in future work. On the opposite side, sport scientists can benefit the process from having increased awareness of technical

aspects of the modelling process. Arguably, this is the benefit of appendices and supplementary material, to make sure no useful information is lost but that the key information and context is placed front and centre. The hypothesis is that if sport scientists begin to play a more active role in the technical aspects of interdisciplinary work (i.e., do some of the 'heavy lifting'), this communication problem will naturally resolve in a substantive way, and they will be able to act as pivot points between experts in different domains. To assist in equipping sport science researchers with baseline knowledge in fitness-fatigue model fitting, this section focusses on briefly outlining the basic theory of certain optimisation algorithms and availability of different solvers in the R environment. To begin with, a general overview of optimisation is provided, followed by brief discussion of common gradient-based methods used to solve nonlinear data fitting problems, and then finally moves on to a discussion of evolutionary methods as a potential avenue of future research in fitness-fatigue modelling.

The principal concept in optimisation is the use of a rigorous mathematical model to determine the most efficient solution to a defined problem (Nash, 2014). At the heart of an optimisation problem is the so-called 'objective'. The objective is a quantitative measure of the suitability of a given solution, and in many cases may reflect a single (scalar) real-world quantity such as financial cost (£) or time and is typically a real-valued function. Therefore, solving an optimisation problem involves either maximisation or minimisation of some objective, through iterative evaluation of a mathematical representation under new inputs in the feasible domain (typically described by the constraints or just a number set, e.g., $\mathbb{R}$). This task is performed by an optimisation algorithm (also sometimes termed a solver), that uses an iterative scheme to methodically choose new input values to find the solution that yields the absolute maximum or minimum value of the objective. In data-fitting problems, the variable inputs in the objective function is commonly a set of model parameters (θ) that are adjusted until the model $(\hat{y}(\theta))$ matches a set of real-world datapoints (y). Other inputs may be fixed in the data fitting problem, such as the model predictors x. Recall that the general least squares problem is described mathematically by:

$$\min \sum_{i=1}^{N} (\hat{y}(\theta, x_i) - y_i)^2 \tag{6.24}$$

The likelihood objective function was also demonstrated in subsection 6.2.2 as an alternative to NLS. The objective function may also be referred to in some texts as the 'cost function', 'loss function', 'utility function' or 'fitness function', often depending on the field in which the problem has arisen and whether maximisation or minimisation is sought. Optimisation is also sometimes referred to as 'mathematical programming', often used to mean the satisfaction of constraints whilst attempting to minimise (or maximise) an objective function (Nash, 2014). The definition appears to arise from the use of mathematical models, theory, and computation to inform decision making about the best use of

limited resources to solve a real-world problem. Although the basic concepts of optimisation are broadly graspable, the area is a large research field at the centre of which are the crossed domains of applied mathematics, statistics, and computer science. Development of an optimisation problem and selection of an appropriate algorithm in many cases may require careful consideration and is best approached under the supervision or collaboratively with an expert or well-informed party. In fitness-fatigue modelling, optimisation of existing models (i.e., model fitting to data, parameter estimation) is an area that has been largely ignored by researchers, further suffering as described earlier by limited reporting of methods used in experimental study (see literature summary tables in appendix B, table B-3).

Recent work, including a study in this thesis (chapter 5) and a recent study by Connor & O'Neill (2020) has examined concerns with the historical approaches used, and the adequacy of common solvers that may be applied in experimentation to solve the nonlinear least-squares problem. It is crucial to appreciate that the methods used to fit FFMs to experimental data cannot be separated or downplayed in the interpretation of results. If in say a least-squares problem, the global minimum can't or isn't found due to the problem containing many local extrema, it is impossible to determine if the parameters obtained can be considered physiologically interpretable, and if carried forward to out-of-sample testing will ultimately effect estimation of the model's ability to predict future response. In other words, in this case the problem has not been adequately solved as it was posed, the solutions reported, and interpretation of associated results is not necessarily sound. In reality, we can rarely ever know whether a solution found is the absolute minimum as most problems are large residual in nature due to some model misspecification. However, some practical efforts can be made to reduce the chances that a given solution is reasonable. For example, fitting the model from multiple starting points over the feasible domain (box constraints) for derivative-based methods, and assessment of the hessian for positive-definiteness indicating that the solution is a minimum and not a saddle (Nash, 2014). Optimality conditions say that, if a point is a minimum of a function it will possess a zero gradient and positive-definite Hessian (Nash, 2014). Another practical step may be the use and comparison of multiple algorithms to fit the models, to identify the optimal set found across multiple approaches. It is concerning that a significant proportion of previous experimental research has ignored basic questions surrounding the suitability of estimation methods and has simply adopted approaches from previous study without further consideration. The ramifications of this on the broader interpretation of the literature are significant as described above, with the efforts clouded by little appreciation for how the solutions were obtained. Though, many of these experimental studies only ever considered in-sample model fit, which already tells us very little about model utility as it pertains to future use in training program optimisation.

The FFM is a nonlinear optimisation problem in its model parameters, and estimation to fit an FFM to available data involves minimising an objective function (e.g., sum of squared errors between modelled and measured performance values) or maximising a likelihood function (Nash, 2014). If a model is linear its parameters, e.g., $ax_1 + bx_2 + c$ (i.e., parameters appear only to the first power), standard calculus could be used to arrive at the normal equations and solve the optimisation problem analytically via matrix algebra (Nash, 2014). However, if minimising the sum-of-squared residuals (error) of a nonlinear model (e.g., $ax_1 \times x_2^b \times c$) an iterative method is generally required, that begins at some set of starting values for the model parameters a, b, c. In chapter 5, the practical implications of the existence of multiple minima in the standard and fitness-delay FFM fitting problem were demonstrated, as it impacts on starting point selection, a classic problem in optimisation of many nonlinear models (Nash, 2014). Most methods to solve optimisation problems rely on traditional calculus and can find local minima and maxima in an efficient manner. However, in instances when functions have multiple local extrema, often they do not often guarantee that the solution found is the right one (e.g., the global minimum). In chapter 5, this concept was demonstrated when the model was re-fit from many different starting points. As predicted, the quasi-Newton algorithm used to solve the NLS problem was able to find multiple local minima in addition to the artificial global minimum, with solutions obtained depending on starting point. Another key aspect of the optimisation process is the incorporation of constraints. The reality of most real-world problems is that they have constraints, although in some cases the optimisation problem can be solved without explicit inclusion (Nash, 2014). In the case of fitting an FFM to data, the most obvious constraint is the use of bounding or box constraints on the model parameters, imposing some upper (u) and lower (l) bound, $(l \leq \theta \leq u)$ on each. The τ parameters in particular may be chosen based on the number of decays over which fitness and fatigue are expected to decay. Another type of constraint that could be enforced is an inequality constraint (e.g., $k_g < k_h$). In some cases, bounds on the inputs may even be necessary to prevent an optimiser attempting to evaluate the objective function at values that are mathematically intractable (e.g., division by zero, or taking the log of a negative number).

Gradient descent is perhaps the most basic approach to finding a local minimum of an objective function f (e.g., eq. 6.24), by computing the gradient of f at a point, then travelling a certain distance in the direction in which this gradient is reduced (i.e., "downhill"), before repeating this step. If f is an n-dimension function (i.e., multidimensional), the gradient is simply the n-dimensional slope of f (Nash, 2014). In other words, the approach doesn't really change. Gradient descent forms the basis of the general methods of steepest descents, and is one of the foremost approaches to optimisation, generally attributable to the work of the mathematician Cauchy (Nash, 2014). If the gradient descent algorithm is at a point k in the feasible domain of the parameter space θ, denoted θ_k, and the objective function f is defined and differentiable around x_k, then the objective function decreases fastest in the

direction of the negative gradient of f from the point x_k, that is $-\nabla f(x_k)$. Where $\nabla f(x)$ is the Jacobian matrix (generalisation of first-derivatives in multivariable problems). If we consider a step-size (or learning rate) of λ, i.e., how far we move downhill, it follows that the gradient descent algorithm finds the next position to evaluate x_{k+1} by:

$$x_{k+1} = x_k - \lambda \nabla f(x_k) \tag{6.25}$$

Although gradient descent can be effective when a function is straightforward and F convex over its feasible domain (i.e., all local minima are global minima), such as in the case of fitting linear models, it can often struggle with pathological functions in n-parameters, requiring many iterations (Nash, 2014). It is also a local optimiser by nature, and if initiated from a single point in the case where many local optima exist, it has a good chance of returning a local extremum rather than the global minimum we are normally interested in. The best choice of step-size (λ) is also an area of much debate and relevant practical problem in the implementation of gradient descent. Too large a step-size, and you may overshoot the minimum, too small, and convergence will be much slower than necessary (Nash, 2014). Gradient descent is also defined as a *first order* algorithm, because it uses knowledge of a functions first-derivative.

Newton's method is another equally well-known approach in optimisation. It is also attributable to the mathematician Joseph Raphson and so is sometimes referred to as the Newton-Raphson method. In its origin, it was an iterative root-finding method for a twice differentiable function f, and when applied in its modern form within optimisation applications is a second-order method for finding the minimum of f (Nash, 2014). The generalisation of the second derivative of f in the multivariable (n-parameter) case is called the Hessian ($\nabla^2 f$) matrix. In contrast to gradient descent, which attempts to find the minimum value of f directly, Newton's method iteratively attempts to move toward where the derivative of f is zero, i.e., the location of the function where there is no slope (Wright and Nocedal, 1999; Nash, 2014). In other words, the gradient descent approach moves in a direct manner, whereas Newton's method uses the curvature of the function. To understand how this works, it is helpful to first examine the method in its original root finding form for the 1 variable case, say for the function $y = f(x)$. The root-finding method begins at some initial point $\big(x_0, f(x_0)\big)$, which can be thought of as an initial guess to the solution of $f(x) = 0$ (i.e., the root of f). The method then computes a linear approximation of $f(x_0)$ (a tangent line of f at x_0) and then solves this to find the x intercept of the linear approximation. Recall that a linear approximation of f at x_0 is given by the line $y = f(x_0) + f'(x_0)(x - x_0)$. Letting the x intercept of this linear approximation be denoted x_1, x_1 can then be found by solving $0 = f(x_0) + f'(x_0)(x_1 - x_0)$, which gives:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{6.26}$$

If we repeat the process above from the next point $\big(x_1, f(x_1)\big)$, and so on…then a sequence emerges:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{6.27}$$

Which in theory will converge to the solution (root) of f, $f(x) = 0$. In optimisation applications, the problem changes from finding $f(x) = 0$, to $f'(x) = 0$, because we are interested now in finding the point on the function at which the derivative is zero (i.e., the slope is at the bottom of the valley or the top of a hill). Therefore, we instead require (at least) a second-order polynomial approximation (quadratic equation) of f at x_k which we can then find the minimum of and repeat the process. This is the crux of the difference in the method as it evolved for us in optimisation. Taylor expansions provide us with one such tool to do this, and following the steps above a sequence (at the iterates) will emerge shortly from the truncated (second order) Taylor approximation below:

$$f(x_k + \Delta x) \approx f(x_k) + f'(x_k)\Delta x + \frac{1}{2}f''(x_k)\Delta x^2 \tag{6.28}$$

And therefore, the next point $\big(x_{k+1}, f(x_{k+1})\big)$ emerges from $x_{k+1} = x_k + \Delta x$, where $\big(x_k + \Delta x, f(x_k + \Delta x)\big)$ is the minimiser of this quadratic approximation (i.e., the stationary point). Setting the derivative of this quadratic approximation to zero and solving gives:

$$\frac{d}{d\Delta x}(f(x_k) + f'(x_k)\Delta x + \frac{1}{2}f''(x_k)\Delta x^2 = 0 \tag{6.29}$$

$$f'(x_k) + f''(x_k)\Delta x = 0 \tag{6.30}$$

$$\Delta x = \frac{f'(x_k)}{f''(x_k)} \tag{6.31}$$

And therefore, Newton's method iteratively performs the following step to find $f'(x) = 0$:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \tag{6.32}$$

This concept is illustrated graphically in figure 6.14 at the top of the next page.

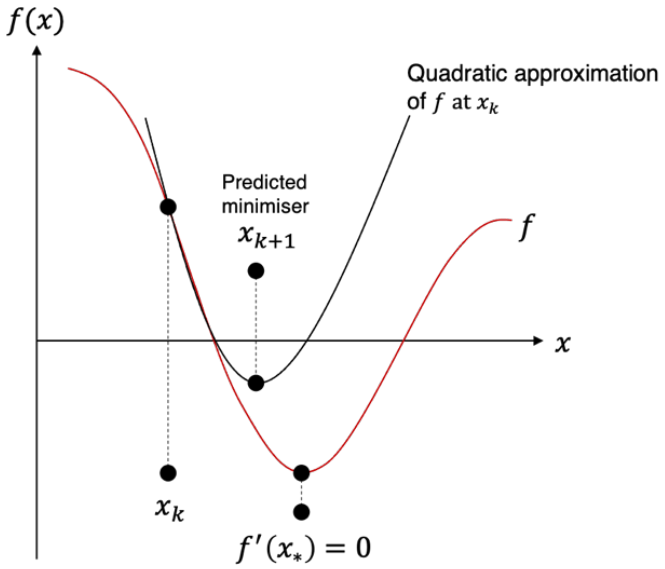


Figure 6.14: A graphical illustration of the update step in Newton's method

Finally, formally defining Newton's method in the multivariable case. Let $f: \mathbb{R}^n \to \mathbb{R}$ be sufficiently smooth, and the algorithm is initialised at $x_0 \in \mathbb{R}$,

$$x_{k+1} = x_k - \frac{\nabla f(x_k)}{\nabla^2 f(x_k)}, k \geq 0 \tag{6.33}$$

Where $1/\nabla^2 f(x)$ is the inverse of the Hessian. This method may also be modified to include a smaller step size ($0 < \lambda \leq 1$). If $\lambda \neq 1$ Newton's method may be referred to as relaxed or damped. Readers are referred to (Wright and Nocedal, 1999), for further discussion of rationale behind the use of a modified step size in Newton's method, particularly in the areas of convergence and their use in quasi-Newton methods.

General advantages of second-order methods such as Newton compared to first-order methods such as gradient descent are that they are generally much faster if the second derivative is known or easy to compute. However, if the analytic expression for the second derivative is complicated or mathematically intractable, numerical approaches for computing the second derivative are required and are expensive. In addition, Newton's method may fail to converge if the starting point x_0 is too far away from the solution. The method is also not generally globally convergent on f, and the presence of multiple minima can cause problems. For high dimension problems, these risks are typically higher (Nash, 2014). It may seem obvious from the definition, but the method also does not

work if the inverse of the Hessian is not tractable. Certain features in the objective function, such as saddle points, can be problematic for Newton's method, even causing it to move in the wrong direction in some cases and requiring a regularisation term (α) along the diagonal of the Hessian to offset negative eigenvalues of the Hessian, such that the multivariable case would then be defined:

$$x_{k+1} = x_k - \frac{\nabla f(x_k)}{(\nabla^2 f(x_k) + \alpha \boldsymbol{I})} \tag{6.34}$$

For further discussion of regularisation terms and the issues surrounding the use of Newton's method to train complex models with equally complex objective functions, readers are referred to (Goodfellow *et al.*, 2016). There also exist quasi-Newton methods, which have featured in the work so far, specifically the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm and its limited memory modification and box constrained variants (Byrd *et al.*, 1995b; Henao, 2014). These involve an approximation of the Hessian matrix (or more directly its inverse). The primary benefit of quasi-Newton methods is that they are computationally cheaper to the full Newton method as the Hessian does not need to be computed at each step and they require $\mathcal{O}(n^2)$ operations per iteration, compared to the full Newton method that requires $\mathcal{O}(n^3)$ (Henao, 2014). Therefore, in instances where the analytic expression for the second derivative is unavailable, and therefore the Hessian must be computed numerically, quasi-Newton methods may prove a useful alternative. The BFGS algorithm is a well-known optimisation algorithm with implementations available in many programming languages and mathematical suites (e.g., mathematica, MATLAB, R, and the sciPy extension to Python). In the standard BFGS algorithm (without the limited memory modification) an approximation (B) to the Hessian is slowly improved by a secant method (finite-difference approximation) that uses first-derivative information. The algorithm follows the following steps:

1. Initiate from a starting point $\mathbf{x}_0$ and approximate Hessian matrix B_0
2. A direction $\boldsymbol{\delta}_k$ is obtained from $\boldsymbol{\delta}_k = -\nabla f(x_0) B_k^{-1}$
3. A one-dimensional line-search (of some kind) is performed to find an acceptable step-size λ_k in the direction $\boldsymbol{\delta}_k$ found from the first step. Such that, $\mathbf{s}_k = \lambda \mathbf{p}_k$
4. $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$, there $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$
5. Finally, the approximate Hessian matrix B is updated to

$$B_{k+1} = B_k + \frac{\mathbf{y}_k \mathbf{y}_k^{\mathrm{T}}}{\mathbf{y}_k^{\mathrm{T}} \mathbf{s}_k} - \frac{B_k \mathbf{s}_k \mathbf{s}_k^{\mathrm{T}} B_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} B_k \mathbf{s}_k}$$

The BFGS algorithm

In some cases, for the standard BFGS algorithm, B_0 may be initialised to the identify matrix $\boldsymbol{I}$, in which case the first step is equivalent to gradient descent, but subsequent updates are improved by better approximation to the Hessian. In the implementation of this algorithm in the R package *optim*, this is the process used. During the line search, failure to improve the function value is a component of the termination tests (Nash, 2014). The limited memory modification of the BFGS algorithm is the default algorithm used in the *optimx* R package when box constraints are required (the BFGS algorithm with bounding is not available in R). The L-BFGS algorithm uses less memory to update the approximation to the inverse of the Hessian, by only storing a record of the last m iterations (rather than an $n \times n$ matrix where n is the number of variables in the problem). As such, L-BFGS only requires $\mathcal{O}(mn)$ operations per iteration so is well suited to problems where the number of free parameters n is large; as is the case of the fitness-fatigue model.

There are many practical resources (including those already mentioned) that are available for readers to obtain greater familiarity with the technical aspects relating to these algorithms and written by more experienced individuals and educators. With this in mind, it is important to refresh the purpose of this section for the reader, which is to introduce the most relevant technical material in the context of fitting FFMs, whilst attempting to avoid tangential information. Therefore and prior to discussing evolutionary algorithms, the following section will only highlight the existence of two further 'classical approaches that may be of interest to solving nonlinear least-squares problems: The Gauss-Newton algorithm, and Levenberg-Marquardt method. Gauss-Newton can be thought of as a Newton method with rectangular Jacobian matrix, where each update is a linear least squares problem (i.e., assuming the least-squares function is locally quadratic and finding the minimum). Problems can arise using the Gauss-Newton method if the objective function is highly nonlinear and for large residual problems. More detail surrounding this algorithm can be found in (Nash, 2014). However, many prefer the Levenberg-Marquardt method that combines gradient descent with the Gauss-Newton method. It is generally more robust, due to its ability to find solutions far away from the final minimum in well-behaved functions as it behaves more like a gradient-descent approach when far from x_* but more like the Gauss-Newton method when close.

Some data fitting problems, including the FFM, are awkward and challenging to solve. If the field is to progress, this reality must be recognised and embraced by researchers, not ignored. In the context of fitness-fatigue modelling, the main issue with application of algorithms discussed so far is that they are local minimisers. They can struggle when problems do not behave well, and in the presence of multiple local minima and other features such as saddle points (as shown in chapter 5). It is therefore important that future research look toward methods that may facilitate global optimisation, including approaches as simple as refitting the algorithms discussed so far from multiple random starting points (a basic stochastic local search) (Nash, 2014). These are the focus of the remaining written part of this

subsection. At the end of the subsection, a table of available algorithms is presented, accompanied by a code listing that demonstrates some of them in R from the model implementations and mock data developed in subsection 6.2.2 for MLE.

The class of evolutionary algorithms are of particular interest in fitness-fatigue modelling, representing a slightly more brute force approach to the problem of finding global optima. There has been very little previous study of these in FFM experimental (Turner *et al.*, 2017; Méline *et al.*, 2018; Philippe *et al.*, 2018) or theoretical (Connor and O'Neill, 2020) research. The evolutionary class are a group of algorithms that are based on patterns and concepts observable in nature, specifically evolution (survival of the fittest, genetic inheritance) as reflected in mechanisms (tuning parameters) such as selection, mutation, reproduction, and recombination that control the behaviour of the search (Bäck and Schwefel, 1993; Philippe *et al.*, 2018). The area of evolutionary algorithms is extensive and too vast to cover in any meaningful depth here, and readers are directed to a wealth of resources available that consider the specifics of the many algorithms that exist. However, this section will discuss broadly some advantages and disadvantages of evolutionary strategies as a tool for global optimisation, as well as practical implications that arise with their use. This short brief is intended to stimulate ideas for alternatives to common local optimisation methods that the reader may wish to investigate further themselves. The key advantage of evolutionary algorithms is that they can scale well to high dimensionality problems, particularly where there is a high number of local minima, as the search direction does not rely on derivatives. They are often robust to poorly behaved objective functions and may be able to find better solutions to problems that have not consistently been solved well by other more standard techniques (Fleming and Purshouse, 2002). A key disadvantage of evolutionary algorithms is that some way of ending the search process must be devised. Typically, an evolutionary algorithm can either reach a maximum runtime (number of iterations allowed), or a threshold that is set by the user either on reduction of the objective function or some other measure if the minimum value was known (which it isn't in fitness-fatigue modelling). This contrasts with methods such as gradient descent that can provide guarantees on optimality under some conditions due to the use of the curvature of the functions. In addition, they often require understanding of the underlying theory (i.e., genetics) or computational study to tune the parameters for the problem at hand and maximise performance of the algorithm. This can be a time-consuming process, and so more generalised recommendations for tuning parameters are turned to, that may not provide the best results possible with regards to time efficiency or solutions found. Connor & O'Neill (2020) examined the differential evolution (DE) algorithm as an alternative approach to the quasi-Newton L-BFGS-B algorithm, comparing optimality of solutions found. Differential evolution is based on the evolution of an initial population (of candidate solutions) under genetic operators such as crossover and mutation. It is a direct search method in that it investigates points close to current positions to find the lowest objective without gradient information, generating new parameters via a set of evolutionary parameters,

repeating the process until it reaches a solution below a specified threshold or runs out of iterations. The authors identified in their study that this approach had shown promise for nonlinear dose response modelling in the realm of toxicology and pharmacology (Ma, Bair and Motsinger-Reif, 2020), and so may be a good candidate for further study with the FFM. The authors also reported that the differential evolution method produced a less variation in model fit, parameter variability, and hold-out performance compared to the L-BFGS-B algorithm, although noted that it took more time on average to fit. With no 'true conditions' established, as would be possible within a simulation approach, the authors were unable to determine whether the best set of parameters were found, however, it did appear as though differential evolution was able to find a lower absolute objective value than BFGS. DE is available within R within the package *DEoptim* (Mullen *et al.*, 2011), along with several other evolutionary and natural strategies (see table 6.2). Of other significant note is the package *GA* (Scrucca, 2013), which also provides derivative-based local search via the L-BFGS-B algorithm in optim at pseudorandom intervals across the iterations. This type of functionality is particularly helpful, as it increases the chance of finding the absolute minimum by incorporating guarantees on optimality that come with some derivative-based methods with robustness to poorly behaved nonlinear functions with many local minima and is certainly one that appeals as an avenue for further work.

To summarise, this subsection has drawn attention to the availability of multiple algorithms in R for solving the nonlinear least-squares and maximum likelihood problems that might be used for fitting collected training and performance data to FFMs. These have included standard first and second-order methods (e.g., gradient-descent, quasi-Newton, and Gauss-Newton), as well as those from the class of evolutionary strategies (e.g., genetic algorithms, differential evolution, particle swarm, and covariance matrix adaptive evolution) that are stochastic, derivative-free methods for global optimisation that can do well in the presence of many local extrema. The purpose of the section was to introduce the reader to the theory behind optimisation and classical algorithms and provide an overall picture of the data fitting problem and provide some ideas as to existing tools (algorithms) that may be investigated within R for experimental research. The primary recommendation is that researchers be alert to the existence of local minima, a problem that represents one of the most likely causes of issues in future work. Additionally, it is recommended that researchers explore approaches that utilise stochastic methods, compare multiple approaches, and include some basic checks on solutions found. As a minimum standard, if using a first or second-order algorithm that requires starting points, stochastic local search is appropriately implemented as repeated refitting of the FFM from multiple starting points (and selecting the best, whilst reporting the variability). There are many further (and more complete) resources available for researchers to better understand concepts in optimisation (Wright and Nocedal, 1999; Goodfellow *et al.*, 2016), including those specifically for R (Nash, 2014). Furthermore, researchers should be aware of the documentation that exists for specific packages (such as those listed in table 6.2). These are often thorough and should be read to ensure correct usage. Finally, it is

recommended for sport science researchers conducting experimental research and are unsure of practical methods and theory of model fitting should look to draw knowledge, advice, and help from experts in this area. Many of these experts can be found in the faculty of computer science departments in their own institutions, so they often need not look too far.

Table 6.2: A non-exhaustive list of prospective optimisation algorithms in R for fitting FFMs.

Package	Description	Algorithm	Function	Function details	Documentation
optimx	Functions for minimisation of general optimisation problems	Wrapper for stochastic local search (multiple algorithms)	`multistart`	Wrapper function use of several starting-point algorithms available in *optimx*. Supports a matrix of starting parameters as input (that can be generated randomly prior). Results compiled and returned as a dataframe.	(Nash *et al.*, 2020)
optimx		BFGS, Conjugate Gradient	`optimx`	Wrapper function for several algorithms including BFGS (and L-BFGS-B), CG (conjugate gradient), Nelder-Mead (direct search)	(Nash *et al.*, 2020)
stats	R statistical functions	Gauss-Newton	`nls`	Function to determine the NLS estimates of model parameters of a nonlinear model via Gauss-Newton (some other algorithms available)	see `?nls` in R
DEoptim	Global optimisation by evolutionary strategy or natural strategy	Differential evolution	`DEoptim`	Derivative-free optimisation approach, similar in function call to *optimx* but requires some care. Several tuning parameters available.	(Mullen *et al.*, 2011)
pso		Particle swarm	`psoptim`	General implementation of particle swarm optimisation. Same basic function call structure as *optimx* (plug-and-play). Some tuning parameters available.	(Bendtsen and Bendtsen, 2011)
GA		Genetic algorithms	`ga`	Similar in function call to *optimx* but requires some care. Maximises by default, so objective function may need to be multiplied by -1. Several tuning parameters available. Includes a random local search option via L-BFGS-B.	(Scrucca, 2013)
cmaes		Covariance matrix adaptive evolution strategy	`cmaes`	Same basic function call structure as *optimx* (plug-and-play). Some tuning parameters to play around with.	(Trautmann *et al.*, 2015)

Finally, code listings are provided to demonstrate the plug and play nature of using different available optimisation algorithms in R (highlighted in the table 6.2), to fit models to the synthetic data developed in listing 6.17 (subsection 6.2.2), via the functions developed so far in this chapter. Readers are encouraged to download code files associated with this subsection (listings 6.26 to 6.28).

Example 1: `multistart` wrapper function (*Optimx* package) - L-BFGS-B algorithm

Listing 6.26	Standard model fitted to synthetic data (listing 6.17) via MLE (solved by L-BFGS-B) under the multistart function in optimx

```
require(optimx)   # install.packages("optimx")

# Develop matrix of parameter values (you could do this randomly)
pars <- as.matrix(expand.grid("p0" = seq(80, 120, length.out = 2), "kg" = seq(0.5, 2,
length.out = 2),
                  "Tg" = seq(5, 40, length.out = 2), "kh" = seq(0.5, 2, length.out = 2),
                  "Th" = seq(5, 40, length.out = 2), "sigma" = seq(0.1, 1, length.out = 2)))

# Apply the multistart algorithm under the "L-BFGS-B" algorithm in optimx
standard_model1 <- optimx::multistart(parmat = pars,
                                      fn = standardObjectiveLL,
                                      method = "L-BFGS-B",
                                      # ORDER: c(p0, kg, Tg, kh, Th, sigma)
                                      lower = c(60, 0.1, 1, 0.1, 1, 0.1),
                                      upper = c(200, 3, 50, 3, 50, 10),
                                      loads = loads,
                                      perfVals = mockPerformances_noerror)
head(standard_model1)
```

(Output returned as a dataframe with each row representing a fitting iteration from a different start)

Of note, with this approach to multi-start local search it can expand rapidly when attempting to cover the parameter space in small incremental distances. This is inevitable (a classic issue with grid search approaches in general), and why a stochastic approach is more likely a better method to devising starting points (rather than trying to incrementally cover the parameter space in a uniform grid). If large computer resources are available, for large grids parallelising the fitting iterations is also a good approach but would require albeit simple development of your own parallel multi-start type function. Parallelisation is discussed and demonstrated in the next subsection (6.2.5) via the use of a simple looping construct. Very few studies have applying derivative-based methods have reported the utilisation of some form of multistart procedure (Rozendaal, 2017; Scarf *et al.*, 2019).

Example 2: `DEoptim` function (*DEoptim* package) – Differential Evolution algorithm

Listing 6.27	Fitting the standard model to synthetic data within an MLE approach solved via differential evolution

```
standard_model2 <- DEoptim::DEoptim(standardObjectiveLL,
                                    lower = c(60, 0.1, 1, 0.1, 1, 0.1),
                                    upper = c(200, 3, 50, 3, 50, 10),
                                    DEoptim.control(
                                      iter = 1000,
                                    ),
                                    loads = loads,
                                    perfVals = mockPerformances_noerror)
```

Notes (listing 6.27):

- In this implementation, the algorithm is set to terminate at the maximum number of iterations (1000). Alternatives exist for specifying a 'value to reach' (argument `VTR`). See (Mullen *et al.*, 2011)
- Other control arguments for DEoptim include strategy (6 available), number of population members, evolutionary parameters (crossover probability, differential weighting factor) and the ability to specify an initial population and store populations at each iteration. See Mullen *et al.* (2011).

Example 3: `ga` function (*GA* package), `cmaes` function (*cmaes* package), `pso` function (*psoptim* package) – Genetic algorithm, Covariance-matrix-adaptive-evolutionary strategy, Particle swarm optimisation

Listing 6.28	Fitting the standard model to synthetic data within an MLE approach solved via genetic algorithm, covariance-matrix-adaptive-evolution strategy, and particle swarm optimisation

```
# GA, cmaes, psoptim packages
standard_model3 <- GA::ga(type = "real-valued",
                          fitness = standardObjectiveLL,
                          perfVals = mockPerformances_noerror,
                          loads = loads,
                          lower = c(60, 0.1, 1, 0.1, 1, 0.1),
                          upper = c(200, 3, 50, 3, 50, 10),
                          maxiter = 1000,
                          monitor = TRUE,  # Provide output to console
                          optim = TRUE,    # With L-BFGS-B local search at random
                          maximise =TRUE   # Passed to standardObjectiveLL
                          )

standard_model4 <- pso::psoptim(par = c(NA,NA,NA,NA,NA,NA,NA),
                                fn = standardObjectiveLL,
                                lower = c(60, 0.1, 1, 0.1, 1, 0.1),
                                upper = c(200, 3, 50, 3, 50, 10),
                                control = list(maxit = 1000),
                                loads = loads,
                                perfVals = mockPerformances_noerror)

standard_model5 <- cmaes::cma_es(par = c(95, 1, 28, 2, 5, 0.1), # guess
                                 fn = standardObjectiveLL,
                                 lower = c(60, 0.1, 1, 0.1, 1, 0.1),
                                 upper = c(200, 3, 50, 3, 50, 10),
                                 control = list(maxit = 1000),
                                 loads = loads,
                                 perfVals = mockPerformances_noerror)
```

Notes (listing 6.28):

- As described at the end of the previous subsection, the value returned by the `standardObjective` function must be changed for `ga`, as the package maximises by default. Recall earlier that the maximise argument in `standardObjective` allows us to change the returned value to either negative log-likelihood (`maximise = FALSE`, default) or the log-likelihood (`maximise = TRUE`). Hence, we can pass this as an additional argument within the `ga` function call.
- The par argument in psoptim is simply there to tell the algorithm the dimensionality of the problem, hence NA values can be used in the vector supplied to this argument. See (Bendtsen and Bendtsen, 2011) for discussion of algorithm parameters that may be adjusted to control the behaviour of the search.
- The algorithm CMA-ES requires initial values. See (Hansen, 2016) for further discussion of this algorithmic strategy.

6.2.5 Cross validation (Implementing a walk forward approach)

When evaluating an FFM for forecasting accuracy, or to assess its counterfactual properties, we are interested in how well it does at predicting data not used to train it (i.e., its generalisability). Sole evaluation of an FFM on training data, as has historically been the case, can only provide overly optimistic estimates of the model's performance. It also does not provide insight into how well the model may perfrom if it were used operationally, such as for predicting future response to different training programs. This is a fundamental flaw in the previous FFM experimental literature, and also the reason why model predictive validity is still not clear over 40 years later. However, some more recent studies have made attempts to improve the quality of model evaluation through inclusion of some form of cross validation (Chalencon *et al.*, 2015; Kolossa *et al.*, 2017; Turner *et al.*, 2017; Williams *et al.*, 2018; Stephens Hemingway *et al.*, 2019; Imbach *et al.*, 2020). The central question of this thesis, and arguably the central question that should be encompassed in all future research, is whether FFMs can be used operationally within training program design frameworks. To make steps toward answering this question, approaches used to model evaluation in experimental work must be addressed, more consistently highlighted as a key aspect of FFM experimental research, and available resources improved. In chapter 2, cross-validation (CV) was briefly examined, and terms such as "in-sample" and "out-of-sample" were described. Refreshing these concepts, cross-validation involves splitting the available data up in some way. A portion of the data is then used to train the model, and some are held back. The model is then asked to make predictions for that hold-back period (testing). The data that are held back are typically referred to as out-of-sample, or 'unseen', and similarly the data used to train the model are referred to as in-sample or 'seen'. The term data in an FFM context refers to timeseries data comprising both measured performance values (observations) and inputs (quantified training loads).

In the context of FFM's, it was also discussed in chapter 2 that out-of-sample data may be categorised further, dependent on whether the associated inputs (ω) were:

1. Quantified from a relatively homogeneous training program (i.e., similar to or from the one used to train the model)
2. Quantified from an evidently non-homogeneous training program (i.e., different to the one used to train the model). For example, from a future training block where the structure, shape, frequency is markedly different.

In the relation to FFM research, evaluation of model predictions against these two types of hold-back (out-of-sample) data serves slightly different purposes. The first provides an estimate of overfitting in the model, the latter provides the crucial insight into the central question of whether fitted FFMs can

be used to derive future training programs. Arguably, there is no reason beyond a lack of available resources and knowledge as to why future experimental research cannot assess both.

In chapter 2, it was highlighted that cross-validation approaches such as *k*-fold cross-validation and bootstrap resampling are not suitable for time-series data, as they assume there is no relationship between observations (independence), which is almost never true in this context. It is better to apply approaches that respect the temporal order, and one method forms the focus of this subsection and will be described shortly.

First consider simple a 'hold-out' cross validation approach (as it was described in chapter 2), in the case where data are only available for one training program. In other words, all of the out-of-sample data that can be made available comes from a homogeneous training program to the one used to fit the model. If hold-out cross validation were to be applied, choosing where to split the timeseries will likely affect the results (estimation of model prediction errors on out-of-sample data). To avoid this, and to achieve a more robust estimation of model performance, a walk-forward approach may be more preferential (de Prado, 2018). In a walk-forward approach, multiple splits are made across the timeseries with the amount of data used to train the model incrementally increasing (expanding) at each split. The results are then averaged across the splits for a chosen metric such as the mean-average-percentage-error (MAPE) or root-mean-squared-error (RMSE). In some cases, an 'optimal' set may be extracted based on either best model fit (e.g., the lowest objective value in minimisation), or more routinely, the best predicting set (e.g., the set with the lowest associated MAPE or RMSE value on out-of-sample data). In the more advanced case where training and performance data from a subsequent (consecutive) training program exists (and this program is dissimilar from the first (block 1)) the researcher may also wish to fit a model to the entirety of the block 1 data, and test resulting predictions computed up to the end of the consecutive block 2 loads, against block 2 performance data. In this manner, both model overfit is assessed as well as the counterfactual properties of the model, e.g., whether the model can accurately predict what happens under a different set of circumstances (loadings). Collectively, this approach is a more robust method of model evaluation than previous in-sample only testing adopted in the literature (see Table B-2, appendix B), with the exception of a few more recent studies that have included some form of cross-validation or out-of-sample assessment (Chalencon *et al.*, 2015; Ludwig, Schaefer and Asteroth, 2016; Kolossa *et al.*, 2017; Turner *et al.*, 2017; Williams *et al.*, 2018; Stephens Hemingway *et al.*, 2019; Imbach *et al.*, 2020).

Implementing an expanding-window walk forward type method is the practical focus of this subsection, to put researchers in a position where a clear and flexible resource exists for implementing this type of approach and can serve as a template and educational tool. The VDR model is used, although given the extensiveness of subsections previously it is hoped that the reader can by now adjust

this function further (and other code as required) to incorporate aspects such as a fitness-delay. The walk-forward method implemented is illustrated graphically below in Figure 6.15 (from chapter 2).

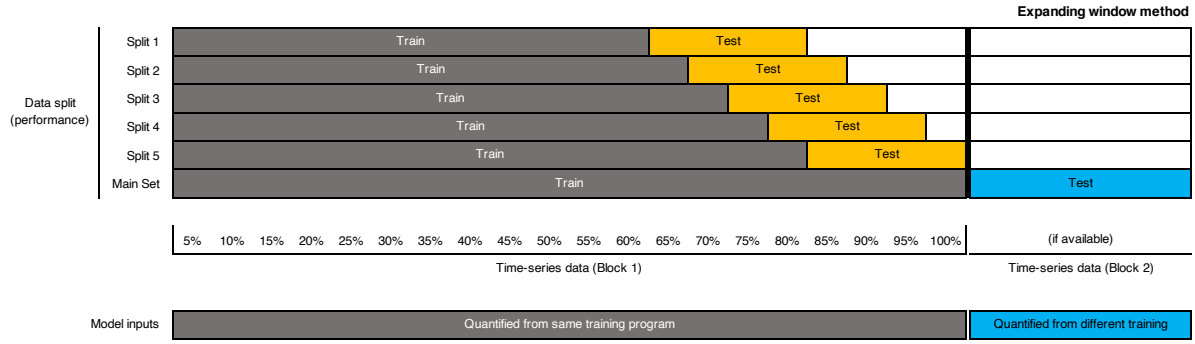


Figure 6.15: An illustration of the expanding-window walk forward CV method (From chapter 2)

As a first step in this implementation, synthetic data are developed representative of the general structure and type that researchers are expected to be working with. This is presented first so readers are aware of the structure of the data required as input to the functions developed shortly.

Rather than develop a dataset to illustrate the process in a freehand manner, say by manually entering reasonable values in a spreadsheet package, it was simpler and more reproducible (for the reader) to follow the previous sections and use the model simulation functions developed so far to a set of generate performance values under two different load series (representing different training programs). It is also important to recognise that the purpose of any synthetic data developed here - and in previous subsections of this chapter - is to illustrate the control flow of the fitting and evaluation approach. The results of the process, particularly predictive validity (or even ability for the optimiser to return the true parameters) are not under investigation or of interest in this section. In a similar fashion, the optimisation algorithm used is nonspecific in its selection, and could be swapped out for any of those described in section 6.2.4 (or other) with some modification. Although the method of synthetic data development will be outlined in detail, readers are advised not to become concerned with the minutia of this aspect, as the focus of this section is on the subsequent CV implementation and establishing a flexible framework that can be applied when real world training and performance data are held. In simple terms, the most important aspect for the reader is the manner in which the data are structured as input to the fitting function, rather than the values of the data. Also note that functions developed for model fitting under expanding-window CV will be rigid in the sense that they will not be able to handle data that isn't passed in the same format as described shortly. It was considered that input checking could be added to these functions, and flexibility to handle different formats, but this would have increased the size of the code substantially decreasing readability and diluting the focus of this subsection. The code is also written in a modular fashion to that several preliminary procedures are

carried out by subprograms (separate functions) to avoid the main function becoming excessively crowded and masking the overall control flow.

A timeseries dataset of 150 days in length was developed, split into two consecutive blocks of 100 and 50 days (block 1 and block 2, respectively). Each block comprises a series of training load values and associated performance measurements generated via simulation. The load series of block 1 ($B1$) and block 2 ($B2$) are distinct in shape, pattern, frequency, and magnitude. To generate associated performance measurements linked to these training load values, for each block, the VDR model was simulated under a set of pre-determined 'true' parameter values under consecutive loads $\omega = \{\omega_{B1}, \omega_{B2}\}$. The true parameter values selected were arbitrary in that they solely based on their rough ability to produce a realistic profile, and that the τ parameters were not physiologically unrealistic. A moderate value of parameter τ_{h_2} was chosen to model a relationship between previous loads, reflected in the performance profile. The values selected for the true parameters are clearly stated within the code of listing 6.29 on the following page.

The VDR model simulation function developed so far in section 6.2.2 generates performance at 1-day time steps, so some datapoints were removed in a random fashion to reflect a slightly less regular and lower-frequency measurement pattern in the mock data; as would be expected in the real world. A total of 56 measurements over a 150-day period were extracted. The two blocks were again distinct in that synthetic performance measurements evolved from different training load distributions, reflecting the concept of a non-homogeneous training program in the subsequent training period (necessary for robust evaluation/estimation of model utility). However, as described previously the simulated data is arbitrary, in that the same set could have been constructed by random guesswork as its purpose is to illustrate the CV approach to follow. Figure 6.17 presents plots of the mock data developed in listing 6.29, and Figure 6.16 shows a print-out of the structure of the input data. In the subsequent implementation of the CV approach, it is block 1 that will be partitioned (split) into train and test sets as required to estimate model uncertainty, and block 2 data will be retained as the testing data for future performance under a different load distribution to the one used to train the model.

Code from this section is available to download directly from:
github.com/bsh2/thesis/c6/cross_validation.R

First, the following code was run to load the necessary prediction and objective functions into the R environment. Recall that these functions facilitate the mock data to be developed, and fitting of the VDR model. Specifically, the functions `simulateVDR2` and `vdrObjectiveLL` developed in subsection 6.2.2 were loaded. Recall that `vdrObjectiveLL` is based on a maximum likelihood estimation approach.

Listing 6.29 Developing mock data used to demonstrate the cross-validation implementation

```
block1_loads <- rep(c(1, 1.2, 0.5, 1.8, 2, 0.25, 0.7,0.9, 0, 0.5, 1, 0.8,
                      1.2, 1.3, 0.9, 0, 0, 2, 1.1, 0.5), 5)
block2_loads <- round(abs(rnorm(50, 1.3, 0.5)),1)

loads <- data.frame("day" = 0:150,
                    "load" = c(0, block1_loads, block2_loads),
                    "block" = c(rep(1, 101), rep(2, 50)))

# Generating two blocks of associated performance data
true_parameters <- c(95, 0.85, 26, 1.2, 5, 1) #c(p*,kg,Tg,kh,Th,Th2)
performances <- simulateVDR2(true_parameters, loads = loads)
performances$block <- loads$block[2:151]

# Reduce measurement frequency and regularity
set.seed(101)
performances <- performances[c(1, sort(as.integer(sample(2:149, 56), replace = FALSE),
                                       decreasing = FALSE), 150), ]

# Develop some bounds for the parameter space
bounds <- data.frame("lower" = c(80, 0.1, 2, 0.1, 2, 0.2, 0.2),
                     "upper" = c(120, 3, 50, 3, 50, 3, 5))

# Collate the data (input format for the CV function)
dat <- data.frame("day" = loads$day,
                  "load" = loads$load,
                  "performance" = rep(NA, length(loads$load)),
                  "block" = loads$block)
dat$performance[performances$day + 1] <- performances$performance
```

```
> head(dat)
  day load performance block
1   0  0.0          NA     1
2   1  1.0    95.00000     1
3   2  1.2          NA     1
4   3  0.5          NA     1
5   4  1.8    94.13314     1
6   5  2.0          NA     1
```

Figure 6.16: Structure of the input data developed in listing 6.29

[NA values are used to indicate no measurement was taken on a given day]

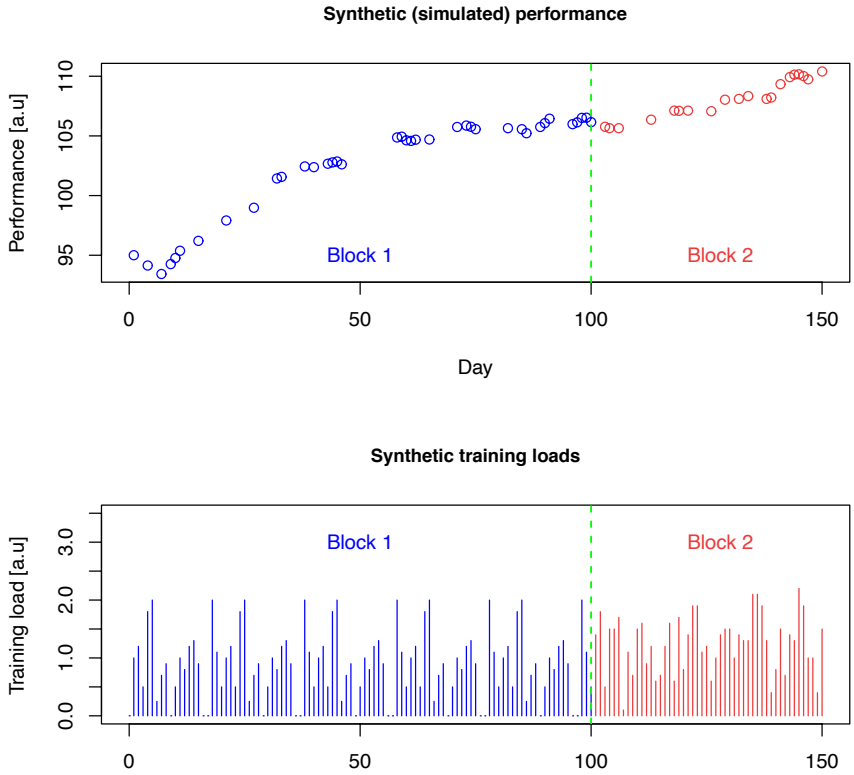


Figure 6.17: Synthetic data developed to demonstrate the cross-validation approach (plotted)

The process of implementation of the expanding window method is now followed, beginning with an overview of parallelisation in R followed by presentation of ancillary functions developed, and then finally discussion of the main control flow (function). These ancillary functions take care of the smaller tasks (procedures) that will be executed within the main function, including:

- Calculating the mean average percentage error (function `mape`)
- Creating a grid of random starting values for the algorithm (function `create_grid`)
- Generating the time-series splits (function `generate_splits`)
- Executing the train-test splits (function `train_test`)

Furthermore, the code presented shortly relies on several R packages available from the comprehensive R archive network (CRAN). These include some packages necessary for the train-test splits to be carried out in a parallelised manner, maximising available compute resources and improving time efficiency of the overall process. This aspect of computation become more important when alternative algorithms are used that often require a high number of iterations to find good solutions (e.g., derivative-free methods and derivative-free methods that also include a local search component). The package dependencies for this section are shown in Table 6.3 on the following page, and these must be installed and loaded into the environment for the code to work.

Table 6.3: Package dependencies for the cross-validation approach

Package	Purpose / provides	Install packages	Load packages
optimx	L-BFGS-B algorithm with multi-start (facilitates repeated fitting from a grid of starting parameters)	`install.packages("optimx", "caret", "RcppAlgos", "doSNOW", "foreach")`	`library(optimx, caret, RcppAlgos, parallel, doSNOW, foreach)`
caret	Contains a useful function that we make use of that takes care of splitting up the data (`createTimeSlices`)		
RcppAlgos	Contains a useful function (`comboSample`) that allows us to randomly sample from a vector of values over the bounds		
parallel	Specifying the socket cluster		
doSNOW	Registering the cluster (parallel backend)		
foreach	Parallelised loop for training and testing the splits at the same time.		

Parallel computing involves the specific use of computing resources to perform multiple tasks simultaneously, that otherwise would normally be carried out in a sequential fashion. Imagine the real-world example of a landscaper that has 10 holes to dig in the ground. They can only ever dig one hole at a time (sequentially). However, if they were to recruit 9 other landscaper (workers), and direct each of them to the remaining holes (one each), the process could be sped up significantly as all the holes would be dug in parallel. This crude metaphor describes what happens when we parallelise a computation or task. A master worker organises and distributes the tasks to a group of other workers that perform their required duty, return the results back to the master, and collect a new task (if available) until all are complete. In some cases, these workers are referred to as nodes. The CPU (central processing unit) is at the heart of every computer. Each CPU will typically have multiple cores (e.g., dual or quad-core). In simplistic terms, each core can carry out one task, so in a quad-core processor there are four workers available. If more than one machine (physical or virtual) is combined, this is known as a cluster, and the number of available workers can be expanded from a handful to over a thousand. In its default state, R only ever initiates one process on a single core to perform a task, known as single threading. In normal use cases this is preferable as many users of R will often run several other applications at a given time. However, when performing many iterative tasks that individually or collectively require a substantial amount of time, parallelisation provides one way of reducing the overall time required if these tasks or iterations can be easily distributed. In some cases, problems cannot be parallelised and are often referred to as "inherently serial" (e.g., when one task is dependent on the output of another). In others, they may be described as "perfectly parallel" (e.g., when no dependency between tasks exists and one can start and end at any point regardless of the status of another). There are several packages in R that provide facilities to parallelise operations, with their function depending on the method that they use. In general, there are two methods of parallelisation: 1) sockets; and 2) forking. A socket approach creates a new instance of R on each core and uses principles of networking (just in an internal sort of manner on the machine). A forking approach copies the version of R as it is currently exists, placing it on the new core. There are several advantages and disadvantages to both of these approaches, but for the purposes of this work we will focus on the socket method because it works on any operating system (e.g., windows, Linux, MacOS). The forking method does not work on windows, and thus is not the most useful approach to demonstrate. However, if the reader is working on a non-windows system then forking may prove a more efficient approach and worthwhile investigating depending on use-case.

The parallel package in R is the amalgamation of two predominant former packages (*SNOW*, and *multicore*) that provide a parallel backend. We will also make use of the package *foreach*, that provides a looping construct similar to the normal for loop in R, but that also supports parallel execution. However, to use the foreach package, we must first set up the parallel backend. A SNOW-like parallel backend for the foreach package is most versatile for the following implementation as it works on both

windows and unix-like systems. To create the parallel socket cluster and register the SNOW parallel backend we make use of the functions `makeCluster` and `registerDoSNOW` from the packages *parallel* and *doSNOW* (Weston and Computing, 2015) respectively. In addition, the function `detectCores` from the parallel package can be used to find out how many cores the machine has available. Briefly, logical cores are the number of physical cores multiplied by the number of threads that can run on each core (via a concept called hyperthreading). If a core can run two threads for example, and we have two cores, there are $2 \times 2 = 4$ logical cores available. The function `detectCores` provides the number of logical cores by default. In summary, the cluster is created and registered as follows:

```
cores <- detectCores(logical = TRUE)            # Determine no. cores
cluster <- makeCluster(cores, type = “SOCK”)    # Specify the socket cluster
registerDoSNOW(cluster)                         # Register the cluster/backend
```

In the above example code, we have created copies of R running in parallel on the specified number of cores, that are communicating via sockets. This gives us the parallel backend set up ready for use with the parallel foreach loop. The foreach loop is then declared via the following basic structure:

```
foreach(i = 1:5) %doPar% { sqrt(i) }      # Parallelised foreach loop (example)
```

Depending on the task, it is also sometimes necessary to export objects from the master instance and other required packages to the available nodes on the parallel backend. This can be done during the `foreach` call through the `.export` and `.packages` arguments, respectively (see help files of the package: run `?foreach`). Any object passed as an argument in a function within the foreach loop is automatically exported to the workers. We also need to stop the cluster when we are done with it, as follows:

```
stopImplicitCluster(cluster)                    # Stop the cluster (example)
```

In the context of the cross-validation approach, it is helpful to parallelise the train-test splits, particularly given that this process will be repeated several times for each split as the algorithm must retrain the model from multiple starting points to reduce starting point sensitivity affecting the obtained solutions. This is discussed shortly, and highlighted in the algorithm written in pseudocode, provided in Appendix C-2. Parallelisation in the context of fitness-fatigue modelling, as this thesis has shown, has multiple applications (computational experimentation under simulation, model fitting and cross validation), and may be useful when applying multiple rounds of evolutionary algorithms each requiring a high number of iterations due to the differences in termination criteria. This aspect of implementation may also have uses within hierarchical modelling, where models are trained on data from multiple athletes and train-test splits take a long time to run.

Given that parallelisation via foreach has been discussed, the ancillary functions described earlier are now presented. These will be used in the main CV function (control flow) for the approach. The first function that is straightforward and requires no further explanation beyond the code is one to compute the mean average percentage error (MAPE) from two vectors of measured and model predicted performance. Recall that the MAPE function is as follows:

$$MAPE = \left(\frac{1}{n}\sum_{k=1}^{n}\frac{|\widehat{p_k} - p_k|}{p_k}\right) * 100 \tag{6.35}$$

Listing 6.30 Function to compute the mean-average-percentage-error (MAPE) from two vectors of measured and predicted performance values

```
mape <- function(measured, predicted){
  return(mean(abs((measured - predicted)/measured))*100)
}
```

Next, the function `generate_splits` is presented to derive the index values associated with the train-test splits over a set of data. To complete this task, we make use of the package *caret* (Kuhn *et al.*, 2020), that provides a helpful function `createTimeSlices` for splitting up data into train and test sets that move in time. In the context of creating expanding window test-train splits, the function `createTimeSlices` takes as input: 1) the time-series data; 2) the initial number of consecutive values in each training set sample (`initialWindow`); 3) the number of consecutive values in a test set sample (`horizon`); 4) whether all samples begin from the first element (`fixedWindow = FALSE`); and 5) a value reflecting by how much each sample should expand over the time series (`skip`). Returned by the function is a list containing the index values relating to the original data set defining each train and test split. At this point, the data itself hasn't been isolated or split up into separate objects.

The `generate_splits` function developed below, that makes use of `createTimeSlices` (from *caret* package), itself takes as input a set of arguments that define the required values described above, but as percentages, converting them to days (consecutive values) before passing them on to `createTimeSlices`.

Listing 6.31 Function to develop expanding window splits

```
generate_splits <- function(initialWindow, testHorizon, expandRate, dat){
  # Convert supplied arguments for CV into 'days'
  initialWindow = round(length(dat$day) * initialWindow/100, 0)
  testHorizon = round(length(dat$day) * testHorizon/100, 0)
  expandRate = round(length(dat$day) * expandRate/100, 0)
  # Create splits
  splits <- createTimeSlices(dat$day,
                             initialWindow = initialWindow,
                             horizon = testHorizon,
                             fixedWindow = FALSE,
                             skip = expandRate)
  return(splits)
}
```

The next function developed below creates a grid of starting values by randomly sampling from inside the defined bounds of the parameter space (`nStarts` times). We move slightly inside the bounds for this sampling process to avoid starting the algorithm on the bound of any particular parameter, which can sometimes cause strange behaviour. The function makes use of the `comboSample` function from the `RcppAlgos` package that provides high performance tools for combinatorics. There are multiple possible implementations for this sampling in R, but this is just one simple method. The function takes as input the bounds and the number of sets to generate and returns a matrix of parameter values (the combination of each element in a row of the matrix represents a set).

Listing 6.32 Function to create a random grid of starting parameters across the bounds $[l, u]$ to be used for iterative model fitting from multiple starting points

```
create_grid <- function(bounds, nStarts, initial){

  if (initial == FALSE){
    set.seed(101)
    parmat <- data.frame(
      "p0" = comboSample(seq(bounds$lower[1] + 1, bounds$upper[1] - 1, 2.5),
                         m = 1, n = nStarts),
      "kg" = comboSample(seq(bounds$lower[2] + 0.1, bounds$upper[2] - 0.1, 0.1),
                         m = 1, n = nStarts),
      "Tg" = comboSample(seq(bounds$lower[3] + 1, bounds$upper[3] - 1, 1.5),
                         m = 1, n = nStarts),
      "kh" = comboSample(seq(bounds$lower[4] + 0.1, bounds$upper[4] - 0.1, 0.1),
                         m = 1, n = nStarts),
      "Th" = comboSample(seq(bounds$lower[5] + 1, bounds$upper[5] - 1, 1.5),
                         m = 1, n = nStarts),
      "Th2" = comboSample(seq(bounds$lower[6] + 0.1, bounds$upper[6] - 0.1, 0.1),
                          m = 1, n = nStarts),
      "sigma" = comboSample(seq(bounds$lower[7] + 0.1, bounds$upper[7] - 0.1, 0.1),
                            m = 1, n = nStarts)
    )  # We also add or subtract a little to get away from starting on the bounds
  }

  if (initial == TRUE){
    set.seed(101)
    parmat <- data.frame(
      "p0" = comboSample(seq(bounds$lower[1] + 1, bounds$upper[1] - 1, 2.5),
                         m = 1, n = nStarts),
      "kg" = comboSample(seq(bounds$lower[2] + 0.1, bounds$upper[2] - 0.1, 0.1),
                         m = 1, n = nStarts),
      "Tg" = comboSample(seq(bounds$lower[3] + 1, bounds$upper[3] - 1, 1.5),
                         m = 1, n = nStarts),
      "kh" = comboSample(seq(bounds$lower[4] + 0.1, bounds$upper[4] - 0.1, 0.1),
                         m = 1, n = nStarts),
      "Th" = comboSample(seq(bounds$lower[5] + 1, bounds$upper[5] - 1, 1.5),
                         m = 1, n = nStarts),
      "Th2" = comboSample(seq(bounds$lower[6] + 0.1, bounds$upper[6] - 0.1, 0.1),
                          m = 1, n = nStarts),
      "sigma" = comboSample(seq(bounds$lower[7] + 0.1, bounds$upper[7] - 0.1, 0.1),
                            m = 1, n = nStarts),
      "qg" = comboSample(seq(bounds$lower[8] + 0.1, bounds$upper[8] - 0.1, 0.1),
                         m = 1, n = nStarts),
      "qh"= comboSample(seq(bounds$lower[9] + 0.1, bounds$upper[9] - 0.1, 0.1),
                        m = 1, n = nStarts)
    )  # We also add or subtract a little to get away from starting on the bounds
  }
  return(as.matrix(parmat))
}
```

Notes (listing 6.32):

- `initialWindow` is the size of the training set in the first split (supplied by user as % of the timeseries, default 60%)
- `testHorizon` is the size of the testing set in a given split (supplied by user as % of the timeseries, default 20%)
- `expandRate` is the size at which the training data expands over the timeseries from each previous split (supplied by the user as a % of the timeseries, default 4%)

Before the main control function is developed, a final ancillary function (`train_test`) is presented on the following page, that is called within the main control flow to process the train-test splits (i.e., fit, evaluate) iteratively in a parallelised for-loop. This function takes as input the required data, a list containing the index values of all the splits, a reference value for the current split (i.e., the index of the loop), and a set of starting values and bounds for the optimisation algorithm. Returned as output is a list that contains the fitted parameter values from each set of starting values along with associated out-of-sample test scores ($MAPE_{TEST}$). Also returned is the average performance (average of $MAPE_{(TEST)}$) across all iterations of the current split processed by the function. Readers may find it most helpful to refer to the algorithm in Appendix C-2 at this point, to clarify the exact control flow of the implementation in its entirety prior to presentation of the main function next.

Listing 6.33 Function to train/test the VDR model for a given split. The model is repeatedly trained/tested under different algorithm starting points for the L-BFGS-B minimiser

```
train_test <- function(dat, parmat, bounds, main, splits = NA, currentSplit = NA, initial){

  if(main == FALSE){
    # If we are training-testing on splits vs. training on the whole of block 1
    training_data <- dat[splits$train[[currentSplit]], ]
    testing_data <- dat[splits$test[[currentSplit]], ]}

  if(main == TRUE){
    # If training on the whole block 1 and testing on block 2
    training_data <- dat[dat$block == 1, ]
    testing_data <- dat[dat$block == 2, ]
  }

  # Isolate a vector of days on which measurements exist for train and test data
  measure_idx_train <- subset(training_data$day, !is.na(training_data$performance))
  measure_idx_test <- subset(testing_data$day, !is.na(testing_data$performance))

  # Isolate a vector of measurements in for train and test data
  measurements_train <- subset(training_data$performance, !is.na(training_data$performance))
  measurements_test <- subset(testing_data$performance, !is.na(testing_data$performance))

  # Put data in required format for objective function vdrObjective()
  dat_temp <- data.frame("day" = measure_idx_train, "performance" = measurements_train)
  load_temp <- training_data[, c("day", "load")]

  # Fitting iterations
  fittedModel <- optimx::multistart(parmat,
                                    fn = vdrObjectiveLL,
                                    lower = bounds$lower,
                                    upper = bounds$upper,
                                    method = "L-BFGS-B",
                                    control = list(maxit = 500,
                                                   trace = FALSE),
                                    loads = load_temp,
                                    perfVals = dat_temp,
                                    initial = initial)

  # Compute predicted performance for the entire time-series (blocks 1 + 2)
    temp_predictions <- sapply(1:dim(parmat)[1],
           function(i) simulateVDR2(pars = as.numeric(fittedModel[i, 1:6]),
                                    loads = dat[,c("day", "load")],
                                    if (initial == TRUE){
                                      initialPars = as.numeric(fittedModel[i, 8:9])
                                     } else {
                                      initialPars = c(0,0)
                                     })$performance)
  # Extract predictions at required days to evaluate model performance
  predictions_training <- temp_predictions[measure_idx_train, ]
  predictions_testing <- temp_predictions[measure_idx_test, ]

  # Compute metrics (MAPE)
  mape_train <- sapply(1:dim(predictions_training)[2],
                       function(i) mape(measured = measurements_train,
                                        predicted = predictions_training[,i]))
  mape_test <- sapply(1:dim(predictions_testing)[2],
                      function(i) mape(measured = measurements_test,
                                       predicted = predictions_testing[,i]))

  # Collect these metrics and calculate average split statistics
  fittedModel <- cbind(fittedModel, mape_train, mape_test)
  stats <- c("mean_mape_train" = mean(mape_train),
             "mean_mape_test" = mean(mape_test),
             "sd_mape_train" =  sd(mape_train),
             "sd_mape_test" = sd(mape_test))

  # Develop output
  output <- list("fittedModel" = fittedModel, "predictions" = temp_predictions,
                 "stats" = stats)
  return(output)
}
```

Finally in developing the CV approach, the main function (control flow) for the expanding-window CV method can be defined (listing 6.34).

Listing 6.34 Main cross-validation function: Expanding-window method (VDR FFM) – Multistart L-BFGS-B

```
expandingWindow_CV <- function(dat,
                               bounds,
                               initialWindow = 60,
                               testHorizon = 20,
                               expandRate = 4,
                               nStarts = 10,
                               cores = NULL,
                               initial = FALSE){

  # Generate splits (note split vectors gives you an index position vs. a 'day', think t-1!)
  splits <- generate_splits(initialWindow, testHorizon, expandRate, dat[dat$block == 1, ])
  nSplits <- length(splits$train)   # Number of splits

  # Create an array of random starting values over the bounds
  parmat <- create_grid(bounds, nStarts, initial = initial)

  # Iterate over the splits (train-test)
  if (is.null(cores)){
    cores <- detectCores(logical = TRUE)
  } # If cores not specified by user
  cl <- makeCluster(cores, type = "SOCK")              # Make cluster
  registerDoSNOW(cl)                                  # Register cluster
  fitted_splits <- foreach(i = 1:nSplits, .verbose = TRUE, .packages = c("optimx"),
                           .export = c("train_test", "vdrObjectiveLL", "simulateVDR2",
                                       "mape")) %dopar%{
                                         train_test(dat, parmat, bounds,
                                                    main = FALSE, splits = splits,
                                                    initial = initial,
                                                    currentSplit = i)}
  # By default results returned as a list
  stopCluster(cl)                                      # Stop cluster
  names(fitted_splits) <- paste0("split_",1:nSplits) # Add names to the list for each split

  # Compute model performance across splits and add to the existing list object
  mape_train_across <- matrix(NA, nrow = nStarts, ncol = nSplits)
  mape_test_across <- matrix(NA, nrow = nStarts, ncol = nSplits)
  for (i in 1:nSplits){
    mape_train_across[,i] <- fitted_splits[[i]]$fittedModel$mape_train
    mape_test_across[,i] <- fitted_splits[[i]]$fittedModel$mape_test
  }
  fitted_splits$across_splits <- list("training_mape" = c("mean" = mean(mape_train_across),
                                                          "sd" = sd(mape_train_across)),
                                      "testing_mape" = c("mean" = mean(mape_test_across),
                                                         "sd" = sd(mape_test_across)))

  fitted_splits$starting_pars <- parmat

  # Fit the model to all block 1 data, and test on block 2 (Main train/test split)
  mainModel <- train_test(dat, parmat, bounds, main = TRUE, initial = initial)

  # Extract the best set by lowest -logLik value and by prediction on test set
  mainModel$bestSet$fit <- mainModel$fittedModel[mainModel$fittedModel$value ==
                                                   min(mainModel$fittedModel$value), ]
  mainModel$bestSet$test <- mainModel$fittedModel[mainModel$fittedModel$testing_mape ==
                                                    min(mainModel$fittedModel$mape_test), ]

  fitted_splits$main <- mainModel
  fitted_splits$splits <- splits
  fitted_splits$nSplits <- nSplits
  fitted_splits$nStarts <- nStarts

  # Output results
  return(fitted_splits)
}
```

Notes (listing 6.34):

- `initialWindow` is the size of the training set in the first split (supplied by user as % of the timeseries, default 60%)
- `testHorizon` is the size of the testing set in a given split (supplied by user as % of the timeseries, default 20%)
- `expandRate` is the size at which the training data expands over the timeseries from each previous split (supplied by the user as a % of the timeseries, default 4%)
- Argument `cores` allows the user to specify the number of cores for to use in the parallelisation of the train-test splits (supplied as a scalar, integer) - Defaults to all available cores.
- Argument `nStarts` is a scalar (integer) specifying the number of starting sets to use in the optimisation process for the L-BFGS-B algorithm.
- Data objects supplied to argument `dat` should be in the format described toward the beginning of this subsection when mock data was developed (see listing 6.29).
- Argument `bounds` should be supplied as a dataframe of two columns, as follows (where $l < u$):

`bounds <- data.frame("lower" = c(`$p^{*,(l)}, k_g^{(l)}, \tau_g^{(l)}, k_h^{(l)}, \tau_h^{(l)}, \tau_{h_2}^{(l)}, \sigma^{(l)}$`),`
`"upper" = c(`$p^{*,(u)}, k_g^{(u)}, \tau_g^{(u)}, k_h^{(u)}, \tau_h^{(u)}, \tau_{h_2}^{(u)}, \sigma^{(u)}$`)`

In listing 6.33, it is pointed out the function `multistart` in the package `optimx` (Nash et al., 2020) provides a simple way to repetitively fit a given model for multiple starting points, with the results of this process collected in a dataframe (lines 28-37). This function `train_test` also accounts for the case when you wish to fit to the entire block 1 data and test predictions on block 2, as will be demonstrated shortly using the mock data developed earlier; facilitated via the function argument `main = TRUE` assuming object passed to argument `dat` is the full dataset of both block 1 and block 2 data.

Finally in this subsection, and capping off section 6.2, this CV implementation is demonstrated using the synthetic data developed, and a basic set of plots that can be derived from this cross-validation process are then presented, similar to those in Turner *et al.* (2017). At this stage, if the user wishes to replicate this process in their own R environment, it is recommended they download the necessary code file (github.com/bsh2/thesis/c6/cross_validation.R) to avoid having to copy out the code as it has been listed so far. Given the mock data `(dat)` developed so far in listing 6.29, and the bounds given below, calling the cross-validation process for this data is now as simple as (whilst assuming default settings for non-declared arguments, but increasing the number of random starting parameters per train-test split to 100):

Listing 6.35 Demonstrating the cross-validation method (function call) for fitting the VDR model

```
1  example <- expandingWindow_CV(dat = dat,
2                                bounds = bounds,
3                                initialWindow = 60,
4                                testHorizon = 20,
5                                expandRate = 4,
6                                nStarts = 10,
7                                cores = 1)
```

The output of this function, assigned to `example`, is discussed next and then plots developed to visualise the results.

Output from the cross-validation function: Returned from `expandingWindow_CV` is a list containing

- A further list `example$split_i` (one for each split $i = 1 - N$) containing:
  - A dataframe of fitted parameters and associated measures of model error from the multiple fitting iterations (from different starting points)
  - A dataframe of model simulated performance values over the whole length of the input data (`dat`) for each set of fitted model parameters
  - Average MAPE (training and test sets) across the fitted parameters
- A list containing the average MAPE for the train and test sets across all splits (`example$across_splits`)
- A matrix containing the starting parameter values used to fit the model for each train-test split (`example$starting_pars`)
- The index values used for each split over the input data (`example$splits`)
- The total number of train-test splits (`example$nSplits`)
- A list containing the results of training on the whole block 1 data, and testing on the whole of block 2 (`example$main`). See discussion of this concept at the start of the subsection. Contains all the same elements as `example$split_i`, as well as the *'optimal'* set across the iterations by objective value and $MAPE_{TEST}$.

To provide some ideas on how these results may be visualised, three sets of plots are presented on the following pages, developed from the key output of this cross-validation example under the synthetic data (listing 6.35). To begin with, in Figure 6.18, boxplots are presented to illustrate variation in parameter values and model errors within and across the splits (not including the 'main' split, i.e., train on block 1, test on block 2). Recall that within-split variation occurs because each test-train split comprises multiple runs of the fitting algorithm initiated from different starting points (the total number being specified by the user). The between-split variation occurs due to the natural differences in where the dataset is split (as discussed with the simple case of the hold-out approach earlier). In Figure 6.19, boxplots were also used to summarise the distribution of parameters for the main train-test split (i.e., train on all of block 1 data, test on block 2). In Figure 6.20, a time-series plot is developed that tracks all the fitted model predictions across all iterations for the main train-test split, revealing variation in the model's ability to predict unseen future response under a different input distribution. This plot is simply an overlaid line plot, but more sophisticated region (shaded) plots could be developed using the polygon function in base R's graphics (plotting) system. The code for figures 6.18, 6.19, 6.20 are available in the associated code file for this section.

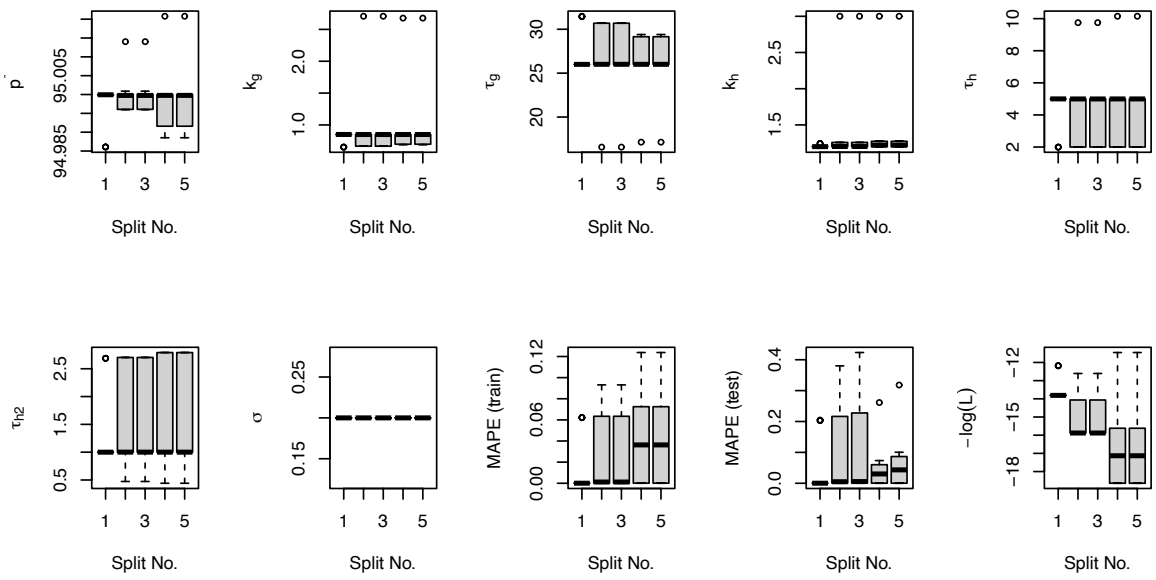


Figure 6.18: Boxplots summarising both fitted parameter and train/test prediction error variation across the expanding-window splits, within split variation is due to the multiple iterations from different start points

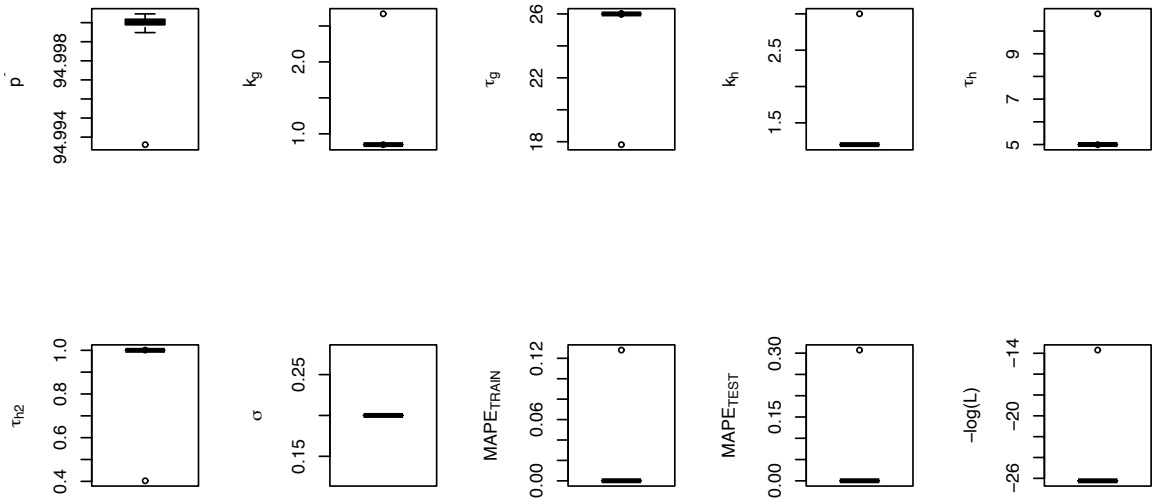


Figure 6.19: Similar boxplots to figure 6.18, summarising fitted parameter and train/test prediction error variation across the main train-test split (block 1 train, block 2 test), with the model training also comprising multiple fitting iterations from many starting points.

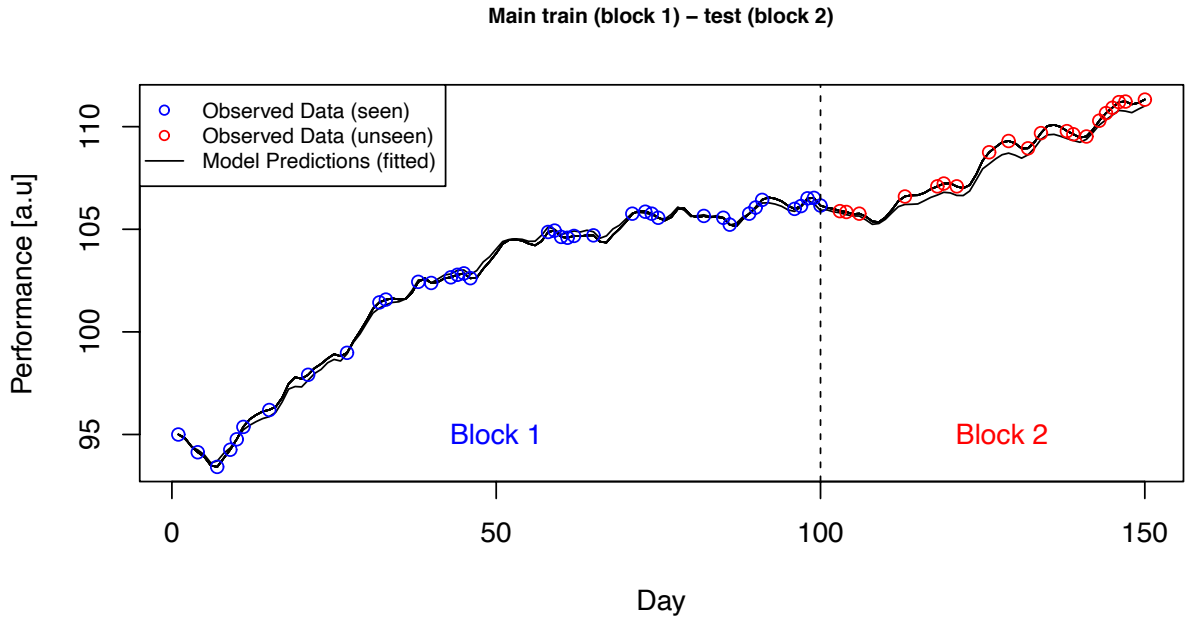


Figure 6.20: Plotting the variation in iterations (lines) from the main train-test split (train on block 1, blue points; test on block 2, red points) over the time-series

6.3 External saturation of model inputs: Considerations

Recall from chapter 2 that the external threshold function introduced by Hellard *et al.* (2005) to saturate the training response to increasing loads (external to the model structure) is described mathematically by the following function:

$$hill(\omega) = \kappa\left(\frac{\omega^{\gamma}}{\delta^{\gamma} + \omega^{\gamma}}\right) \tag{6.36}$$

The behaviour (shape) of this function depends on choice of its parameters κ, δ, γ. The parameter κ is the 'absolute' threshold level for which loads beyond this point will provide no further benefit, under the FFM mechanics described by the time-invariant parameters (k, τ) for each component. The parameters δ and γ control the shape of saturation profile, such as the sensitivity (γ) to a new training load, and inertia (i.e., resistance) of the function to (reaching) the threshold value. High values of the δ parameter have a suppressive effect on response across the curve. Collectively, the resulting shape (curve) of the function is affected by choice of these two parameters (δ, γ). To explore the behaviour of this function in R, it is helpful to define a simple function that takes as input these parameters and a vector of training loads, providing as output the saturated values (listing 6.36). Code for this section is available from github.com/bsh2/thesis/c6/threshold_saturation.R.

Listing 6.36	Threshold saturation function (simulation)

```
hillTransform <- function(kappa, delta, gamma, loads){
  hill_w <- kappa * (loads^(gamma) / (delta^(gamma) + loads^(gamma)))
  return(hill_w)
}
```

The `hillTransform` function in listing 6.36 can be evaluated for a vector of increasing loads, and called repeatedly in some iterative structure (e.g., a loop or apply) to examine several different values of the parameters δ, γ and produce plots illustrating the behaviour of this function for different choices of δ, γ. Such plots were first presented in Hellard *et al.* (2005) and recreated in chapter 2 at the point where the reader was introduced to this modification. They are recreated again here in figure 6.21 to remain self-contained in this section.

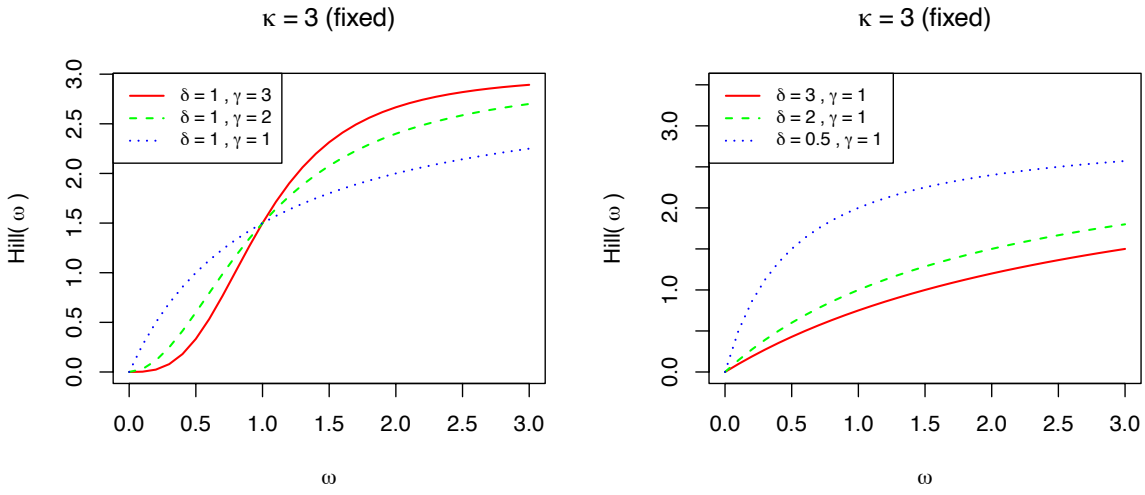


Figure 6.21: Hill-saturation for increasing loads under different values of δ, γ, with κ fixed

Although the behaviour of this hill function is relatively clear for different choices of the parameter values in figure 6.21, what remains less clear is how these parameter values should be chosen or estimated in practice. Selecting κ manually and not fitting it within an optimisation procedure is perhaps the most straightforward of the three parameters from a conceptual standpoint, as it represents the maximum load at which further responses will cause no additional increase. In effect it is the upper 'cap' on training load response to any single session. One possible approach in practice for deriving an estimate of κ is to have a coach elicit an estimate as follows: 1) the coach imagines the hardest possible training session (possibly informed from previous recorded data) that their athlete could tolerate before failure with respect to substantial breakdown in quality, ability, or performance in the movement and/or exhaustion, that would also result in extremely high fatigue at the expense of future training; 2) The session is then quantified within the training load framework and the resultant used as the value of κ. One possible issue that arises from discussion of κ is that the threshold for fatigue may differ to that of fitness, in particular it may be considerably higher as increases in training load may

still stimulate further increases in fatigue (leading to overtraining). This could necessitate the need for two separate hill functions, one for each component, increasing the number of free parameters in the problem even further and potentially exacerbating existing issues with overfitting and existence of local minima. As a starting point in an optimisation framework, appropriate bounding of these terms may provide some benefit to these issues. However, researchers could also seek to reduce the scope of the estimation problem further by developing methods to reduce the number of parameters that need to be estimated in the first place. For example, developing manual methods to derive fixed estimates of κ, δ, γ based on knowledge of the expected load-response profile. However, deriving either or both of the δ and γ parameters is seemingly less straightforward than κ, and it is perhaps inevitable that at least one of these will require some form of fitting to data. Based on these concepts, there are a few avenues that could be investigated in the further when considering the use of the saturation threshold function in research:

1) The FFM is fitted under hill transformed inputs with the parameters κ, δ, γ optimised by a search algorithm, and allowed to vary over:
   a. Relaxed bounds on each parameter (little consideration given)
   b. Tight bounds on each parameter based on consideration of the behaviour of the hill function (sensitivity, resistance) and likely values of κ

2) The FFM is repeatedly fitted under hill transformed inputs via fixed values of κ, δ, γ varying at each iteration over a grid. This approach can be thought of as forming an outer loop around the fitting process where at each iteration the hill parameters are fixed to a new set and the model then refit by some optimisation algorithm (which is itself an iterative loop). Selection of values carried forward may then be based on:
   a. The lowest achieved sum of squared errors over the grid and an inspection to check that this solution appears to provide a reasonable response saturation profile.

3) Estimation of κ, δ, γ within a Bayesian framework where priors are elicited for each parameter.

4) Fix one or more of κ, δ, γ via prior estimation, taking into account expected threshold and behaviour of the function, with any remaining parameters estimated within the FFM fitting process (either via approach 1 or 2 above). This reduces the scope of the fitting problem from 3 to 1-2 parameters per external saturation function.

It makes sense that option 1a would be expected to perform worst out of the four avenues presented as this level of complexity appears likely to induce too many local optima, although evocation of tighter bounds may help (1b). Option two theoretically reflects an improvement in terms of reducing the chance that the additional parameters disrupt the ability for the optimisation algorithm to find the best

solution for the FFM. Option three represents an interesting avenue of future investigation. Bayesian approaches may provide an effective means of bridging the gap between researchers and practitioners. Physical training is frequently referred to as a discipline that incorporates both science and art. Bayesian approaches provides a means to directly incorporate the intuitions and tacit knowledge of practitioners to the modelling process. Where practitioners are very confident of the adaptation response of a specific athlete, they can provide relatively tight priors, thereby strongly influencing the predictions of an FFM, particularly where the amount of performance data are limited. In contrast, where large amounts of data are present, priors will have less influence on the FFM predictions. Minimal alteration to the code presented in this thesis would be required to run Bayesian FFMs. Where research and processes are required to effectively implement Bayesian approaches includes the means of extracting appropriate priors for coaches and the extent to which these priors are univariate or multivariate. Option four represents an informal approach that crosses over into aspects of prior elicitation for Bayesian methods.

Discussed already, κ could be fixed based on an estimated derived from available data, coaching intuition, and quantified training loads. Not yet discussed is whether there is an approach or avenue of investigation for generating estimates of γ or δ or both based on intuition or measurement. Examining the behaviour of the hill function once more, we note that for certain values of γ, δ, the curve follows a sigmoidal 'S'-shape that goes from convex to concave at an inflection point. This is visible in the additional plots in figure 6.22 below, that demonstrate that as γ decreases from high values ($\gamma = 6$) to low values ($\gamma = 1$) when $\delta = 1$ is fixed ($\kappa = 3, \omega = 0.1 - 3$), the curve changes from S-shaped to concave over the whole domain (approximately when $\gamma < 2$). Also notable in figure 6.22 is that when δ is increased to 2, the same plots are flattened over the domain, with the inflection point also moving to the right (larger values of ω). Described conceptually, this sigmoidal shape for the load-response profile says that response starts off small for low values of ω (relative to κ), then enters a proportional (approximately linear) period for moderate loads, before saturating toward high values of ω. Such a functional form matches many biological phenomena and is likely to be of use in the context of fitness-fatigue modelling.

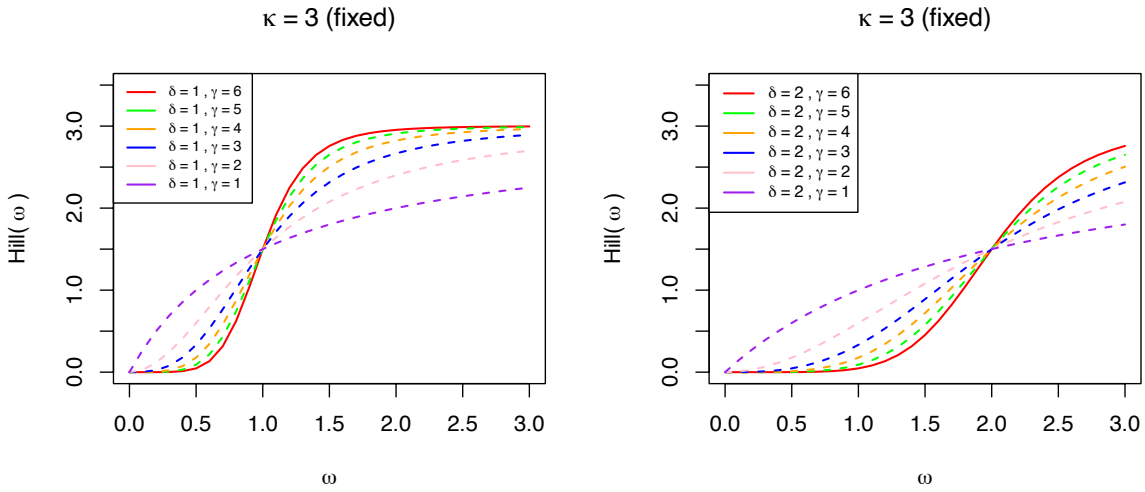


Figure 6.21: Left: Sigmoidal curve produced by the hill function for changing γ with δ fixed to 1. **Right**: Demonstration of the flattening of the curve and inertia of the hill function to the threshold value by increasing δ to 2 and reproducing the plot on the left with the same values of γ.

Given that the hill function can transform loads under a sigmoidal profile under the right conditions for γ and δ (for different κ and ω the values might differ to those used in the plots), visual inspection of the saturation function for varied values prior to optimisation may provide a simplistic approach for deriving estimates (or at least tighter bounds) for parameters γ or δ or both. For example, in the context of the loads used for the plots presented in figures 6.21 and 6.22, $\delta = 1$ could be fixed, such that it becomes more common for a given larger load (approximately loads above $\omega = 1.5$) to reach the maximum transformed value κ (also dependent on largeish γ). Alternatively, if $\delta = 2$ only very high loads will be transformed to values close to κ (but still far off). This is shown in figure 6.22. If δ cannot be fixed, it seems reasonable at the very least that tighter bounds could be established for the optimisation problem via this approach, for example $1 \leq \delta \leq 2$ in our example. Similarly, if a value of γ is fixed, or tight bounds are derived, such that a sigmoidal response curve for transformed loads is achieved, this might be done based on the behaviour of the curve after the inflection point. For lower values of γ the curve becomes flatter after this point, such that increasing loads still have relatively different and increasing transformed values. For higher values of γ, after the inflection point there is a steep portion of the curve that reaches a peak, beyond which transformed loads are relatively constant (similar). Much of the discussion here is theoretical and has not been tested under experimental conditions.

The aim of this section was to provide some initial options for researchers considering investigation of this saturation function. This area is in clear need of attention, as the potential benefit of the function (i.e,, introduction of non-linearity in response) is clouded by the complexity it adds to the optimisation problem.

6.4 State-space reformulation and Kalman filtering

Recall in chapter 2 it was shown that the standard FFM can be put into state space form, as first shown in Kolossa *et al*. (2017). The state variables (fitness and fatigue) at time $n+1$ could be described by:

$$\mathbf{x}_{n+1} = \mathbf{A}_n\mathbf{x}_n + \mathbf{B}_n\omega_n + \mathbf{v}_n \tag{6.37}$$

Where $\mathbf{x}_{n+1}$ is a vector comprising fitness and fatigue on day $n+1$ (and therefore $\mathbf{x}_n$ is the state vector comprised of fitness and fatigue at time n), $\mathbf{A}_n$ is a diagonal 2×2 "transition matrix" of coefficients that multiply the current fitness and fatigue values, $\mathbf{B}_n$ is a 2×1 matrix of coefficients that multiplies the scalar training input ω, and $\mathbf{v}_n$ is the state noise quantified by a 2×2 covariance matrix that is generally donated by $\mathbf{Q}$. The state noise describes the random change in state (e.g., fitness and fatigue) above and beyond the deterministic component involving training and past fitness or fatigue.

The state ($\mathbf{v}_n$) and observation (η_n) error terms are distributed independent and identically Gaussian distributed. Matching these terms to the standard FFM, these matrices are described as follows:

$$\mathbf{x}_n = \begin{pmatrix} g(n) \\ h(n) \end{pmatrix} \qquad (state\ variables\ vector \mid 6.38)$$

$$\mathbf{v}_n = N{\sim}(\mathbf{0}, \mathbf{Q}_n), \qquad \eta_n{\sim}N(0, \xi^2) \qquad (state\ and\ observation\ error \mid 6.39)$$

Therefore,

$$\mathrm{Var}(\mathbf{v}_n) = \mathbf{Q}_n = \mathbf{Q} = \begin{pmatrix} \sigma_g^2 & \sigma_{g,h} \\ \sigma_{g,h} & \sigma_h^2 \end{pmatrix} \qquad (Q\ matrix \mid 6.40)$$

Where the covariance $\sigma_{g,h}$ can be rewritten $\sigma_{g,h} = \sigma_g \cdot \sigma_h \cdot \rho_{g,h}$ where $\rho_{g,h}$ is the correlation between the state error term elements.

$$\mathbf{A}_n = \mathbf{A} = \begin{pmatrix} e^{-\frac{1}{\tau_g}} & 0 \\ 0 & e^{-\frac{1}{\tau_h}} \end{pmatrix} \qquad (transition\ matrix \mid 6.41)$$

$$\mathbf{B_n} = \mathbf{B} = \begin{pmatrix} e^{-\frac{1}{\tau_g}} \\ e^{-\frac{1}{\tau_h}} \end{pmatrix} \qquad (coefficient\ matrix \mid 6.42)$$

$$\mathbf{x}_0 = \begin{pmatrix} g(0) \\ h(0) \end{pmatrix} \qquad (initialisation\ vector \mid 6.43)$$

Where, $\mathbf{x}_0$ is treated as an unknown vector parameter, ω_0 is set to $\bar{\omega}$ and the matrix $\mathbf{M}_0 = Var(\mathbf{x}_0)$ is typically set to have large values on the diagonal to represent the uncertainty in the initial value.

The state of the system is not observed directly but is accessible by means of indirect measurement of performance p_n, from the equation:

$$\hat{p}_n = p^* + \mathbf{C}_n\mathbf{x}_n + \eta_n \qquad (system\ estimate \mid 6.44)$$

Where $\mathbf{C}_n = \mathbf{C} = \left(k_g, -k_h\right)$ a 1×2 matrix and η_n is the observed performance noise described by actual measurement errors with variance denoted by ξ^2 ("*xi*").

To modify the above state-space model to reflect the VDR model, the matrix $\mathbf{B}_n$ can be modified to:

$$\mathbf{B}_n = \begin{pmatrix} e^{-\frac{1}{\tau_g}} \\ k_{h_2}^i e^{-\frac{1}{\tau_h}} \end{pmatrix}, \qquad k_{h_2}^i = \sum_{j=1}^{i} \omega_j \cdot e^{\frac{-(i-j)}{\tau_{h_2}}} \qquad (VDR\ modification \mid 6.45)$$

Taking this a step further, the hill saturation function could also be included, as follows:

$$k_{h_2}^i = \sum_{j=1}^{i} \omega_h(j) \cdot e^{\frac{-(i-j)}{\tau_{h_2}}}, \qquad \omega_h(j) = \kappa_g \left(\frac{\omega_j^{\gamma_h}}{\delta_h^{\gamma_h} + \omega_j^{\gamma_h}} \right) \qquad (VDR\ modification + hill \mid 6.46)$$

Where $\kappa_h, \delta_h, \gamma_h$ are the hill model parameters for the fatigue state.

All code presented onwards in this section is available from **github.com/bsh2/thesis/kalman_filter.R** First, a function is developed to instantiate a state-space FFM from a set of parameters and initial conditions (Listing 6.37).

Listing 6.37 Defining a function to instantiate a state-space model from parameters

```
state_space_FFM <- function(pars){

  # Transition matrix
  A <- matrix(c(exp(-1 / pars$tau_g), 0, 0, exp(-1 / pars$tau_h)), ncol = 2)

  # State intercept
  B <- matrix(c(exp(-1 / pars$tau_g), exp(-1 / pars$tau_h)), ncol = 1)

  # Measurement matrix
  C <- matrix(c(pars$k_g, -1 * pars$k_h), ncol = 2)

  # Variances
  Q <- matrix(c(pars$sigma_g^2, rep(pars$rho_gh * pars$sigma_g * pars$sigma_h, 2),
                pars$sigma_h^2), ncol = 2)
  xi <- pars$xi

  # Prior distribution of fitness and fatigue (initial conditions)
  x_0 <- c(pars$g_0, pars$h_0)
  M_0 <- matrix(c(pars$sd_g0^2,
                  rep(pars$rho_gh0 * pars$sd_g0 * pars$sd_h0, 2), pars$sd_h0^2),
                ncol = 2)

  model <- list(A = A, B = B, C = C, Q = Q, xi = xi, x_0 = x_0,
                M_0 = M_0, p_star = pars$p_star)

  return(model)
}
```

The named list `pars` should contain an element for each of: $\left(p^*, k_g, k_h, \tau_g, \tau_h, \xi, \sigma_g, \sigma_h, \rho_{g,h}, g_0, h_0, \sigma_{g_0}\ \sigma_{h_0}, \rho_{(g,h)_0}\right)$

Training loads ω are taken to be exogenous (i.e., evolving from a predetermined plan rather than a downstream impact of performance). A simulation function can then be defined to evaluate the state-space model under a training load series and parameters. This simulation function is developed in listing 6.37 and makes use of the package *MASS* (Ripley *et al.*, 2013) to draw from a multivariate normal distribution for state noise.

Listing 6.38 Simulation function: State-space model under `loads` (`ss_model` argument obtained via `state_space_FFM` function under `pars`)

```
simulate_ss_model <- function(ss_model, loads){

  # Set up vectors and state matrix structures
  T <- length(loads$load)
  performance <- numeric(length(loads$load))
  X <- matrix(rep(NA, 2 * T), ncol = 2)

  # Simulate (note %*% is matrix multiplication operator in R)
  for (n in 1:T){
    # A priori mean and variance of state
    if (n == 1){
      X[n, ] <- ss_model$x_0  # Unconditional: x_0
    } else{
      # Conditional: x_n | x_(n-1)
      X[n, ] <- mvrnorm(1, (ss_model$A %*% X[n - 1, ] +
                            ss_model$B * loads$load[n - 1]), ss_model$Q)
      # Uses MASS package function mvrnorm(n, mu, sigma) to draw from
    }
    # Simulate conditional system state: p_n | x_n
    performance[n] <- rnorm(1, ss_model$p_star + ss_model$C %*% X[n, ], ss_model$xi)
  }

  simulation <- data.frame("day" = 1:T, "load" = loads$load, "performance" = performance,
                           "fitness" = X[,1], "fatigue" = X[,2])
  return(simulation)
}
```

Therefore, specifying a set of parameters (including for the noise terms and initial conditions), and applying the same loads from listing 6.29 (subsection 6.2.5), it is possible to apply these two functions sequentially to simulate the state-space model with random noise terms (listing 6.39).

Listing 6.39 Simulating the state-space FFM under mock data with random noise terms

```
loads <- rep(c(1, 1.2, 0.5, 1.8, 2, 0.25, 0.7, 0.9, 0, 0.5, 1, 0.8, 1.2, 1.3,
               3, 0.9, 0, 0, 2, 1.1), 5)
loads <- data.frame("day" = 0:length(loads), "loads" = c(0,loads))

pars <- list(p_star = 95, k_g = 0.85, k_h = 1.2, tau_g = 26, tau_h = 5, g_0 = 10,
             h_0 = 6, xi = 2.5, sd_g0 = 2, sd_h0 = 1, rho_gh0 = 0, sigma_g = 2,
             sigma_h = 1.2, rho_gh = 0.3)

# Instantiate Kalman model and simulate under the parameters
ss_model <- state_space_FFM(pars)
set.seed(109)
simulated_ss_model <- simulate_ss_model(ss_model, loads)
```

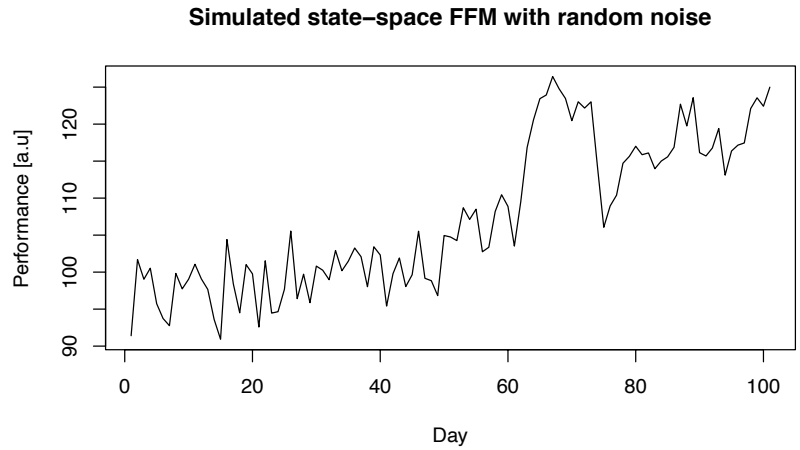


Figure 6.23: Simulated state-space FFM with random noise (listing 6.39, lines 10-12)

Next, the filtering aspect is introduced. Expression of an FFM as a state-space model linearised around the set point p^* has a key advantage that uncertainty in the system state and measured performance can be modelled, addressing the reality of model error and uncertainty (Kolossa *et al.*, 2017). One approach to this, as first demonstrated within the FFM literature by Kolossa *et al.* (2017), is to combine the state-space model with a Kalman filter to improve estimates of fitness and fatigue in the presence of new observed data. At time step n Kalman filter aims to generate an *a posteriori* state estimate $\hat{\mathbf{x}}_n$ based on an *a priori* estimate $\mathbf{z}_n$ from the system dynamics and the observed performance value p_n. The extent to which the *a posteriori* estimate is updated depends on the relative extent of the uncertainty in the state to the uncertainty in the measurement (the ratio of $\mathbf{Q}$ to ξ^2). When the uncertainty in the state is large relative to the uncertainty in the measurement, the "Kalman gain" will be high and the filter will place more weight on the incoming performance data resulting in relatively large corrections in the *a posteriori* estimate. In contrast, when measurement uncertainty is large relative to uncertainty in the state, little weight will be placed on the incoming data and there will be minimal correction to the initial *a priori* estimate. At each time step *n* the fitness and fatigue estimates are updated for all observations up to and including p_n, via the following recursive flow:

Kalman-filtering procedural flow:

1) An *a priori* state estimate $\mathbf{z}_n$ predicts the expected performance $\hat{p}_n$ through the matrix $\mathbf{C}$
2) Expected performance $\hat{p}_n$ is then compared with measured performance p_n if available, and the residual error (ϵ) calculated or otherwise set to zero if no measurement is available
3) Kalman gain $\mathbf{K}_n$ is calculated depending on state and observation noise variances:

$$\mathbf{K}_n = \mathbf{M}_n\mathbf{C}^{\mathrm{T}}\left(\xi^2 + \mathbf{C}\mathbf{M}_n\mathbf{C}^{\mathrm{T}}\right)^{-1} \tag{6.47}$$

Where $\mathbf{M}_n$ is the covariance matrix of $\hat{\mathbf{x}}_n$, iteratively updated by:

$$\mathbf{M}_{n+1} = \mathbf{Q} + \mathbf{A}\mathbf{M}_n\mathbf{A}^{\mathrm{T}} - \mathbf{A}\mathbf{M}_n\mathbf{C}^{\mathrm{T}}\left(\xi^2 + \mathbf{C}\mathbf{M}_n\mathbf{C}^{\mathrm{T}}\right)^{-1}\mathbf{C}\mathbf{M}_n\mathbf{A}^{\mathrm{T}} \quad (6.48)$$

4) The filter uses the error feedback to correct the state estimates, affecting the prediction of performance at the next time step. The *a posteriori* state estimate is updated by the feedback:

$$\hat{\mathbf{x}}_n = \mathbf{z}_n + \mathbf{K}_n(p_n - \mathbf{C}\mathbf{z}_n) \quad (6.49)$$

5) The training impulse ω_n is multiplied by the time-varying coefficient matrix $\mathbf{B}$ and added to the state variables such that at the start of the next time-step a predicted *a priori* state estimate $\mathbf{z}_n$ is formed by the deterministic mechanism of the system, where

$$\mathbf{z}_k = \mathbf{A}\hat{\mathbf{x}}_{n-1} + \mathbf{B}_{n-1}\omega_{n-1} \quad (6.50)$$

After initialisation, the updating of matrix $\mathbf{M}_n$ governs how the Kalman gain evolves over time and the strength of the filtering effect of the model. In a non-stationary case (such as the FFM) initialisation of $\mathbf{M}_0$ falls into three categories: 1) "known" where values are set; 2) "approximate diffuse" where $\mathbf{M}_0 = \kappa\boldsymbol{I}$ for large κ; or 3) "exact diffuse" which relies on limits as variances approach infinity (Fulton, 2017).

When using the Kalman filter, in the context of general FFMs, the model parameters must be estimated from training and performance data. This is achieved through algorithmically minimising some loss criterion, for example, the residual sum of squares between modelled and measured performance data, which, in the case of gaussian errors, coincides with the likelihood function (Mannakee *et al.*, 2016). With even the standard FFM, optimisation is for all practical purposes analytically intractable and numerical procedures are used that require starting values for parameters $\{p^*, k_g, k_h, \tau_g, \tau_h\}$. While the available algorithms differ (as discussed in subsection 6.2.4), they all are iterative in nature, stopping when the loss function falls below some prespecified threshold. When using the Kalman filter, the likelihood is available as a by-product of filtering operations (Fulton, 2017), and thus the Kalman filter can be fit with the same optimisation routines as the standard FFM with additional starting values for the extra parameters and initial conditions. Given that an entire run of the filtering algorithm is required to obtain the likelihood of the sample, a "double loop" results when paired with a numerical optimisation procedure, and time to convergence may be slow for long series, in particular because R is by definition single threaded. Additionally, given the additional parameters that must be estimated in the Kalman filter model given the same setup as the standard FFM, some "sloppiness" in parameter estimation is likely (Transtrum *et al.*, 2015). The full $\boldsymbol{Q}$ matrix may also be challenging to recover. This procedure is followed next, beginning with the development of a Kalman filtering function based on the steps described previously, that takes as input: 1) a set of training load and measured

performance data; and 2) a state-space model instantiated under a set of parameters via the function `state_space_FFM` developed in listing 6.37. This filtering function forms the 'inner loop' in the optimisation procedure, with the outer loop modifying the state-space model supplied under iteratively changing parameters.

Listing 6.40 Kalman filtering function

```
kalman_filter <- function(ss_model, dat){

  # Extract properties of the dataset
  T <- nrow(dat)
  loads <- dat$load
  measured <- dat$performance

  # Set up object structures
  loglike <- numeric(T)
  X <- matrix(rep(NA, 2 * T), ncol = 2)  # a posteriori state estimate
  M <- matrix(rep(NA, 4 * T), ncol = 4)  # Vectorised state vcovs
  Z <- matrix(rep(NA, 2 * T), ncol = 2)  # a priori state estimates

  # Kalman updating equations
  for (n in 1:T){
    if (n == 1){   # Initialisation
      z_n <- ss_model$A %*% ss_model$x_0 + ss_model$B * mean(loads)
      P_n <- ss_model$Q + ss_model$A %*% ss_model$M_0 %*% t(ss_model$A)
    } else{
      z_n <- ss_model$A %*% X[n-1, ] + ss_model$B * loads[n-1]
      P_n <- ss_model$Q + ss_model$A %*% matrix(M[n-1,], ncol = 2) %*% t(ss_model$A)
    }

    # Likelihood of performance measurement
    S_n <- ss_model$xi^2 + ss_model$C %*% P_n %*% t(ss_model$C)
    e_n <- measured[n] - (ss_model$p_star + ss_model$C %*% z_n)
    loglike[n] <- dnorm(e_n, mean = 0, sd = sqrt(S_n), log = TRUE)

    # A posterori mean and variance
    K_n <- P_n %*% t(ss_model$C) %*% (1 / S_n) # Kalman gain
    X[n, ] <- z_n + K_n %*% e_n
    M[n, ] <- as.vector((diag(2) - K_n %*% ss_model$C) %*% P_n)
    Z[n, ] <- z_n

  } # End for loop
  p_hat <- ss_model$p_star + X %*% t(ss_model$C) # Filtered predictions of performance
  output <- list(p_hat = p_hat, g_hat = X[, 1], h_hat = X[, 2], M = M, Z = Z,
                 loglike = loglike)
  return(output)
}
```

Following an example of the operation of this function in listing 6.40, Kalman filtering is demonstrated when all the parameters are set to the 'truth' as per listing 6.38, using the simulated performance data generated in listing 6.39 as the input for the argument `dat`. Even in this case when the truth parameters are known, the state is still latent (due to the noise terms incorporated in listing 6.38) and needs to be estimated.

Listing 6.41	Kalman filtering demonstration for the simple case of the simulated data under true parameters in listing 6.38

```
filtered_model <- kalman_filter(ss_model, dat = simulated_ss_model)

# Plot the results
plot(filtered_model$p_hat, type = "l", lty = 2, col = "blue", ylab = "Performance [a.u]",
     xlab = "Day", lwd = 1.5)
points(simulated_ss_model$performance, lty = 1, pch = 16, col = "red")
legend("topleft", c("Kalman filtered FFM under true parameters", "Simulated data"),
       lty = c(2, NA), pch = c(NA, 16), col = c("blue", "red"), lwd = c(1.5, NA), cex = 0.75)
```

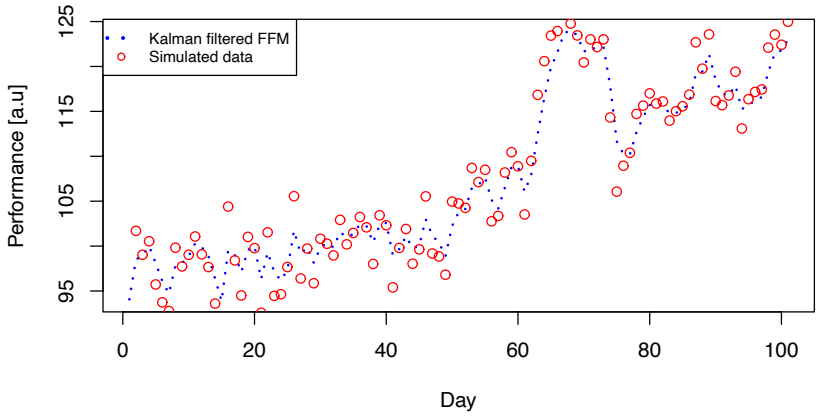


Figure 6.24: Kalman-filtered model under true parameters & simulated data developed in listing 6.39

Examining the output of the Kalman filter for the example in listing 6.41, the likelihood of a performance trajectory for experimental data is easily computed within the filtering flow, with maximum likelihood therefore representative of the most convenience choice. However, the term 'easy' does not necessarily apply to the optimisation problem. Consider the case where the simulated data generated in listing 6.39 was in fact real-world data, and we didn't know the true parameters associated with it, when fitting the state-space FFM with Kalman filter we may need to first initialise the state-space model under a set of initial guesses. This qualifies as a more than slightly tricky problem compared to selecting starting parameters for the classic ad-hoc approach to fitting the standard model without noise terms or initial components. Given the additional parameters, initial conditions, and noise terms to consider, and issues identified in chapter 5, pathological curvature of the objective function and existence of local optima are only likely to be exacerbated in this fitting problem. Nevertheless, if attempting to fit the Kalman filter within an MLE perspective via a first or second-order search algorithm, some possible suggestions are given in Swinton *et al.* (2021) as to how starting point values may be derived. These included:

- $\mathbf{M_0}$ (Unconditional variance of the initial state) – ½ the magnitudes of the initial state vector for the standard deviations, carrying the interpretation that 2 standard deviations is 100% of the initial state guess. Setting zero as the covariance.

- The initial state update of the Kalman filter depends on an initial state value (fitness and fatigue) but also on a training load for $t = 0$. Rather than setting $\omega_0 = 0$, or creating yet another parameter to estimate, the reader may choose to use the average training load in the data set.
- For σ_g, σ_h, a rough state error prediction can be obtained by computing $g(t) - e^{-\frac{1}{\tau}} g(t-1)$ and taking the standard deviation over the time-series (doing the same for fatigue).
- The initial state correlation $\rho_{(g,h)_0}$ set to zero

Considering the attention given to optimisation earlier in section 6.2.4 and the issues described above, it makes some sense to jump straight to global optimisation methods (e.g., evolutionary strategies) when developing a function to fit the state-space model with Kalman filter via MLE. These algorithms typically do not require the user to specify initial values, commonly using a starting population seeded randomly over the bounds.

Below (listing 6.42), an R function for data fitting is developed that applies a genetic algorithm to fit the space-state model with Kalman filtering to available data. The `fit_filtered_model` function only requires the user to supply modelling data and a set of box-constraints for the parameters (although we fix $p_{(g,h)_0} = 0$, leaving 13 free parameters in the problem). With this many parameters, there is no evidence to suggest that an algorithm such as L-BFGS-B would perform well on its own, considering the discussion of local optima and evidence thereof presented in chapter 5. However, the genetic algorithm can be combined with the L-BFGS-B algorithm such that a local search is still performed at stochastic intervals throughout the iterative process. This is a preferable approach, given the ability to satisfy optimality conditions with local optimisers, and the overall reach of global optimisers. The user can also tune whether the local search is initiated from members of the population with good fitness, or via some kind of quasi-uniform assignment. The function in listing 6.42 can be thought of as the 'outer loop' described earlier in the use of Kalman-filtering for fitness-fatigue modelling. As it is presented below, the genetic algorithm is not tuned in any particular way beyond the defaults provided in the package *GA* (Scrucca, 2013).

Listing 6.42 Defining a function to fit a state-space model with Kalman filter to data

```
fit_filtered_model <- function(dat, box){

  extract_pars <- function(par){
    pars <- list(p_star = par[1],
                 k_g = par[2],
                 k_h = par[3],
                 tau_g = par[4],
                 tau_h = par[5],
                 g_0 = par[6],
                 h_0 = par[7],
                 xi = par[8],
                 sd_g0 = par[9],
                 sd_h0 = par[10],
                 rho_gh0 = 0, # Fix rho_gh0 to zero
                 sigma_g = par[11],
                 sigma_h = par[12],
                 rho_gh = par[13])
    return(pars)
  }

  log_likelihood <- function(par){
    pars <- extract_pars(par)
    temp_model <- state_space_FFM(pars)
    filtered_model <- kalman_filter(ss_model = temp_model, dat = dat)
    # Note as GA is a maximiser by default, we just need the log likelihood
    return(sum(filtered_model$loglike))
  }

  maximise_likelihood <- GA::ga("real-valued",
                                fitness = log_likelihood,
                                lower = box$lower,
                                upper = box$upper,
                                maxiter = 5000,
                                monitor = TRUE,
                                optim = TRUE, # Local search via L-BFGS-B (random interval)
                                optimArgs = list(method = "L-BFGS-B",
                                                 lower = box$lower,
                                                 upper = box$upper,
                                                 poptim = 0.2,      # Probability of search
                                                 pressel = 0.3),
                                popSize = 100,
                                parallel = TRUE
                                )

  pars <- extract_pars(maximise_likelihood@solution)
  plot(maximise_likelihood)

  fitted_model <- state_space_FFM(pars)
  return(list("fitted_model" = fitted_model, "optimisation" = maximise_likelihood))
}
```

Demonstrating this function in practice, listing 6.43 returns to the simulated data from listing 6.8, imagining that it reflects some form of 'real-world' measured data for which we do not know the underlying true parameters (which requires recovery via optimisation). In listing 6.43, generous but reasonable bounds are supplied for the parameters, discussion of which is also given in the appendices of Swinton *et al.* (2021). Supplying the simulated data and these bounds to the function `fit_filtered_model` yields the parameters from the fitted state-space model under the Kalman filter, in the form of the instantiated state-space model. To retrieve the associated filtered performance values the result must be passed back again through `kalman_filter` (listing 6.40). It was decided that it was more illuminating to do this outside of the fitting function, but could easily be included within such

that the filtered performance values were returned rather than the fitted state-space model. A high number of iterations (2000) were used in this example, which will be excessive for the reader when playing around with these models in their own environment. However, this was intended to give the genetic algorithm a good shot (in this example) at recovering close to the true parameters, given the high dimensionality of the fitting problem. Toward this end, and for the purposes of the example (listing 6.42) the genetic algorithm used was tuned further as follows:

- The probability of the algorithm performing a local search at each iteration of was increased to 0.2 (default 0.1), increasing the overall number of local searches performed.
- The local search is started from a random solution selected with probability proportional to its 'fitness' (objective value). The pressure selection argument of the local search was set to 0.3 (default 0.5) such that a lower value assigns quasi-uniform probabilities to any solution. The argument for this was that local searches should try to span the search space rather than cluster (take place) at potential local minima already found. Particularly given the ill-conditioning of the problem space.
- The population size was increased from the default 50, to 100.
- A `parallel = TRUE` argument was supplied, parallelising the fitting process via SNOW type functionality (windows) or multicore (on OSX, Unix, Linux).

This change in the implementation is visible in **github.com/bsh2/thesis/c6/kalman_filter.R**. The choice of these tuning parameters tended toward options that inherently decrease the efficiency of the algorithm, but possibly improve its chances of recovering the true parameters for the FFM fitting problem. What is apparent from the results of this process (figures 6.25 and 6.26) is that although the approach reflects a reasonable initial effort, the optimisation problem is clearly rather challenging with this high a number of free parameters (model parameters, initial conditions, noise terms), and an ad-hoc fitting approach is unlikely to be sufficient. However, genetic algorithms provide a lot of power when tuned correctly with regard to genetic parameters and strategy selected (Scrucca, 2013), and if the parameter space is appropriate constrained there may be further wriggle room for improvement in this area.

Listing 6.43	Kalman filtering demonstration for the simple case of the simulated data under true parameters in listing 6.38

```
# c(p*, kg, kh, Tau_g, Tau_h, g_0, h_0, xi, sd_g0, sd_h0, sigma_g, sigma_h, rho_gh)
box_constraints <- data.frame("lower" = c(50, 0.1, 0.1, 1, 1, 1, 0.5, 0.5, 0.01, 0.01,
                              1, 1, -0.999), "upper" = c(150, 5, 5, 50, 50, 20,
                                                         20, 10, 5, 5, 5, 5, 0.999))

fitted_kalman_model <- fit_filtered_model(dat = simulated_ss_model, box = box_constraints)
fitted_kalman_predictions <- kalman_filter(fitted_kalman_model$fitted_model, simulated_ss_model)

```

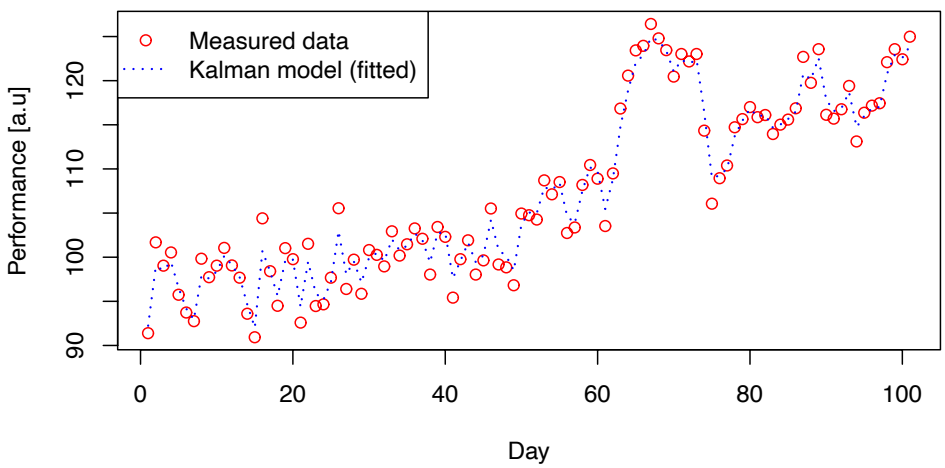


Figure 6.25: Fitted Kalman model

Fitted Kalman-filtered model	**Truth model**
$A [,1] [,2] [1,] 0.9801987 0.0000000 [2,] 0.0000000 0.8417934 $B [,1] [1,] 0.9801987 [2,] 0.8417934 $C [,1] [,2] [1,] 1.079778 -1.970515 $Q [,1] [,2] [1,] 1.000000 1.970457 [2,] 1.970457 3.890477 $xi [1] 2.088804 $x_0 [1] 15.255718 1.407712 $M_0 [,1] [,2] [1,] 0.0001 0.0000 [2,] 0.0000 0.0001 $p_star [1] 80.54032	$A [,1] [,2] [1,] 0.9622687 0.0000000 [2,] 0.0000000 0.8187308 $B [,1] [1,] 0.9622687 [2,] 0.8187308 $C [,1] [,2] [1,] 0.85 -1.2 $Q [,1] [,2] [1,] 4.00 0.72 [2,] 0.72 1.44 $xi [1] 2.5 $x_0 [1] 10 6 $M_0 [,1] [,2] [1,] 4 0 [2,] 0 1 $p_star [1] 95

Figure 6.26: R output: A comparison between the truth model underpinning the simulated data, and the fitted Kalman-filtered model with parameters recovered by the optimisation procedure.

In figure 6.26, the τ parameters can be recovered from the $\mathbf{A}$ matrix by $\tau_g = -1/\log(\mathbf{A}[1,1])$ and $\tau_h = -1/\log(\mathbf{A}[2,2])$, the k_g and k_h parameters recovered via the $\mathbf{C}$ matrix, $\sigma_g, \sigma_h, \rho_{g,h}$ from the $\mathbf{Q}$ matrix, as follows:

```
kg <- C[1,1]
kh <- -1 * C[1,2]
Tg <- -1 / log(A[1,1])
Th <- -1 / log(A[2,2])
sigma_g <- sqrt(Q[1,1])
sigma_h <- sqrt(Q[2,2])
rho_gh <- sqrt(Q[1,2] / (sigma_g * sigma_h))
```

6.5 Summary

The purpose of this chapter has been to provide extensive resources and demonstrate how to develop further resources for researchers interested in fitness-fatigue modelling and the implementation of FFMs. The chapter has drawn together theory and practical insight gathered over the course of this project in areas such as parameter estimation, cross-validation, and parallel computing. Models examined included the standard, fitness-delay, VDR, the nonlinear variant of the original ODE system, as well as the state-space Kalman-model reformulation. Fitting FFMs has also been illustrated under both nonlinear least-squares and maximum-likelihood, with accompanying theory provided. The structure of the chapter has required the reader to engage with the underlying concepts in the code tools, in contrast to provision of polished software or packaged code. In this way, the reader gains more flexibility and understanding in how the tools can be applied in future research, serving both an educational and practical purpose. Although this thesis has highlighted that FFMs like many models are imperfect and can suffer from a range of limitations, as a collective, they still reflect one of the most advanced class of models available in sport science for performance modelling. Therefore, at present, this chapter reflects the most definitive self-contained practical resource in this area.

Chapter 7: Summary and conclusions

The thesis has comprised a range of individual works including extensive literature reviews, the presentation of mathematical formulae with derivations, findings from multiple large scale *in silico* experiments, and a detailed treatise on how to implement FFMs in the programming language R. Given the range of the work, this chapter provides a succinct summary to the work, highlighting the main experimental findings, impact of the research on current understanding, limitations of the work, and directions for future study.

7.1 Experimental findings

Chapter 4: *Suitability of a quasi-Newton algorithm for estimating FFMs: Sensitivity, troublesome local optima, and implications for future research*

This study provided novel insight into the effects of two key factors of experimental data FFM accuracy, including measurement error and testing frequency. The study generated lower-bound estimates that demonstrated that practitioners should focus on collecting data comprising performance tests that generate measurements with high reliability. It was also recommended that researchers consider the use of multiple trials (Swinton *et al.*, 2018), and filtering techniques to reduce noise due to biological variability and/or instrumentation error. In addition, the study demonstrated deleterious effects of measurement error at low testing frequencies indicating that researchers should attempt to maximise testing frequency via maximal-effort tests that do not encompass large fatigue effects.

Chapter 5: *The effects of measurement error and testing frequency on the standard fitness-fatigue model applied to synthetic resistance training data*

This study highlighted the existence of many unique local optima in the search space (both minima and saddle points) reflected in sensitivity to initial values when fitting an FFM to data via a first-order algorithm (with second-order approximation) and identified that this problem can be exacerbated with increased degrees of freedom in the model. The local optima found in this study also typically demonstrated good or sometimes indistinguishable model fit to the true solutions but simultaneously differed substantively in at least one parameter to the true values, indicating that local minima are not necessarily clustered around or at the global minima as is often the case with models such as neural networks. Collectively, the findings of this study add weight to the hypothesis that there exists substantial doubt in reported estimates across prior literature where local optimisers have been used under ‘one-shot’ runs, or where studies have lacked the relevant detail to indicate that these issues have been considered or measures taken to address them.

7.2 Impact of the work on current understanding in research and practice

Fitness-fatigue modelling is a relatively small and niche area of research in sport science that has been surrounded by several historical issues that are reflected in the typical approaches of prior experimentation and limitations in the accessibility of the literature body. Important initial steps have been taken in recent research to address some of the model limitations and processes used to fit FFMs including incorporating variants of cross-validation (Chalencon, 2015; Ludwig, Schaefer and Asteroth, 2016; Kolossa *et al.*, 2017; Turner *et al.*, 2017; Williams *et al.*, 2018; Stephens Hemingway *et al.*, 2019; Imbach *et al.*, 2020) and trialling alternative parameter estimation approaches beyond derivative-based methods (Turner *et al.*, 2017; Méline *et al.*, 2018; Philippe *et al.*, 2018; Connor and O'Neill, 2020). The present thesis expands greatly on this work with impact including original contributions to knowledge through novel *in silico* experiments (chapters 4, 5). Additionally, the thesis creates extensive impact through the synthesis of existing information on FFMs (chapter 2) and the development of code resources (chapter 6) providing the most extensive resource to date on the topic. The set of researchers in sport science with the requisite skills and knowledge to investigate performance modelling has been fundamentally limited by the underlying literature body, and it is likely this is one of the main reasons why progress in this area has lagged behind other areas relative to its historical timeframe. Therefore, it is expected that the main impact of this thesis will be the increased accessibility of the area through the comprehensive overview and analysis provided, elucidating key concepts and practical resources to improve understanding and increase the available infrastructure for future research.

7.3 Limitations of the work

The primary limitation of the studies carried out in both chapters 4 and 5 (the *in-silico* studies) was that these were case-based (low *n* studies externally), despite being high *n* internally (i.e., studying several conditions via many iterations). That is, they examined a small number of modelling inputs (training loads and simulated performance profiles), and subsequently this limits how results may be generalised to all inputs and performance profiles. Importantly, these cases were developed to be representative and based on literature, and if repeated for a wider set of cases it is likely these studies would resolve to similar findings. However, there would still be a small amount of variability expected and a small proportion of cases that resolve to surprising outcomes (e.g., that are not affected by low measurement frequency or less affected by high errors, or where the search space incurs fewer local minima and the FFM is easier to fit). Collectively, these studies have highlighted that there do exist cases where measurement practices and choice of parameter estimation are important. Therefore, researchers should still operate on the notion that reducing measurement error, increasing measurement frequency, and appropriate parameter estimation approaches are likely to both improve

outcomes, robustness, and therefore credibility of future work. A further limitation of chapter 4 was that it only provides lower-bound estimates, whereby there are a range of other complex additional factors that would be expected to influence an FFMs prediction of an athlete's response to training, including model misspecification.

A secondary limitation of the thesis is that no suitable real-world data were collected or obtained to test key FFMs once promising models, key experimental factors, and alternative parameter estimation approaches had been identified from chapters 2,4, and 5. This was due in part to the tail end of this project crossing into the covid-19 pandemic and also the challenges in obtaining sufficient data. Whilst monitoring of athletes has become commonplace, collection of high-quality data including complete and detailed training logs over extended periods of time with frequent performance measurements remains a challenge. In addition, to avoid the limitations of previous FFM research, data are required from large numbers of athletes across multiple sports comprising different combinations of training inputs and outcomes to obtain generalisable assessments of FFMs across a range of contexts. Even if this data were made available, resources were not widely available prior to this thesis to implement models. Therefore, for maximising impact the code tools developed in chapter 6 still arguably represent the most appropriate step beyond chapters 4 and 5 in the research project. These permit researchers to investigate FFMs via both prospective and retrospective studies, beyond what could be achieved from a further experimental chapter in this thesis. In addition, these resources combined with the reviews may improve the accessibility of the research area to practitioners, and in ideal case spur new interest from industry for research collaboration on a wider scale.

Finally, the main limitation of chapter 6 is that the use of the resources presented require the researcher to possess a reasonable baseline knowledge of the R programming language. This is contrast to an R package which is straightforward to run, or dashboard-type software that provide the user with a high level of abstraction away from the code. However, the tools for fitting and evaluating FFMs are contained in user-defined functions that do not require much more beyond using an R package. Perhaps more importantly, the code offers the field a set of transparent resources developed from first principles, which arguably reflect the most appropriate initial step in developing robust infrastructure for performance modelling in sport science. In other words, before masking the functionality to the user, the functionality should first be presented to the field and refined or improved as required. Going beyond this chapter, the next step in this software development process should be packaging the R functions with further input-checking, or as mentioned dashboard-type software, to further improve the efficiency of future work.

7.4 Directions of future study

As the world begins to re-emerge from the restrictions of the covid-19 pandemic in many countries and as athletes return to formalised training and researchers to university campuses, it is hoped that the required data collection environments for FFM research may soon become feasible again (both laboratory and field). In the interim, researchers may look toward other means of obtaining suitable data for testing FFMs. For example: 1) the collaborative use of particular datasets from experimental studies already in the literature, repurposed to more advanced models and methods; 2) from practitioners working in industry with access to suitable training and performance data from elite level groups, as demonstrated in Rozendaal (2017); 3) from remote recruitment and participation via online platforms used by endurance athletes to log training and regularly record physical performance efforts (e.g., TrainingPeaks ®, Strava ®). Ultimately, contemporary FFMs require further evaluation under real-world data and using robust methods of model testing to ascertain validity and utility for informing training program design and tapering periods, and it is argued that study of FFMs should remain an active area of research in sport science. In the area of performance modelling there is also considerable scope for ingenuity and interdisciplinary collaboration. Researchers must, however, ensure that they adopt appropriate measurement protocols for their studies and take care in the estimation of model parameters. Toward this end, adequately solving the model fitting problem given the concerns outlined in chapter 5 with using only derivative-based methods remains one of the key areas in the evaluation of FFM utility that requires further input from researchers. Until addressed, possibly with the help of experts in areas of numerical methods and optimisation, and until researchers are confident that methods identified from study reflect a convincing approach for fitting FFMs, the results of future work may be limited by uncertainty in the adequacy of solutions obtained. In this area, promising approaches include the study of evolutionary algorithms and in particular hybrid approaches which combine the global optimisation properties of evolutionary algorithms with the optimality conditions of derivative-based methods (Méline *et al.*, 2018; Philippe *et al.*, 2018; Connor and O'Neill, 2020; Stephens Hemingway *et al.*, 2021). Future work may also wish to consider the use of informative priors under a Bayesian optimisation approach, and develop appropriate approaches to elicit priors based on prospective methods in available literature (Garthwaite, Kadane and O'Hagan, 2005; Johnson *et al.*, 2010). In addition, research may consider matching athletes and fitting multi-level models, sharing information to obtain parameter values representative of athletes with similar adaptive responses.

In the area of infrastructure development for research, there is room to extend the tools developed in chapter 6 to incorporate input checking and additional FFMs and methods. The resources developed could also be contained within an R package or used as the backend for a web-based dashboard that improves the efficiency of the research process, but that has the downside of less transparency due to

lower direct interaction with and control of the underlying code. However, with respect to the code tools developed and associated repository (fitnessfatigue.com), care should be taken to ensure that any new functionality is still well documented, and this information is made clearly available.

Lastly, future research will benefit most from increased transparency and reproducibility around data, assumptions, and methods of implementation (particularly code used to fit and evaluate models). It is therefore recommended where possible that future studies utilise the basic tenants of reproducible research including availability of documented code and inclusion of raw data and/or analysis (Peng, 2011). Furthermore, of upmost importance is consideration of the key audience when presenting and communicating the work. Researchers should aim to publish in sport science journals wherever possible, address the work and narrative to the general sport scientist, and prioritise communication of the impact of the study and practical implications of any results on understanding model validity and in reference to training program design (see the research model developed in chapter 3). Where necessary, researchers should also endeavour to create educational resources around their work and include these in supplementary files to elucidate complicated or novel procedural aspects, and greater care must be taken with regard to the pedagogical style used across the field.

Bibliography

1. Agostinho, M. F. *et al.* (2015) 'Perceived training intensity and performance changes quantification in judo', *Journal of Strength and Conditioning Research*, 29(6), pp. 1570–1577. doi: 10.1519/JSC.0000000000000777.
2. Al-Otaibi, N. M. (2017) *Statistical modelling of training and performance using power output and heart rate data collected in the field*. University of Salford.
3. Appleby, B., Newton, R. U. and Cormie, P. (2012) 'Changes in strength over a 2-year period in professional Rugby Union players', *Journal of Strength and Conditioning Research*. doi: 10.1519/JSC.0b013e31823f8b86.
4. Arandjelovic, O. (2013) 'Computer simulation based parameter selection for resistance exercise', *arXiv preprint arXiv:1306.4724*.
5. Arandjelović, O. (2017) 'Computer-aided parameter selection for resistance exercise using machine vision-based capability profile estimation', *Augmented Human Research*, 2(1), pp. 1–19.
6. Baca, A. and Perl, J. (2018) *Modelling and simulation in sport and exercise*. Routledge.
7. Bäck, T. and Schwefel, H.-P. (1993) 'An overview of evolutionary algorithms for parameter optimization', *Evolutionary computation*, 1(1), pp. 1–23.
8. Baker, D. (1998) 'Applying the in-season periodization of strength and power training to football', *Strength and Conditioning Journal*. doi: 10.1519/1073-6840(1998)020<0018:atispo>2.3.co;2.
9. Baker, D. (2007) 'Cycle-length variants in periodized strength/power training', *Strength and Conditioning Journal*. doi: 10.1519/00126548-200708000-00001.
10. Banister, E. W. *et al.* (1975) 'A Systems Model of Training for Athletic Performance', *Australian Journal of Sports Medicine*, 7(3), pp. 57–61.
11. Banister, E. W. *et al.* (1986) 'Modelling the training response in athletes', in *The 1974 Olympic Scientific Congress Proceedings. Sport and Elite Performers*, pp. 7–23.
12. Banister, E. W. and Calvert, T. W. (1980) 'Planning for future performance: implications for long term training', *Canadian Journal of Applied Sport Science*, 5, pp. 170–176.
13. Banister, E. W., Carter, J. B. and Zarkadas, P. C. (1999) 'Training theory and taper: validation in triathlon athletes', *European Journal of Applied Physiology and Occupational Physiology*, 79(2), pp. 182–191. doi: 10.1007/s004210050493.
14. Banister, E. W. and Hamilton, C. L. (1985) 'Variations in iron status with fatigue modelled from training in female distance runners', *European Journal of Applied Physiology and Occupational Physiology*, 54(1), pp. 16–23. doi: 10.1007/BF00426292.
15. Banister, E. W., Morton, R. H. and Fitz-Clarke, J. R. (1992) 'Dose/Response Effects of Exercise Modeled from Training: Physical and Biochemical Measures', *The Annals of physiological anthropology*, 11(3), pp. 345–356.
16. Bartling, S. and Friesike, S. (2014) *Opening science: The evolving guide on how the internet is*

changing research, collaboration and scholarly publishing. Springer Nature.

17. Bates, D. M. and Watts, D. G. (1988) *Nonlinear Regression Analysis and Its Applications*. New York: Wiley. doi: 10.2307/1268866.
18. Bendtsen, C. and Bendtsen, M. C. (2011) 'Package "pso"'. Version.
19. Beneke, R. and Boning, D. (2008) 'The limits of human performance', *Essays in biochemistry*, 44(1), p. 11.
20. Binswanger, M. (2014) 'Excellence by nonsense: The competition for publications in modern science', in *Opening Science*. Springer, Cham, pp. 49–72.
21. Bishop, D. (2008) 'An applied research model for the sport sciences', *Sports Medicine*, 38(3), pp. 253–263.
22. Bompa, T. O. and Buzzichelli, C. (2018) *Periodization-: theory and methodology of training*. Human kinetics.
23. Borresen, J. and Lambert, M. I. (2009) 'The Quantification of Training Load, the Training Response and the Effect on Performance', *Sports Medicine*, 39(9), pp. 779–95. doi: 10.2165/11317780-000000000-00000.
24. Bourdon, P. C. *et al.* (2017) 'Monitoring athlete training loads: Consensus statement', *International Journal of Sports Physiology and Performance*, 12(S2), pp. 161–170. doi: 10.1123/IJSPP.2017-0208.
25. Breiman, L. (1996) 'Bagging predictors', *Machine Learning*. doi: 10.1007/bf00058655.
26. le Bris, S. *et al.* (2004) 'Applying a systems model of training to a patient with coronary artery disease', *Medicine and Science in Sports and Exercise*, 36(6), pp. 942–948. doi: 10.1249/01.MSS.0000128247.82321.32.
27. le Bris, S. *et al.* (2006a) 'A systems model of training for patients in phase 2 cardiac rehabilitation', *International Journal of Cardiology*, 109(2), pp. 257–263. doi: 10.1016/j.ijcard.2005.06.029.
28. le Bris, S. *et al.* (2006b) 'High versus low training frequency in cardiac rehabilitation using a systems model of training', *European Journal of Applied Physiology*, 96(3), pp. 217–224. doi: 10.1007/s00421-005-0043-2.
29. Burns, P. (2011) *The R inferno*. Lulu. com.
30. Busso, T. *et al.* (1990) 'A systems model of training responses and its relationship to hormonal responses in elite weight-lifters', *European Journal of Applied Physiology and Occupational Physiology*, 61(1), pp. 48–54. doi: 10.1001/jama.1937.02780230056030.
31. Busso, T. *et al.* (1992) 'Hormonal adaptations and modelled responses in elite weightlifters during 6 weeks of training', *European Journal of Applied Physiology and Occupational Physiology*, 64(4), pp. 381–386. doi: 10.1007/BF00636228.
32. Busso, T. *et al.* (1997) 'Modeling of adaptations to physical training by using a recursive least squares algorithm', *Journal of Applied Physiology: Modeling in Physiology*, 82(7), pp. 1685–1693.

33. Busso, T. *et al.* (2002) ‘Effects of training frequency on the dynamics of performance response to a single training bout’, *Journal of Applied Physiology*, 92(2), pp. 572–580. doi: 10.1152/japplphysiol.00429.2001.
34. Busso, T. (2003) ‘Variable dose-response relationship between exercise training and performance’, *Medicine and Science in Sports and Exercise*, 35(7), pp. 1188–1195. doi: 10.1249/01.MSS.0000074465.13621.37.
35. Busso, T. (2017) ‘From an indirect response pharmacodynamic model towards a secondary signal model of dose-response relationship between exercise training and physical performance’, *Scientific Reports*, 7(November 2016), pp. 1–11. doi: 10.1038/srep40422.
36. Busso, T., Candau, R. and Lacour, J. R. (1994) ‘Fatigue and fitness modelled from the effects of training on performance’, *European Journal of Applied Physiology and Occupational Physiology*, 69(1), pp. 50–54. doi: 10.1007/BF00867927.
37. Busso, T., Carasso, C. and Lacour, J. R. (1991) ‘Adequacy of a systems structure in the modeling of training effects on performance’, *Journal of Applied Physiology*, 71(5), pp. 2044–2049.
38. Busso, T. and Thomas, L. (2006) ‘Using Mathematical Modeling in Training Planning’, *International journal of sports physiology and performance: Invited Commentary*, 1(4), pp. 400–405. doi: 10.1123/ijspp.1.4.400.
39. Byrd, R. H. *et al.* (1995a) ‘A Limited Memory Algorithm for Bound Constrained Optimization’, *SIAM Journal on Scientific Computing*, 16(5), pp. 1190–1208. doi: 10.1137/0916069.
40. Byrd, R. H. *et al.* (1995b) ‘A Limited Memory Algorithm for Bound Constrained Optimization’, *SIAM Journal on Scientific Computing*. doi: 10.1137/0916069.
41. Calvert, T. W. *et al.* (1976) ‘A Systems Model of the Effects of Training on Physical Perfoffnance’, *IEEE Transactions on Systems, Man, and Cybernetics*, 6(2), pp. 94–102. Available at: https://www.math.fsu.edu/~dgalvis/journalclub/papers/11_28_2016.pdf.
42. Candau, R., Busso, T. and Lacour, J. R. (1992) ‘Effects of training on iron status in cross-country skiers’, *European Journal of Applied Physiology and Occupational Physiology*, 64(1), pp. 497–502.
43. Carlock, J. M. *et al.* (2004) ‘The relationship between vertical jump power estimates and weightlifting ability: A field-test approach’, *Journal of Strength and Conditioning Research*. doi: 10.1519/R-13213.1.
44. Chalencon, S. *et al.* (2012) ‘A Model for the Training Effects in Swimming Demonstrates a Strong Relationship between Parasympathetic Activity, Performance and Index of Fatigue’, *PLoS ONE*, 7(12), pp. 1–10. doi: 10.1371/journal.pone.0052636.
45. Chalencon, S. *et al.* (2015) ‘Modeling of performance and ANS activity for predicting future responses to training’, *European Journal of Applied Physiology*, 115(3), pp. 589–596. doi: 10.1007/s00421-014-3035-2.
46. Chalencon, S. (2015) *Prediction of performance in swimming by Autonomic Nervous System*

activity assessment : mathematical modeling. University of Lyon.

47. Chiu, L. Z. F. and Barnes, J. L. (2003) 'The Fitness-Fatigue Model Revisited: Implications for Planning Short- and Long-Term Training', *Strength and Conditioning Journal*, 25(6), pp. 42–51. doi: 10.1519/1533-4295(2003)025<0042.
48. Cissik, J., Hedrick, A. and Barnes, M. (2008) 'Challenges applying the research on periodization', *Strength & Conditioning Journal*, 30(1), pp. 45–51.
49. Clarke, D. C. and Skiba, P. F. (2013) 'Rationale and resources for teaching the mathematical modeling of athletic training and performance', *American Journal of Physiology - Advances in Physiology Education*, 37(2), pp. 134–152. doi: 10.1152/advan.00078.2011.
50. Cole, S. R. *et al.* (2010) 'Illustrating bias due to conditioning on a collider', *International journal of epidemiology*. 2009/11/19, 39(2), pp. 417–420. doi: 10.1093/ije/dyp334.
51. Connor, M., Fagan, D. and O'Neill, M. (2019) 'Optimising Team Sport Training Plans With Grammatical Evolution', pp. 2474–2481. doi: 10.1109/cec.2019.8790369.
52. Connor, M. and O'Neill, M. (2020) 'Optimizing the Parameters of A Physical Exercise Dose-Response Model: An Algorithmic Comparison', *arXiv preprint arXiv:2012.09287*.
53. Corlett, J. T. (1977) *A System Model of Physical Training and Athletic Performance*, *MSc. Thesis*. Simon Fraser University.
54. Corlett, J. T., Calvert, T. W. and Banister, E. W. (1978) 'Cybernetics of Human Physical Performance', *Current Topics in Cybernetics and Systems*, pp. 180–182.
55. Cormack, S. J. *et al.* (2008) 'Reliability of measures obtained during single and repeated countermovement jumps', *International Journal of Sports Physiology and Performance*. doi: 10.1123/ijspp.3.2.131.
56. Coutts, A. J., Crowcroft, S. and Kempton, T. (2017) 'Developing athlete monitoring systems', *Sport, recovery, and performance: Interdisciplinary insights*.
57. Cronin, J. B., Hing, R. D. and McNair, P. J. (2004) 'Reliability and validity of a linear position transducer for measuring jump performance', *Journal of Strength and Conditioning Research*. doi: 10.1519/1533-4287(2004)18<590:RAVOAL>2.0.CO;2.
58. Cunanan, A. J. *et al.* (2018) 'The general adaptation syndrome: a foundation for the concept of periodization', *Sports Medicine*, 48(4), pp. 787–797.
59. Davidian, M. and Giltinan, D. M. (2003) 'Nonlinear models for repeated measurement data: An overview and update', *Journal of Agricultural, Biological, and Environmental Statistics*. doi: 10.1198/1085711032697.
60. Dennis Jr, J. E. and Schnabel, R. B. (1996) *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM.
61. Domotor, Z. (2011) 'Philosophy of Science, Mathematical Models in', in Meyers, R. A. (ed.) *Mathematics of Complexity and Dynamical Systems*. New York, NY: Springer New York, pp. 1407–1422. doi: 10.1007/978-1-4614-1806-1_89.

62. Ekins, S., Mestres, J. and Testa, B. (2007) ‘In silico pharmacology for drug discovery: methods for virtual ligand screening and profiling’, *British journal of pharmacology*, 152(1), pp. 9–20.
63. Everitt, B. and Skrondal, A. (2002) *The Cambridge dictionary of statistics*. Cambridge University Press Cambridge.
64. Fecher, B. and Friesike, S. (2014) ‘Open science: one term, five schools of thought’, *Opening science*, pp. 17–47.
65. Fitz-Clarke, J. R., Morton, R. H. and Banister, E. W. (1991) ‘Optimizing athletic performance by influence curves’, *Journal of Applied Physiology*, 71(3), pp. 1151–1158.
66. Fleming, P. J. and Purshouse, R. C. (2002) ‘Evolutionary algorithms in control systems engineering: a survey’, *Control engineering practice*, 10(11), pp. 1223–1241.
67. Fulton, C. (2017) *Estimating time series models by state space methods in Python: Statsmodels*. Available at: http://www.chadfulton.com/files/fulton_statsmodels_2017_v1.pdf.
68. Gabbett, T., Jenkins, D. and Abernethy, B. (2009) ‘Game-based training for improving skill and physical fitness in team sport athletes’, *International Journal of Sports Science & Coaching*, 4(2), pp. 273–283.
69. Garthwaite, P. H., Kadane, J. B. and O’Hagan, A. (2005) ‘Statistical methods for eliciting probability distributions’, *Journal of the American Statistical Association*, 100(470), pp. 680–701.
70. Goodfellow, I. *et al.* (2016) *Deep learning*. MIT press Cambridge.
71. Gouba, E. *et al.* (2013) ‘Applying a Mathematical Model to the Performance of a Female Monofin Swimmer’, *Applied Mathematics*, 04(12), pp. 1673–1681. doi: 10.4236/am.2013.412228.
72. Greig, L. *et al.* (2020) ‘Autoregulation in Resistance Training: Addressing the Inconsistencies’, *Sports Medicine*. doi: 10.1007/s40279-020-01330-8.
73. Haff, G. G. (2010) ‘Sport science’, *Strength & Conditioning Journal*, 32(2), pp. 33–45.
74. Hamra, G., MacLehose, R. and Richardson, D. (2013) ‘Markov chain Monte Carlo: an introduction for epidemiologists’, *International journal of epidemiology*, 42(2), pp. 627–634. doi: 10.1093/ije/dyt043.
75. Hansen, N. (2016) ‘The CMA evolution strategy: A tutorial’, *arXiv preprint arXiv:1604.00772*.
76. Harris, G. R. *et al.* (2000) ‘Short-Term Performance Effects of High Power, High Force, or Combined Weight-Training Methods’, *Journal of Strength and Conditioning Research*. doi: 10.1519/00124278-200002000-00003.
77. Hayes, P. R. and Quinn, M. D. (2009) ‘A mathematical model for quantifying training’, *European Journal of Applied Physiology*, 106(6), pp. 839–847. doi: 10.1007/s00421-009-1084-8.
78. Hellard, P. *et al.* (2005) ‘Modeling the Residual Effects and Threshold Saturation of Training: A Case Study of Olympic Swimmers’, *Journal of Strength and Conditioning Research*, 19(1), pp. 67–75.
79. Hellard, P. *et al.* (2006) ‘Assessing the limitations of the Banister model in monitoring training’, *Journal of Sports Sciences*, 24(5), pp. 509–520. doi: 10.1080/02640410500244697.

80. Henao, W. (2014) ‘An L-BFGS-B-NS optimizer for non-smooth functions’, *Master’s thesis*.
81. Herold, J. L. and Sommer, A. (2020) ‘A mathematical model-based approach to optimize loading schemes of isometric resistance training sessions’, *bioRxiv*, p. 2020.04.16.044578. doi: 10.1101/2020.04.16.044578.
82. Hilbert, M. and Lopez, P. (2012) ‘How to Measure the World’s Technological Capacity to Communicate, Store, and Compute Information Part I: Results and Scope.’, *International Journal of Communication (19328036)*, 6.
83. Imbach, F. *et al.* (2020) ‘Training Load Responses Modelling in Elite Sports: How to Deal with Generalisation? (Preprint)’, *Research Square*.
84. Impellizzeri, F. M. *et al.* (2020) ‘Acute: chronic workload ratio: conceptual issues and fundamental pitfalls’, *International journal of sports physiology and performance*, 15(6), pp. 907–913.
85. Impellizzeri, F. M., Marcora, S. M. and Coutts, A. J. (2019) ‘Internal and external training load: 15 years on’, *International journal of sports physiology and performance*, 14(2), pp. 270–273.
86. Ishii, H. *et al.* (2008) ‘Prediction of swim performance in junior female swimmers by dynamic system model’, *Human Performance Measurement*, 5(1), pp. 1–8. Available at: http://www.shobix.co.jp/hpm/tempfiles/journal/2008/07J042.pdf.
87. James, D. A. and Petrone, N. (2016) *Sensors and Wearable Technologies in Sport: Technologies, Trends and Approaches for Implementation*. Springer.
88. Jeffreys, I. (2015) ‘Evidence based practice in strength and conditioning—reality or fantasy’, *Prof Strength Cond*, 39, pp. 7–14.
89. Jeffries, A. *et al.* (2020) ‘Development of a revised conceptual framework of physical training for measurement validation and other applications’, *Preprint*. doi: 10.31236/osf.io/wpvek.
90. Jobson, S. A. *et al.* (2009) ‘The analysis and utilization of cycling training data’, *Sports Medicine*, 39(10), pp. 833–844. doi: 10.2165/11317840-000000000-00000.
91. Johnson, S. R. *et al.* (2010) ‘Methods to elicit beliefs for Bayesian priors: a systematic review’, *Journal of clinical epidemiology*, 63(4), pp. 355–369.
92. Kellmann, M. *et al.* (2018) ‘Recovery and performance in sport: Consensus statement’, *International Journal of Sports Physiology and Performance*, 13(2), pp. 240–245. doi: 10.1123/ijspp.2017-0759.
93. Kolossa, D. *et al.* (2017) ‘Performance estimation using the fitness-fatigue model with Kalman filter feedback’, *International Journal of Computer Science in Sport*, 16(2), pp. 117–129. doi: 10.1515/ijcss-2017-0010.
94. Krzyzanski, W., Perez-Ruixo, J. J. and Vermeulen, A. (1999) ‘Basic pharmacodynamic models for agents that alter the lifespan distribution of natural cells’, *Journal of Pharmacokinetics and Biopharmaceutics*, 27(5), pp. 467–489. doi: 10.1007/s10928-008-9092-6.
95. Kuhn, M. *et al.* (2020) ‘Package “caret”’, *The R Journal*, p. 223.
96. Kumyaito, N., Yupapin, P. and Tamee, K. (2018) ‘Planning a sports training program using

Adaptive Particle Swarm Optimization with emphasis on physiological constraints', *BMC Research Notes*, 11(1), pp. 1–6. doi: 10.1186/s13104-017-3120-9.

97. Kuper, G. H. and Sterken, E. (2007) 'Modelling the development of world records in running', *Statistical thinking in sports*, pp. 7–31.
98. Lambert, J. D. (1991) *Numerical methods for ordinary differential systems*. Wiley New York.
99. Liebermann, D. G. *et al.* (2002) 'Advances in the application of information technology to sport performance', *Journal of Sports Sciences*. doi: 10.1080/026404102320675611.
100. Lucía, A. *et al.* (2003) 'Tour de France versus Vuelta a España: Which is harder?', *Medicine and Science in Sports and Exercise*, 35(5), pp. 872–878. doi: 10.1249/01.MSS.0000064999.82036.B4.
101. Ludwig, M. and Asteroth, A. (2016) 'Predicting performance from outdoor cycling training with the fitness-fatigue model', *Workshop Modelling in Endurance Sports*, pp. 3–6.
102. Ludwig, M., Schaefer, D. and Asteroth, A. (2016) 'Training simulation with nothing but training data simulating performance based on training data without the help of performance diagnostics in a laboratory', *icSPORTS 2016 - Proceedings of the 4th International Congress on Sport Sciences Research and Technology Support*, (November), pp. 75–82. doi: 10.5220/0006042900750082.
103. Ma, J., Bair, E. and Motsinger-Reif, A. (2020) 'Nonlinear Dose–Response Modeling of High-Throughput Screening Data Using an Evolutionary Algorithm', *Dose-Response*, 18(2), p. 1559325820926734.
104. Mangine, G. T. *et al.* (2008) 'The effects of combined ballistic and heavy resistance training on maximal lower- and upper-body strength in recreationally trained men', *Journal of Strength and Conditioning Research*. doi: 10.1519/JSC.0b013e31815f5729.
105. Mann, T. N., Lamberts, R. P. and Lambert, M. I. (2014) 'High responders and low responders: factors associated with individual variation in response to standardized training', *Sports Medicine*, 44(8), pp. 1113–1124.
106. Mannakee, B. K. *et al.* (2016) 'Sloppiness and the geometry of parameter space', in *Uncertainty in Biology*. Springer, pp. 271–299.
107. Matabuena, M. and Rodríguez-López, R. (2016) 'A new approach to predict changes in physical condition: A new extension of the classical Banister model', *arXiv preprint*. Available at: http://arxiv.org/abs/1612.08591.
108. Matabuena, M. and Rodríguez-López, R. (2019) 'An Improved Version of the Classical Banister Model to Predict Changes in Physical Condition', *Bulletin of Mathematical Biology*, 81(6), pp. 1867–1884. doi: 10.1007/s11538-019-00588-y.
109. McBride, J. M. *et al.* (2002) 'The effect of heavy- vs. light-load jump squats on the development of strength, power, and speed', *Journal of Strength and Conditioning Research*. doi: 10.1519/1533-4287(2002)016<0075:TEOHVL>2.0.CO;2.

110. McGarry, T. (2009) ‘Applied and theoretical perspectives of performance analysis in sport: Scientific issues and challenges’, *International Journal of Performance Analysis in Sport*, 9(1), pp. 128–140.

111. Méline, T. *et al.* (2018) ‘Systems model and individual simulations of training strategies in elite short-track speed skaters’, *Journal of Sports Sciences*, 37(3), pp. 347–355. doi: 10.1080/02640414.2018.1504375.

112. Mello, R., Leite, L. R. and Martins, R. A. (2014) ‘Is big data the next big thing in performance measurement systems?’, *Industrial and Systems Engineering Research Conference*, (May), pp. 1837–1846.

113. Millet, G. P. *et al.* (2002) ‘Modelling the transfers of training effects on performance in elite triathletes’, *International Journal of Sports Medicine*, 23(1), pp. 55–63. doi: 10.1055/s-2002-19276.

114. Millet, G. P. *et al.* (2005) ‘Modelling the relationships between training, anxiety, and fatigue in elite athletes’, *International Journal of Sports Medicine*, 26(6), pp. 492–498. doi: 10.1055/s-2004-821137.

115. Mitchell, L. J. G. *et al.* (2020) ‘The impact of different training load quantification and modelling methodologies on performance predictions in elite swimmers’, *European Journal of Sport Science*. doi: 10.1080/17461391.2020.1719211.

116. Mohammad, H. and Waziri, M. Y. (2019) ‘Structured two-point stepsize gradient methods for nonlinear least squares’, *Journal of Optimization Theory and Applications*, 181(1), pp. 298–317.

117. Morgulev, E., Azar, O. H. and Lidor, R. (2018) ‘Sports analytics and the big-data era’, *International Journal of Data Science and Analytics*, 5(4), pp. 213–222. doi: 10.1007/s41060-017-0093-7.

118. Morton, R. H. (1991) ‘The Quantitative Periodization of Athletic Training: A Model Study’, *Sports Medicine, Training and Rehab*, 3(1), pp. 19–28.

119. Morton, R. H., Fitz-clarke, J. R. and Banister, E. W. (1990) ‘Modeling Human Performance in Running’, *The American Physiological Society: Modeling methodology forum*, 69(3), pp. 1171–1177.

120. Morton, R. H. and Fukuba, Y. (2011) ‘Professor Eric W. Banister, 1933–2010: Obituary’, *Research in Sports Medicine*, 19(2), pp. 144–144. doi: 10.1080/15438627.2011.556533.

121. Moxnes, J. F. and Hausken, K. (2008) ‘The dynamics of athletic performance, fitness and fatigue’, *Mathematical and Computer Modelling of Dynamical Systems*, 14(6), pp. 515–533. doi: 10.1080/13873950802246473.

122. Mujika, I. *et al.* (1996) ‘Modeled responses to training and taper in competitive swimmers’, *Medicine and Science in Sports and Exercise*, 28(2), pp. 251–258. doi: 10.1097/00005768-199602000-00015.

123. Mullen, K. *et al.* (2011) ‘DEoptim: An R package for global optimization by differential

evolution', *Journal of Statistical Software*, 40(6), pp. 1–26.

124. Nash, J. C. (2014) *Nonlinear parameter optimization using R tools*. John Wiley & Sons.

125. Nash, J. C. *et al.* (2020) 'Package "optimx"'.

126. Nevill, A. M. and Whyte, G. (2005) 'Are there limits to running world records?', *Medicine and Science in Sports and Exercise*, 37(10), p. 1785.

127. O'Donoghue, P. (2014) *An introduction to performance analysis of sport*. Routledge.

128. Pappalardo, L. and Cintia, P. (2017) 'Quantifying the relation between performance and success in soccer', *Advances in Complex Systems*, 21, pp. 1–38.

129. Passfield, L. and Hopker, J. G. (2017) 'A Mine of Information: Can Sports Analytics Provide Wisdom From Your Data?', *International Journal of Sports Physiology and Performance*, 12(7), pp. 851–855. doi: 10.1123/ijspp.2016-0644.

130. Pearl, J. (2010) 'An introduction to causal inference', *The international journal of biostatistics*, 6(2), p. 7. doi: 10.2202/1557-4679.1203.

131. Peng, R. D. (2011) 'Reproducible research in computational science', *Science*. doi: 10.1126/science.1213847.

132. Perl, J. (2001) 'PerPot: A metamodel for simulation of load performance interaction', *European Journal of Sport Science*, 1(2), pp. 1–13. doi: 10.1080/17461390100071202.

133. Perl, J. and Pfeiffer, M. (2011) 'PerPot DoMo: Antagonistic Meta-Model Processing two Concurrent Load Flows', *International Journal of Computer Science in Sport*, 10/2011(2).

134. Pfeiffer, M. (2008) 'Modeling the Relationship between Training and Performance - A Comparison of Two Antagonistic Concepts', *International Journal of Computer Science in Sport*, 7(2), pp. 13–32.

135. Philippe, A. G. *et al.* (2015) 'Modeling the responses to resistance training in an animal experiment study', *BioMed Research International*, pp. 1–7. doi: 10.1155/2015/914860.

136. Philippe, A. G. *et al.* (2018) 'Modelling performance and skeletal muscle adaptations with exponential growth functions during resistance training', *Journal of Sports Sciences*, 37(3), pp. 254–261. doi: 10.1080/02640414.2018.1494909.

137. Piatrikova, E. *et al.* (2021) 'Monitoring the Heart Rate Variability Responses to Training Loads in Competitive Swimmers Using a Smartphone Application and the Banister Impulse-Response Model', *International Journal of Sports Physiology and Performance*, 1(aop), pp. 1–9.

138. Plisk, S. S. and Stone, M. H. (2003) 'Periodization Strategies', *Strength and Conditioning Journal*. doi: 10.1519/00126548-200312000-00005.

139. de Prado, M. L. (2018) *Advances in Financial Machine Learning*. 1st edn. New Jersey: Wiley.

140. Prado, M. L. De (2018) *Advances in Financial Machine Learning*. New Jersey: John Wiley & Sons Inc.

141. Proshin, A. P. and Solodyannikov, Y. V (2018) 'Physiological Avatar Technology with Optimal Planning of the Training Process in Cyclic Sports', *Automation and Remote Control*,

79(5), pp. 870–883. doi: 10.1134/S0005117918050089.

142. R Core Team (2020) ‘R: A Language and Environment for Statistical Computing’. Vienna, Austria: R Foundation for Statistical Computing. Available at: R-project.org.

143. Rasche, C. and Pfeiffer, M. (2019) ‘Training’, in Baca, A. and Perl, J. (eds) *Modelling and Simulation in Sport and Exercise*. New York, NY: Routledge, pp. 187–207.

144. Revie, M. *et al.* (2017) ‘On modeling player fitness in training for team sports with application to professional rugby’, *International Journal of Sports Science and Coaching*, 12(2), pp. 183–193. doi: 10.1177/1747954117694736.

145. Ripley, B. *et al.* (2013) ‘Package “mass”’, *Cran r*, 538, pp. 113–120.

146. Rozendaal, R. (2017) *Modeling performance of elite cyclists: The Effect of Training on Performance*. Delft University of Technology. Available at: http://repository.tudelft.nl/.

147. Sanchez, A. M. J. *et al.* (2013) ‘Modelling training response in elite female gymnasts and optimal strategies of overload training and taper’, *Journal of Sports Sciences*, 31(14), pp. 1510–1519. doi: 10.1080/02640414.2013.786183.

148. Scarf, P. *et al.* (2019) ‘Modelling the effect of training on performance in road cycling: estimation of the Banister model parameters using field data’, *arXiv preprint*, pp. 1–14.

149. Schaefer, D., Asteroth, A. and Ludwig, M. (2015) ‘Training plan evolution based on training models’, *IEEE Symposium*. doi: 10.1109/INISTA.2015.7276739.

150. Scrucca, L. (2013) ‘GA: a package for genetic algorithms in R’, *Journal of Statistical Software*, 53(4), pp. 1–37.

151. Seabold, S. and Perktold, J. (2010) ‘Statsmodels: Econometric and statistical modeling with python’, in *Proceedings of the 9th Python in Science Conference*, p. 61.

152. Selye, H. (1946) ‘The general adaptation syndrome and the diseases of adaptation’, *The journal of clinical endocrinology*, 6(2), pp. 117–230.

153. Selye, H. (1950) ‘Stress and the general adaptation syndrome’, *British medical journal*, 1(4667), p. 1383.

154. Selye, H. (1951) ‘The general-adaptation-syndrome’, *Annual review of medicine*, 2(1), pp. 327–342.

155. Sen, A. and Srivastava, M. (1990) *Regression Analysis: Theory, Methods and Applications*. New York: Springer. doi: 10.2307/3620306.

156. Shrahili, M. (2014) *Modelling and optimising the sport and exercise training process*. University of Salford.

157. Siedlok, F. and Hibbert, P. (2014) ‘The organization of interdisciplinary research: modes, drivers and barriers’, *International Journal of Management Reviews*, 16(2), pp. 194–210.

158. Skiba, P. F. (2008) *Analysis of power output and training stress in cyclists: The development of the BikeScoreTM algorithm*. Tech. rep. PhysFarm Training Systems LLC.

159. Smit, S. K. and Eiben, A. E. (2010) ‘Parameter tuning of evolutionary algorithms: Generalist

vs. specialist', in *European conference on the applications of evolutionary computation*. Springer, pp. 542–551.

160. Soetaert, K. E. R., Petzoldt, T. and Setzer, R. W. (2010) 'Solving differential equations in R: package deSolve', *Journal of statistical software*, 33.

161. Soetaert, K. and Petzoldt, T. (2010) 'Inverse modelling, sensitivity and monte carlo analysis in R using package FME', *Journal of Statistical Software*. doi: 10.18637/jss.v033.i03.

162. Stein, M. *et al.* (2017) 'How to Make Sense of Team Sport Data: From Acquisition to Data Modeling and Research Aspects', *Data*, 2(1), p. 2. doi: 10.3390/data2010002.

163. Stephens Hemingway, B. *et al.* (2019) 'The effects of measurement error and testing frequency in applying the Fitness Fatigue Model to resistance training : A simulation study', *International Journal of Sports Science and Coaching*, 0(0), pp. 1–12. doi: 10.13140/RG.2.2.19730.56005.

164. Stephens Hemingway, B. *et al.* (2021) 'Traditional and contemporary approaches to mathematical fitness-fatigue models in exercise science: A practical guide with resources. Part I.', *SportRxiv (Preprint)*. doi: 10.31236/osf.io/ap75j.

165. Stephens Hemingway, B., Swinton, P. A. and Ogorek, B. (2021) 'The suitability of a quasi-Newton algorithm for estimating fitness-fatigue models: Sensitivity, troublesome local optima, and implications for future research (An in silico experimental design)', *SportRxiv (Preprint)2*. doi: 10.31236/osf.io/dx7gm.

166. Stevens, J. (1986) *Applied Multivariate Statistics for the Social Sciences*. Hillsdale, NJ: Lawrence Erlbaum Associates. doi: 10.2307/2685203.

167. Stone, M. H., Sands, W. A. and Stone, M. E. (2004) 'The downfall of sports science in the United States', *Strength & Conditioning Journal*, 26(2), pp. 72–75.

168. Stone, M. H., Stone, M. and Sands, W. A. (2007) *Principles and practice of resistance training*. Human Kinetics.

169. Sun, W. and Yuan, Y.-X. (2006) *Optimization theory and methods: nonlinear programming*. Springer Science & Business Media.

170. Suzuki, S. *et al.* (2006) 'Program design based on a mathematical model using rating of perceived exertion for an elite Japanese sprinter: A case study', *Journal of Strength and Conditioning Research*, 20(1), pp. 36–42. doi: 10.1519/R-16914.1.

171. Swinton, P. *et al.* (2021) 'Traditional and contemporary approaches to mathematical fitness-fatigue models in exercise science: A practical guide with resources. Part II.', *SportRxiv (Preprint)*. doi: 10.31236/osf.io/5qgc2.

172. Swinton, P. A. *et al.* (2018) 'A Statistical Framework to Interpret Individual Response to Intervention: Paving the Way for Personalized Nutrition and Exercise Prescription', *Frontiers in Nutrition*. doi: 10.3389/fnut.2018.00041.

173. Taha, T. and Thomas, S. G. (2003) 'Systems Modelling of the Relationship between Training and Performance', *Sports Medicine*, 33(14), pp. 1061–1073. doi: 10.2165/00007256-200333140-

00003.

174. Thomas, L., Mujika, I. and Busso, T. (2008) ‘A model study of optimal training reduction during pre-event taper in elite swimmers’, *Journal of Sports Sciences*, 26(6), pp. 643–652. doi: 10.1080/02640410701716782.

175. Thomas, L., Mujika, I. and Busso, T. (2009) ‘Computer simulations assessing the potential performance benefit of a final increase in training during pre-event taper’, *Journal of Strength and Conditioning Research*, 23(6), pp. 1729–1736. doi: 10.1519/JSC.0b013e3181b3dfa1.

176. Transtrum, M. K. *et al.* (2015) ‘Perspective: Sloppiness and emergent theories in physics, biology, and beyond’, *Journal of Chemical Physics*. doi: 10.1063/1.4923066.

177. Trautmann, H. *et al.* (2015) ‘Package “cmaes”’.

178. Turner, J. D. *et al.* (2017) ‘A nonlinear model for the characterization and optimization of athletic training and performance’, *Biomedical Human Kinetics*, 9(1), pp. 82–93. doi: 10.1515/bhk-2017-0013.

179. Verkhoshansky, Y. and Siff, M. C. (2009) *Supertraining*. Verkhoshansky SSTM.

180. Wallace, L. K., Slattery, K. M. and Coutts, A. J. (2014) ‘A comparison of methods for quantifying training load: Relationships between modelled and actual training responses’, *European Journal of Applied Physiology*, 114(1), pp. 11–20. doi: 10.1007/s00421-013-2745-1.

181. Wang, C.-H. *et al.* (2013) ‘Open vs. closed skill sports and the modulation of inhibitory control’, *PloS one*, 8(2), p. e55773.

182. Wang, C. *et al.* (2020) ‘Analyzing activity and injury: lessons learned from the acute: chronic workload ratio’, *Sports Medicine*, 50(7), pp. 1243–1254.

183. Watkins, C. M. *et al.* (2017) ‘Determination of vertical jump as a measure of neuromuscular readiness and fatigue’, *Journal of Strength and Conditioning Research*. doi: 10.1519/JSC.0000000000002231.

184. Weston, S. and Computing, Re. (2015) ‘doSNOW: Foreach Parallel Adaptor for the “snow” Package’, in *Revolution Analytics*.

185. Wickham, H. (2019) *Advanced r*. CRC press.

186. Williams, S. *et al.* (2018) ‘Modelling the HRV response to training loads in elite rugby sevens players’, *Journal of Sports Science and Medicine*, 17(3), pp. 402–408.

187. Windt, J. and Gabbett, T. J. (2019) ‘Is it all for naught? What does mathematical coupling mean for acute: chronic workload ratios?’ BMJ Publishing Group Ltd and British Association of Sport and Exercise Medicine.

188. Wood, R. E. *et al.* (2005) ‘Applying a mathematical model to training adaptation in a distance runner’, *European Journal of Applied Physiology*, 94(3), pp. 310–316. doi: 10.1007/s00421-005-1319-2.

189. Wright, S. and Nocedal, J. (1999) ‘Numerical optimization’, *Springer Science*, 35(67–68), p. 7.

190. Yang, Z., Cai, X. and Fan, Z. (2014) ‘Epsilon constrained method for constrained multiobjective optimization problems: some preliminary results’, in *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 1181–1186.

191. Zarkadas, P. C., Carter, J. B. and Banister, E. W. (1995) ‘Modelling the effect of taper on performance, maximal oxygen uptake, and the anaerobic threshold in endurance triathletes’, *Advances in Experimental Medicine and Biology*, 393(1), pp. 179–186.

192. Zieffler, A. (2019) *EPsy 8252: Methods in Data Analysis for Educational Research II*. Available at: https://zief0002.github.io/book-8252/.

Appendix A: Mathematical derivations

The mathematical derivations of the standard and fitness-delay model are provided next [9]. The systems of differential equations that define fitness-fatigue models can be solved using the method of Laplace transform. The Laplace transform (denoted $\mathcal{L}$) is an integral transform that can be used to convert a differential equation into an algebraic equation. The algebraic equation can then be solved and then the inverse Laplace transform used to find the solution to the original differential equation that subsequently provides a simple method of computation. The Laplace transform converts a function of a real variable t, to a function of a complex variable s, where:

$$\mathcal{L}\{f(t)\} = F(s) = \int_0^\infty f(t)e^{-st}\,dt$$

To use the Laplace transform to solve differential equations, the Laplace transform of a derivative is required, where

$$\mathcal{L}\{f'(t)\} = sF(s) - f(0);\ \mathcal{L}\{f''(t)\} = s^2F(s) - sf(0) - f'(0)$$

Frequently the final step of solving the differential equation will involve the inverse Laplace transform of a product $F(s) = M(s)N(s)$ such that:

$$f(t) = (m * n)(t) = \int_0^t m(u)n(t-u)du$$

Where $m(t) * n(t)$ is the convolution of the functions m and n.

Also, the inverse Laplace transform of $1/(s-a)$ where a is a constant is e^{at}.

The standard model (Banister *et al.*, 1975)

With the above we can solve the independent system of differential equations that define the standard FFM

$$g'(t) = \omega(t) - \frac{1}{\tau_g}g(t)$$
$$h'(t) = \omega(t) - \frac{1}{\tau_h}h(t)$$

We simply solve each differential equation in isolation. Showing how this would be achieved for fitness $g(t)$.

[9] **These derivations were published in the appendices of:**

Stephens Hemingway, B., Greig, L., Jovanovic, M., Ogorek, B., & Swinton, P. (2021). Traditional and contemporary approaches to mathematical fitness-fatigue models in exercise science: A practical guide with resources. Part I. *SportRxiv (Preprint)*. doi.org/10.31236/osf.io/ap75j

Taking the Laplace transform of each term gives:

$$sG(s) - g(0) = \Omega(s) - \frac{1}{\tau_g} G(s)$$

Where $G(s)$ is the Laplace transform of $g(t)$ and $\Omega(s)$ is the Laplace transform of $\omega(t)$. Under the assumption $g(0) = 0$, we can rearrange the equation above to give the following:

$$sG(s) + \frac{1}{\tau_g} G(s) = \Omega(s)$$

$$G(s)\left(s + \frac{1}{\tau_g}\right) = \Omega(s)$$

$$G(s) = \Omega(s)\left(\frac{1}{\left(s + \frac{1}{\tau_g}\right)}\right)$$

On the right hand side, we have a product and therefore taking the inverse Laplace transform gives

$$g(t) = \omega(t) * e^{-\frac{t}{\tau_g}} = \int_0^t e^{\frac{-(t-u)}{\tau_g}} \omega(u) du$$

This is the solution that was presented in eq. 2.5 Given FFMs are generally conceptualised as impulses performed on single days, the integrals expressed in the solution of the system are discretised and approximated with the area of rectangles. Note there is a change from continuous time (t) to discrete time. In the exercise science literature, any training impulse on day t is omitted such that the index of the series terminates at $t - 1$ to give the following approximation:

$$g(t) = \Delta t \sum_{i=0}^{n-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_g}}$$

Setting $\Delta t = 1$ and $w(0) = 0$, we obtain

$$g(t) = \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_g}}$$

Repeating the same process for fatigue $h(t)$ and then combining with the scaling constants and baseline performance gives the familiar form presented for the standard FFM

$$\widehat{p}(t) = p^* + \underbrace{k_g \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_g}}}_{\text{fitness component}} - \underbrace{k_h \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_h}}}_{\text{fatigue component}}$$

Fitness-delay model (Calvert *et al.*, 1976)

We will now show that the system of differential equations that defines the FFM with a delay term introduced by Calvert et al. (Calvert *et al.*, 1976) is the following:

$$\left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)^{-1} g''(t) = \omega(t) - \left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)^{-1}\left(\frac{1}{\tau_{g_1}}+\frac{1}{\tau_{g_2}}\right) g'(t) - \left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)^{-1} \frac{1}{\tau_{g_1}\tau_{g_2}} g(t)$$

$$h'(t) = \omega(t) - \frac{1}{\tau_h} h(t)$$

As identified at the beginning of this appendix, the same routine can be performed on fatigue $h(n)$ to obtain

$$h(t) = \sum_{i=1}^{t-1} \omega_i \cdot e^{\frac{-(t-i)}{\tau_h}}$$

More work is required for $g(t)$. Taking the Laplace transform of each term gives

$$\left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)^{-1}\left(s^2G(s) - sg(0) - g'(0)\right) = \Omega(s) - \left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)^{-1}\left(\frac{1}{\tau_{g_1}}+\frac{1}{\tau_{g_2}}\right)\left(sG(s) - g(0)\right) - \left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)^{-1}\frac{1}{\tau_{g_1}\tau_{g_2}}G(s)$$

Setting $g(0) = g'(0) = 0$ and rearranging gives

$$\left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)^{-1}\left(s^2G(s) + \left(\frac{1}{\tau_{g_1}}+\frac{1}{\tau_{g_2}}\right)sG(s) + \frac{1}{\tau_{g_1}\tau_{g_2}}G(s)\right) = \Omega(s)$$

$$G(s)\left(s^2 + \left(\frac{1}{\tau_{g_1}}+\frac{1}{\tau_{g_2}}\right)s + \frac{1}{\tau_{g_1}\tau_{g_2}}\right) = \left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)\Omega(s)$$

$$G(s)\left(\left(s+\frac{1}{\tau_{g_1}}\right)\left(s+\frac{1}{\tau_{g_2}}\right)\right) = \left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)\Omega(s)$$

$$G(s) = \frac{\left(\frac{1}{\tau_{g_2}}-\frac{1}{\tau_{g_1}}\right)}{\left(s+\frac{1}{\tau_{g_1}}\right)\left(s+\frac{1}{\tau_{g_2}}\right)}\Omega(s)$$

Partial fraction decomposition gives:

$$G(s) = \Omega(s)\left(\frac{1}{\left(s+\frac{1}{\tau_{g_1}}\right)}\right) - \Omega(s)\left(\frac{1}{\left(s+\frac{1}{\tau_{g_2}}\right)}\right)$$

Application of the inverse Laplace transform gives

$$g(t) = \omega(t) * e^{-\frac{t}{\tau_{g_1}}} - \omega(t) * e^{-\frac{t}{\tau_{g_2}}} = \int_0^t e^{\frac{-(t-u)}{\tau_{g_1}}}\omega(u)du - \int_0^t e^{\frac{-(t-u)}{\tau_{g_2}}}\omega(u)du$$

Approximation and combining the fitness and fatigue components with the associated scaling coefficients and baseline performance gives Calvert's model presented in eq. 2.13 (Chapter 2).

$$\hat{p}(t) = p^* + k_g \cdot \sum_{i=1}^{t-1}\omega_i \cdot \left(\underbrace{e^{\frac{-(t-i)}{\tau_{g_1}}}}_{\text{effect}} - \underbrace{e^{\frac{-(t-i)}{\tau_{g_2}}}}_{\text{delay}}\right) - k_h \cdot \sum_{i=1}^{t-1}\omega_i \cdot e^{\frac{-(t-i)}{\tau_h}}$$

Appendix B: Literature tables

The first table in this appendix (B-1) provides a comprehensive overview and quick-reference list for the fitness-fatigue model literature body including theoretical, descriptive, and experimental research, as well as work demonstrating prospective applications of FFMs toward solving the upstream training program design problem. A set of secondary tables (B-2 to B-3) are provided to summarise research design information from primary research including methods used and population studied.

Table B-1: Overview of the literature

Table B-1: Summary of model literature

Index	Article	Study Type	Model(s)	Outline
1	(Banister *et al.*, 1975)	Model development & Experimental	Standard model	The seminal paper in the FFM literature. Proposal of first systems model of athletic performance in terms of the antagonistic response to training. Experimental aspect involved fitting to data of a top-class swimmer.
2	(Calvert *et al.*, 1976)	Model development & Experimental	Fitness-delay model	Same authors as Banister *et al.* (1975), led by T. Calvert, proposed the addition of a delay term within the fitness-component, and subsequently fitted the adjusted model to the data of (presumably) the same top-class swimmer as Banister *et al.* (1975).
3	(Corlett, 1977)	Experimental	Standard model	Data fitting study (training intervention or observation) and assessment of physiological correlates.
4	(Corlett, Calvert and Banister, 1978)	Experimental	Standard model	Data fitting study (training intervention or observation) and assessment of physiological correlates.
5	(Banister and Calvert, 1980)	No additional details found	No additional details found	No copy located. Canadian Journal of Applied Sport Science discontinued. Full reference provided in bibliography.
6	(Banister and Hamilton, 1985)	Experimental	Standard model	Data fitting study (training intervention or observation) and assessment of physiological correlates.
7	(Banister *et al.*, 1986)	No additional details found	No additional details found	No copy located. Conference proceedings. Full reference provided in bibliography.
8	(Morton, Fitz-clarke and Banister, 1990)	Experimental	Standard model	Data fitting study (training intervention or observation).
9	(Busso *et al.*, 1990)	Experimental	Standard model	Data fitting study (training intervention or observation) and assessment of physiological correlates.
10	(Busso, Carasso and Lacour, 1991)	Model development & Experimental	General model (1-4 components)	Proposal of the general FFM. Data fitting study (training intervention or observation), testing the statistical significance of increasing model complexity.
11	(Morton, 1991)	Computer-experiment	Fitness-delay model	Simulation study assessing model behaviour to different training load series under selected parameters.
12	(Fitz-Clarke, Morton and Banister, 1991)	Model development	Standard model	Presentation of the concept of influence curves to study model behaviour and to optimise training decisions in aspects such as peaking and tapering.
13	(Banister, Morton and Fitz-Clarke, 1992)	Experimental	Standard model	Data fitting study (training intervention or observation) and assessment of physiological correlates.

Index	Article	Study Type	Model(s)	Outline
14	(Busso *et al.*, 1992)	Model development & Experimental	General model + initial components	Proposal of the inclusion of initial components to the general model to capture previous effects. Data fitting study (training intervention or observation) and assessment of physiological correlates.
15	(Candau, Busso and Lacour, 1992)	Experimental	General model (1 component)	Data fitting study (training intervention or observation) and assessment of physiological correlates.
16	(Busso, Candau and Lacour, 1994)	Model development & Experimental	Standard model with modified fitness/fatigue components	Presentation of two possible approaches (one novel) to specifying the fitness and fatigue components within the standard model structure. Comparison of methods under a data fitting study (training intervention or observation).
17	(Zarkadas, Carter and Banister, 1995)	Experimental	Standard model	Study adjacent to the FFM literature, investigating responses to taper periods under different training loads
18	(Mujika *et al.*, 1996)	Experimental	Standard model	Data fitting study (training intervention or observation).
19	(Busso *et al.*, 1997)	Model development & Experimental	Time-varying model & standard model	Presentation of a novel time-varying model via a recursive least-squares algorithm. Data fitting study (training intervention or observation). Comparison of models carried out.
20	(Banister, Carter and Zarkadas, 1999)	Experimental	Standard model	Comparison of simulated taper profiles under the FFM with taper profiles implemented experimentally to assess correspondence with improvement in performance.
21	(Busso *et al.*, 2002)	Experimental	Time-varying model	Data fitting study (training intervention or observation).
22	(Millet *et al.*, 2002)	Experimental	General model	Data fitting study (training intervention or observation).
23	(Busso, 2003)	Model development & Experimental	VDR model / General model	Presentation of the novel variable dose-response FFM. Data fitting study (training intervention or observation). Comparison of models carried out.
24	(Taha and Thomas, 2003)	Review	Various	Review article broadly examining systems modelling of the relationship between training and performance. Covers topics such as existing models, quantification of training and performance, and evaluation of models as the state of the literature existed in 2003.
25	(Chiu and Barnes, 2003)	Descriptive	Standard	A conceptual discussion of the original FFM as it pertains to physical training and its association with the general adaption syndrome (GAS) model and theory of supercompensation.
26	(le Bris *et al.*, 2004)	Experimental	General model (1-4 components)	Data fitting study (training intervention or observation). Comparison of models carried out.
27	(Hellard *et al.*, 2005)	Model development & Experimental	Standard model + threshold saturation	Proposal of an external threshold saturation function to capture the concept of diminishing returns in response to linearly increasing training loads. Data fitting study (training intervention or observation).
28	(Millet *et al.*, 2005)	Experimental	General model	Data fitting study (training intervention or observation).
29	(Wood *et al.*, 2005)	Experimental	Standard model	Data fitting study (training intervention or observation) and assessment of physiological correlates.
30	(le Bris *et al.*, 2006a)	Experimental	General model	Data fitting study (training intervention or observation).
31	(le Bris *et al.*, 2006b)	Experimental	General model	Data fitting study (training intervention or observation).
32	(Hellard *et al.*, 2006)	Descriptive & Experimental	Standard model	An examination of the limitations of the standard model including misspecification and ill-conditioning. Data fitting study (training intervention or observation).
33	(Suzuki *et al.*, 2006)	Experimental	Standard model	Data fitting study (training intervention or observation).

Index	Article	Study Type	Model(s)	Outline
34	(Busso and Thomas, 2006)	Descriptive	N/A	A theoretical discussion and overview of how performance models and model simulations may be used to inform training planning.
35	(Thomas, Mujika and Busso, 2008)	Experimental	VDR model	Data fitting study (training intervention or observation).
36	(Ishii *et al.*, 2008)	Experimental	Standard model	Data fitting study (training intervention or observation).
37	(Pfeiffer, 2008)	Descriptive & Experimental	Standard model, PerPot Meta Model	Descriptive article examining FFM limitations. Data fitting study (training intervention or observation). Comparison of models carried out.
38	(Thomas, Mujika and Busso, 2009)	Experimental	VDR Model	Simulation and discussion of taper profiles under an FFM
39	(Chalencon *et al.*, 2012)	Experimental	Standard model	Data fitting study (training intervention or observation).
40	(Gouba *et al.*, 2013)	Model development & Experimental	Variable baseline performance model & standard model	Proposal of a variable baseline performance model to capture changes in the additive term. Data fitting study (training intervention or observation). Comparison of models carried out.
41	(Sanchez *et al.*, 2013)	Experimental	VDR model	Data fitting study (training intervention or observation)
42	(Clarke and Skiba, 2013)	Review	Standard + threshold saturation	Educational review article, introduction to FFMs, threshold saturation, influence curves, and model fitting.
43	(Wallace, Slattery and Coutts, 2014)	Experimental	VDR model	Data fitting study (training intervention or observation).
44	(Shrahili, 2014)	Experimental	Standard model	Data fitting study (training intervention or observation).
45	(Chalencon *et al.*, 2015)	Experimental	Standard model & VDR model	Data fitting study (training intervention or observation). Comparison of models carried out.
46	(Agostinho *et al.*, 2015)	Experimental	General model (1-2 components)	Data fitting study (training intervention or observation). Comparison of models carried out.
47	(Philippe *et al.*, 2015)	Experimental	General model (1-2 components) & VDR model	Data fitting study (training intervention or observation). Comparison of models carried out.
48	(Schaefer, Asteroth and Ludwig, 2015)	Training program design	Standard model	Implementation of the FFM as a response framework for optimising training programs under constraints via solver(s).
49	(Matabuena and Rodríguez-López, 2016, 2019)	Model development & experimental	Recursive delay-differential model	Proposal of a recursive delay-differential model. Assessment of model under fitted data.
50	(Ludwig, Schaefer and Asteroth, 2016)	Experimental	Standard model	Data fitting study (training intervention or observation).
51	(Ludwig and Asteroth, 2016)	Model development & experimental	Standard model + preload terms	Proposal of preload terms to improve model prediction (similar to initial components). Data fitting study (training intervention or observation).
52	(Turner *et al.*, 2017)	Model development & experimental & training program design	Nonlinear model system	Proposal of a nonlinear systems model to capture saturation of training load, diminishing returns, and overtraining effects within model behaviour. Assessment of the model under fitted data and demonstration of a training program design approach.
53	(Rozendaal, 2017)	Experimental	Fitness-delay model	Data fitting study (training intervention or observation).
54	(Kolossa *et al.*, 2017)	Model development & experimental	VDR model represented as a state-space model	Presented how the FFM can be specified as a state-space model, and demonstration of Kalman filtering to improve model prediction. Data fitting study (training intervention or observation).

Index	Article	Study Type	Model(s)	Outline
55	(Busso, 2017)	Model development & experimental	Secondary-signal model	Proposal of a secondary-signal model of training response. Data fitting study (training intervention or observation).
56	(Philippe *et al.*, 2018)	Model development & experimental	Exponential growth models	Proposal of several exponential-growth models. Data fitting study (training intervention or observation).
57	(Méline *et al.*, 2018)	Experimental	VDR model	Data fitting study (training intervention or observation).
58	(Kumyaito, Yupapin and Tamee, 2018)	Training program design	Standard	Implementation of the FFM as a response framework for optimising training programs under constraints via some solver(s).
59	(Williams *et al.*, 2018)	Experimental	Standard	Data fitting study (training intervention or observation).
60	(Proshin and Solodyannikov, 2018)	Training program design	Standard	Overview of the training program design problem under a response framework described by an FFM.
61	(Connor, Fagan and O'Neill, 2019)	Training program design	Standard	Implementation of the FFM as a response framework for optimising training programs under constraints via some solver(s).
62	(Scarf *et al.*, 2019)	Experimental	Standard	Data fitting study (training intervention or observation).
63	(Stephens Hemingway *et al.*, 2019)	Computer experiment	Standard	Assessment of the effects of measurement error and testing frequency on prediction accuracy of the FFM under an *in-silico* approach.
64	(Rasche and Pfeiffer, 2019)	Descriptive article (book chapter)	Standard to VDR model	A descriptive article covering several principles of performance modelling and FFM research.
65	(Connor and O'Neill, 2020)	Computer experiment	Standard model	Comparison of the differential evolution algorithm and the L-BFGS-B algorithm in fitting the standard FFM.
66	(Mitchell *et al.*, 2020)	Experimental	Standard model, VDR model, and rolling averages model (inc. exponentially weighted)	Data fitting study (training intervention or observation). Comparison of models.
67	(Imbach *et al.*, 2020)	Experimental / descriptive	Exponential growth models	Data fitting study (training intervention or observation). Some discussion of model evaluation and CV approaches.
68	(Stephens Hemingway *et al.*, 2021)	Review article	Standard, fitness-delay, VDR, threshold saturation, nonlinear,	Part 1 of a review series investigating traditional and contemporary methods and models in the FFM field.
69	(Swinton *et al.*, 2021)	Review article		Part 2 of a review series investigating traditional and contemporary methods and models in the FFM field.
70	(Piatrikova *et al.*, 2021)	Experimental	Standard model	Data fitting study (training intervention or observation).
71	(Stephens Hemingway, Swinton and Ogorek, 2021)	Computer experiment	Standard and fitness-delay models	Assessment of starting point sensitivity for the L-BFGS-B algorithm when fitting FFMs to data, and demonstration of existence of many local minima.

Tables B-2 & B-3: Experimental research (population, methods, estimation)

Table B-2: Criterion performance selection and measurement frequency, participant details, and training load quantification approaches over the FFM (human) experimental literature.

Index	Study	Performance	Duration	*n*	Domain	Level	Measurement frequency	TL quantification
1	(Banister *et al.*, 1975)	100m TT swim	105 days	1 (M)	Swimming	Elite	Weekly	Volume × Intensity
2	(Calvert *et al.*, 1976)	100m TT swim	4 seasons	1 (M)	Swimming	Elite	Weekly	Volume × Intensity
4	(Corlett, Calvert and Banister, 1978)	Not specified	4 months	1 (M)	General (cycle ergometry)	Not specified	Weekly	Not specified
6	(Banister and Hamilton, 1985)	TT run (distance subject specific)	300 days	5 (F)	Running	Amateur - Elite	No set frequency	Banister's TRIMP
8	(Morton, Fitz-clarke and Banister, 1990)	TT run and cycle ergometry testing	28 days	2 (M)	Running	Amateur - Intermediate	≥ 2 × weekly	Banister's TRIMP
9	(Busso *et al.*, 1990)	Maximal clean and jerk (1RM)	1 year	5 (M)	Weightlifting	Elite	Weekly	Volume × Intensity
10	(Busso, Carasso and Lacour, 1991)	Sustained power (W) for 1 hour cycle	14 weeks	8 (M)	Cycling	Sedentary	1-2 × weekly	Volume × Intensity
13	(Banister, Morton and Fitz-Clarke, 1992)	TT run (distance subject specific)	60 days	2 (M)	Running	Amateur	≥ 2 × weekly	Banister's TRIMP
14	(Busso *et al.*, 1992)	Maximal clean and jerk (1RM)	1 year	6 (M)	Weightlifting	Elite	Weekly	Volume × Intensity
16	(Busso, Candau and Lacour, 1994)	Hammer throw (distance)	37 weeks	1 (M)	Hammer	Intermediate	Varied (19× total)	Volume × Intensity
17	(Zarkadas, Carter and Banister, 1995)	5k TT run	98 days	11 (M)	Triathlon	Intermediate	1-2 × weekly	Banister's TRIMP
18	(Mujika *et al.*, 1996)	TT swim	44 weeks	10 (M) 8 (F)	Swimming	Elite	1 × every 2-3 weeks	Volume × Intensity
19	(Busso *et al.*, 1997)	Incremental cycle test to exhaustion	14 weeks	2 (U)	Cycling	Recreational	1 × other week	Volume × Intensity
20	(Banister, Carter and Zarkadas, 1999)	5k TT run	94 days	11 (M)	Triathlon	Not specified	≥ 1 × weekly	Banister's TRIMP

Index	Study	Performance	Duration	*n*	Domain	Level	Measurement frequency	TL quantification
21	(Busso *et al.*, 2002)	Maximal power sustained (W) for 5 minutes	15 weeks	6 (M)	General (cycle ergometry)	Recreational	3× weekly	Volume × Intensity
22	(Millet *et al.*, 2002)	Running (mean speed at lactate threshold for 30 mins), swimming (max effort repeats), triathlon events (competitive)	40 weeks	1 (M) 3 (F)	Triathlon	Elite	1-2 × weekly	Banister's TRIMP
23	(Busso, 2003)	Maximal power sustained (W) for 5 minutes	15 weeks	6 (M)	General (cycle ergometry)	Recreational	3× weekly	Volume × Intensity
26	(le Bris *et al.*, 2004)	Time to exhaustion (cycle ergometer)	14 weeks	1 (M)	General (cycle ergometry)	Cardiac Rehabilitation	2× weekly	Banister's TRIMP
27	(Hellard *et al.*, 2005)	Competition swim times over a season	100-200 weeks (subject dependent)	3 (M) 4 (F)	Swimming	Elite	Total: mean = 48.7 ± 9 over a period of 4 ± 2 years	Volume × Intensity
28	(Millet *et al.*, 2005)	Anxiety and wellness scores	40 weeks	1 (M) 3 (F)	Triathlon	Elite	2× weekly	Banister's TRIMP
29	(Wood *et al.*, 2005)	3km TT run	12 weeks	1 (M)	Running	Intermediate	Weekly	Volume × Intensity
30	(le Bris *et al.*, 2006a)	6-minute walk test (metres)	14 weeks	4 (M)	General (treadmill and cycle ergometry)	Coronary artery disease patients	1-2 × weekly	Banister's TRIMP
31	(le Bris *et al.*, 2006b)	6-minute walk test (metres)	14 weeks	4 (M)	General (treadmill and cycle ergometry)	Coronary artery disease patients	1-2 × weekly	Banister's TRIMP
32	(Hellard *et al.*, 2006)	Competitive swim times over a season	60 weeks	4 (M) 5 (F)	Swimming	Elite	Total: mean = 13.2 ± 2.4 over a period of 60 weeks	Volume × Intensity
33	(Suzuki *et al.*, 2006)	400m TT run	1 year	1 (U)	Sprinting	Elite	1× every 3-4 weeks	Volume × Intensity
35	(Thomas, Mujika and Busso, 2008)	Competitive swim times over a season	2 seasons	4 (M) 4 (F)	Swimming	Elite	1× every 1-2 weeks	Volume × Intensity
36	(Ishii *et al.*, 2008)	200m TT swim	134 days	2 (F)	Swimming	Junior	1× every 2 weeks	Volume × Intensity
37	(Pfeiffer, 2008)	Power in all out 8s cycle sprint and 15s Wingate test	17 weeks	3 (M) 3 (F)	General (cycle ergometry)	Recreational	2 × weekly	Total watts (W)
38	(Thomas, Mujika and Busso, 2009)	Mean power (W) for 5-minute all-out test	64-84 days	10(M) 3 (F)	Swimming	Elite (7), unspecified (6)	1-3 × weekly	Volume × Intensity or Banister's TRIMP
39	(Chalencon *et al.*, 2012)	400m TT swim	30 weeks	6 (M) 4 (F)	Swimming	Intermediate - Elite	Weekly	Volume × Intensity

Index	Study	Performance	Duration	*n*	Domain	Level	Measurement frequency	TL quantification
40	(Gouba *et al.*, 2013)	700m TT swim	23 weeks	1 (F)	Monofin Swimming	Elite	1× every 3-4 weeks	Banister's TRIMP & Volume × Intensity
41	(Sanchez *et al.*, 2013)	Maximum chin-ups in 15 seconds	3 months	5 (F)	Gymnastics	Elite	3 × weekly	Volume × Intensity
43	(Wallace, Slattery and Coutts, 2014)	1500m TT run	15 weeks	7 (U)	Running	Amateur	Weekly	Banister's TRIMP & sRPE & rTSS score
44	(Shrahili, 2014)	Extrapolated from training data based on relationship between power and HR	300 days	10 (U)	Cycling	Intermediate - Elite	Approximately 2-3 × weekly	Banister's TRIMP
45	(Chalencon *et al.*, 2015)	Mean velocity during 400m TT swim	30 weeks	6 (M) 4 (F)	Swimming	Intermediate - Elite	Weekly	Volume × Intensity
46	(Agostinho *et al.*, 2015)	Competitive performance (points) & series of physical tests	2 years	10 (M)	Judo	Junior Intermediate	15 measurements total (per athlete)	sRPE and RPE
47	(Philippe *et al.*, 2015)	Ladder climbing with weight for time	4 weeks	11 (U)	Animal (rat)	-	5 × weekly	Work done (W)
49	(Matabuena and Rodríguez-López, 2016, 2019)	5-min power test (W) – Study used data provided in Clarke & Skiba (2013)	165 days	1	General (cycle ergometry)	Recreational	9 × Total	BikeScore (Skiba, 2008)
50	(Ludwig, Schaefer and Asteroth, 2016)	60 minutes peak power output (W) at crank	Up to 2 years (~600 days)	20 (M)	Cycling	Amateur	Not specified	TSS
51	(Ludwig and Asteroth, 2016)	60 minutes peak power output (W) at crank	140 days	20 (M)	Cycling	Amateur	Not specified	TSS
52	(Turner *et al.*, 2017)	Cyclists average power output over the most intense 10-minute interval during each exercise bout	500 days	1 (U)	Cycling	Not specified	18 measurements total over 500 days	BikeScore (Skiba, 2008)
53	(Rozendaal, 2017)	Critical power (W) at the crank	2-5 years	6 (U)	Cycling	Elite	22-24 total per athlete	Linear combination of attributes (volume, intensity, TSS, subjective load, objective load)
54	(Kolossa *et al.*, 2017)	Semi-tethered swim trial (3x repetitions of 20m with increasing resistance) – performance expressed as mean velocity for 60m	160 days	5 (U)	Swimming	Not specified	~ 1 × weekly	Volume × Intensity

Index	Study	Performance	Duration	*n*	Domain	Level	Measurement frequency	TL quantification
55	(Busso, 2017)	Maximal power sustained (W) for 5 minutes	15 weeks	6 (M)	General (cycle ergometry)	Recreational	3× weekly	Volume × Intensity
56	(Philippe *et al.*, 2018)	Ladder climbing with weight for time	4 weeks	11 (U)	Animal (rat)	-	5 × weekly	Work done (W)
57	(Méline *et al.*, 2018)	1-lap maximal effort race	3 months	5 (F) 10(M)	Speed skating	Elite	5 × weekly	Volume × Intensity
59	(Williams *et al.*, 2018)	HRV	8 weeks	8 (M)	Rugby Sevens	Elite	Daily	sRPE
62	(Scarf *et al.*, 2019)	Four metrics (power / HR related)	6-14 months	10 (M)	Cycling	Intermediate - Elite	Approximately 2-3 × weekly	Banister's TRIMP
66	(Mitchell *et al.*, 2020)	Time trials and race results (50m/100m)	66-85 weeks	2 (M) 1 (F)	Swimming	Elite	Approximately 35 total per subject	Volume × Intensity
67	(Imbach *et al.*, 2020)	Standing start time-trials (1.5 lap maximum effort)	3 months	3 (M) 4(F)	Speed Skating	Elite	1 × weekly	Combination of multiple measures
70	(Piatrikova *et al.*, 2021)	Not accessible						

(M) male, (F) female, (U) unspecified. Volume × Intensity indicates a training load quantification strategy that involved some type of sum of total load or distance multiplied by intensity (possibly based on zones or coefficients weighting each exercise). TL = Training load.

Table B-3: Model fitting and evaluation approaches across the experimental FFM literature

Index	Study	Model fitting		Model evaluation	
		Objective Function	Algorithm / Solver	In-sample (Y/N)	Out-of-sample (Y/N)
1	(Banister *et al.*, 1975)	RSS	Not specified	Y	N
2	(Calvert *et al.*, 1976)	Not specified	Not specified	Visual plots	N
3	(Corlett, 1977)	MSE	Not specified	Y	N
6	(Banister and Hamilton, 1985)	RSS	Not specified	N	N
8	(Morton, Fitz-clarke and Banister, 1990)	RSS	Not specified	Y	N
9	(Busso *et al.*, 1990)	RSS	Relaxation method with constraints	Y	N
10	(Busso, Carasso and Lacour, 1991)	RSS	Multiple linear regression	Y	N
13	(Banister, Morton and Fitz-Clarke, 1992)	RSS	Not specified	Y	N
14	(Busso *et al.*, 1992)	RSS	Not specified	Y	N
16	(Busso, Candau and Lacour, 1994)	MSE	Multiple linear regression	Y	N
18	(Mujika *et al.*, 1996)	RSS	Not specified	Y	N
19	(Busso *et al.*, 1997)	Recursive RSS	Not specified	Y	N
20	(Banister, Carter and Zarkadas, 1999)	RSS	Not specified	N	N
21	(Busso *et al.*, 2002)	Recursive RSS	Not specified	Y	N
22	(Millet *et al.*, 2002)	RSS	Not specified	Y	N
23	(Busso, 2003)	RSS	Not specified	Y	N
26	(le Bris *et al.*, 2004)	RSS	Not specified	Y	N
27	(Hellard *et al.*, 2005)	RSS	Gauss-Newton	Y	N
28	(Millet *et al.*, 2005)	RSS	Not specified	Y	N
29	(Wood *et al.*, 2005)	RSS	Not specified	Y	N
30	(le Bris *et al.*, 2006a)	RSS	Not specified	Y	N
31	(le Bris *et al.*, 2006b)	RSS	Not specified	Y	N
32	(Hellard *et al.*, 2006)	RSS	Gauss-Newton	Y	N
33	(Suzuki *et al.*, 2006)	RSS	Multiple linear regression	Y	N
35	(Thomas, Mujika and Busso, 2008)	RSS	Not specified	Y	N
36	(Ishii *et al.*, 2008)	RSS	Not specified	Y	N
37	(Pfeiffer, 2008)	RSS	Not specified	Y	N

Index	Study	Model fitting		Model evaluation	
		Objective Function	Algorithm / Solver	In-sample (Y/N)	Out-of-sample (Y/N)
39	(Chalencon *et al.*, 2012)	RSS	Not specified	Y	N
40	(Gouba *et al.*, 2013)	RSS	Alienor method and OPO	Y	N
41	(Sanchez *et al.*, 2013)	RSS	Not specified	Y	N
43	(Wallace, Slattery and Coutts, 2014)	RSS	Not specified	Y	N
44	(Shrahili, 2014)	Log-likelihood	Not specified – considered starting point sensitivity	Y	N
45	(Chalencon *et al.*, 2015)	RSS	Not specified	N	Y
46	(Agostinho *et al.*, 2015)	RSS	Not specified	Y	N
47	(Philippe *et al.*, 2015)	RSS	Generalised reduced gradient (GRG)	Y	N
49	(Matabuena and Rodríguez-López, 2016, 2019)	RSS	Not specified	N	N
50	(Ludwig, Schaefer and Asteroth, 2016)	RSS	Quasi-Newton	Y	Y
51	(Ludwig and Asteroth, 2016)	Not specified	Not specified	Y	Y
52	(Turner *et al.*, 2017)	Not specified	Genetic algorithm	Y	Y
54	(Kolossa *et al.*, 2017)	RSS	Multi-start interior point search algorithm	Y	Y
55	(Busso, 2017)	Log Likelihood	Newton-type (package NLM in R)	Y	N
56	(Philippe *et al.*, 2018)	RSS	Genetic algorithm	Y	N
57	(Méline *et al.*, 2018)	RSS	Genetic algorithm	Y	N
59	(Williams *et al.*, 2018)	RSS	Not specified (Excel solver)	Y	Y
62	(Scarf *et al.*, 2019)	Log Likelihood	Not specified	Y	N
66	(Mitchell *et al.*, 2020)	RSS	Two stage, BFGS-B & Nelder-Mead	Y	Y
67	(Imbach *et al.*, 2020)	RSS	Newton-type	Y	Y
70	(Piatrikova *et al.*, 2021)	Not accessible	Not accessible	Not accessible	

MSE: Mean-squared errors; **RSS**: Residual sum of squares (equivalent to sum of squared errors);

OPO: Optimisation preserving operators' technique

Appendix C: Algorithms

C-1 NLS sensitivity experiment (from chapter 5)

Requirements:

- Let $f(\theta, \omega)$ denote the fitness-fatigue model function
- Let θ be the parameter space $\theta = (\theta_1, \theta_2, \theta_3, \dots, \theta_m) \in \mathbb{R}^+$ of f
- Let ω denote the training load series $\omega = \{\omega_1, \omega_2, \dots, \omega_n\}$ of length n and time-step 1 day.

BEGIN MAINPROGRAM
  SET lower bound on θ: $l = (l_1, l_2, l_3, \dots, l_m), l_i \in \mathbb{R}^+$
  SET upper bound on θ: $u = (u_1, u_2, u_3, \dots, u_m),\ u_i \in \mathbb{R}^+, u_i > l_i$
  SET η where η^m is the total size of the grid $(\eta, m \in \mathbb{N})$
  SET γ where $n^m \bmod \gamma \neq 0,\ \gamma \in \mathbb{N}$
  COMPUTE Sample η equally spaced points from each open interval $(l_i, u_i), i \in [1, m]$
  COMPUTE Make grid_θ of size η^m by taking all combinations
  COMPUTE Partition grid_θ into γ equal segments $\text{grid}_\theta \equiv \left\{\text{grid}_\theta^{(1)}, \text{grid}_\theta^{(2)}, \dots., \text{grid}_\theta^{(\gamma)}\right\}$
  SET $\theta_{TRUE} \in (l, u)$
  INPUT ω
  COMPUTE vector $p \in \mathbb{R}$ of length n by computing $f(\theta_{TRUE}, \omega)$
  FOR ν = 1 to 3
    INITALISE empty array RESULT_ν of dim $\left(n^m \times (m+3)\right)$
    COMPUTE the sequence s, where s goes from 1 to n by ν
    COMPUTE p_s by sampling p at each point s
    FOR j = 1 to γ
      INPUT $\text{grid}_\theta^{(j)}$ into working memory
      DO IN PARALLEL FOR $k = 1$ to n^m/γ
        COMPUTE $\hat{p}_k$ from $f(\theta_k, \omega),\ \theta_k \in \text{grid}_\theta^{(j)}$
        COMPUTE $\hat{p}_{k_s}$ by sampling $\hat{p}_k$ at each point s
        COMPUTE $\text{RSS}_{\text{INIT}} = \sum\left(\hat{p}_{k_s} - p_s\right)^2$
        OUTPUT Store RSS_{INIT} to array RESULT_ν
          INITIALISE The quasi-Newton method at starting point θ_k
          SOLVE $\min \sum (f(\theta_{FIT}, \omega) - p_s)^2$ s.t. $\theta_{FIT} \in [u, l]$
          OUTPUT Row-bind θ_{FIT} and RSS_{FIT} to array RESULT_ν
          COMPUTE $\hat{p}_{\theta_{FIT}}$ from the model for $\{\theta_{FIT}, \omega\}$
          COMPUTE Fit statistic $\epsilon = \sqrt{\sum\left(\hat{p}_{\theta_{FIT}} - p_s\right)^2}$ (example RMSE)
          OUTPUT Store ϵ to array RESULT_ν
      END PARALLEL FOR
      INPUT Pull grid segments $\left\{\text{grid}_\theta^{(1)}, \dots, \text{grid}_\theta^{(\gamma)}\right\}$ into working memory
      Recombine grid_θ and remove segments
      Columnar-bind grid_θ with RESULT_ν: yields array of dim $\left(n^m \times (2m+3)\right)$
    END FOR
  END FOR
END MAINPROGRAM

Algorithm Notes

- The algorithm was written in pseudocode to make it language agnostic and therefore most understandable.
- R stores and manipulates all objects in the physical memory and therefore to conserve working memory during the implementation we opted to split and save the grid into smaller segments that would be loaded in sequential order. In our experiment we only had 8GB of RAM available on the machine. The constant γ is the number of smaller grids used to conserve memory.
- **`DO IN PARALLEL`** indicates that the operations were distributed to available nodes (via multi-core) and executed in parallel. In our experiment we used 8 available nodes within a single machine.
- Dimensions of arrays are given in `rows × column` format.

C-2 Expanding window cross-validation (from chapter 6)

Requirements:

- $[l, u]$ are the box constraints (bounds) established on the parameter space θ
- $Data_{B1}$ is a time-series data set of training loads and measured performance values $\{\omega_{B1}, p_{B1}\}$, where loads (ω_{B1}) are quantified from some relatively homogenous training program $(B1)$
- $Data_{B2}$ is a time-series data set of training loads and measured performance values $\{\omega_{B2}, p_{B2}\}$, where loads (ω_{B2}) are quantified from some relatively homogenous training program $(B2)$ where $B1 \not\equiv B2$.
- $Data_{B1}$ proceeds $Data_{B2}$ in time-order (consecutive series).
- N_{starts} is the number of starting points in the parameter space to initiate the optimisation algorithm from (fitting iterations):
    - This pseudocode assumes a derivative-based method is used that requires repeated fitting from multiple starting points to find the best or 'good' solution. For stochastic derivative-free methods, the equivalent process is repetition of fitting (with different seeds).

BEGIN MAINPROGRAM

COMPUTE Partition $Data_{B1}$ into S expanding splits $\left\{\left(split_{train}^{(1)}, split_{test}^{(1)}\right), \ldots, \left(split_{train}^{(S)}, split_{test}^{(S)}\right)\right\}$,

Where $\left(split_{train}^{(i)}\right) = \left\{\omega_{train}^{(i)}, p_{train}^{(i)}\right\}$ and $\left(split_{test}^{(i)}\right) = \left\{\omega_{test}^{(i)}, p_{test}^{(i)}\right\}$ for $(i = 1, \ldots, S)$

COMPUTE Sample the space $[l, u]$ N_{starts} times, building a set of starting points θ_{starts}

DO IN PARALLEL FOR i = 1 to S (Train splits in Parallel)

FOR j = 1 to N_{starts}

INITIALISE optimisation algorithm at $\theta_{starts}^{(j)}$

SOLVE train model via MLE on $split_{train}^{(i)}$, returns fitted parameters $\theta^{(i,j)}$

COMPUTE model $predictions^{(i,j)}$ under fitted parameters $\theta^{(i,j)}$ and $\omega = \{\omega_{train}^{(i)}, \omega_{test}^{(i)}\}$

COMPUTE $\epsilon^{(i,j)} \leftarrow$ Evaluate metric of $predictions^{(i,j)}$ on $split_{test}^{(i)}$ (e.g., MAPE or RMSE)

END FOR

COMPUTE Average model performance for the split (across iterations) $\leftarrow \frac{1}{N_{starts}} \sum_{j=1}^{N_{starts}} \left(\epsilon^{(i,j)}\right)$

END PARALLEL FOR

COMPUTE average model performance across all train-test splits $\leftarrow \frac{1}{(N_{starts} \times S)} \sum_{i=1}^{S} \left(\sum_{j=1}^{N_{starts}} \left(\epsilon^{(i,j)}\right)\right)$

FOR j = 1 to N_{starts}

INITIALISE optimisation algorithm at $\theta_{starts}^{(j)}$

SOLVE train model via MLE on $Data_{B1}$, returns fitted parameters $\theta_{main}^{(j)}$

COMPUTE model $predictions_{main}^{(j)}$ under fitted parameters $\theta_{main}^{(j)}$ and $\omega = \{\omega_{B1}, \omega_{B2}\}$

COMPUTE $\epsilon_{main}^{(j)} \leftarrow$ Evaluate metric of $predictions_{main}^{(j)}$ on p_{B2}

END FOR

COMPUTE average model performance for main train-test iterations $\leftarrow \frac{1}{N_{starts}} \sum_{j=1}^{N_{starts}} \left(\epsilon_{main}^{(j)}\right)$

COMPUTE $\theta_{optimal} \leftarrow$ set θ_{main}^{k} where $k = \min_{k} \epsilon_{main}^{(k)}$ from $(k = 1, \ldots, N_{starts})$

END MAINPROGRAM

Algorithm Notes

- The algorithm was written in pseudocode to make it language agnostic and therefore most understandable.
- Large values of N (> 1000) may require consideration of memory allocation limits if implemented in R
- **DO IN PARALLEL** indicates that the operations are to be parallelised by some means (forking, multicore etc.)
- MLE refers to maximum likelihood estimation. Least-squares could equally be used

Appendix D: Chapter 5 supplementary materials

D-1 Parameter estimate distributions (standard model)

Table D-1: Parameter estimate and RSS distributions of solutions obtained for the standard model

Scenario	Convergence	Summary statistics	p^*	k_g	τ_{g_1}	k_h	τ_h	$RSS_{solutions}$	$RSS_{initial}$
Standard model 100% data	True parameters (N = 69204)	Min	100.00	0.72	28.47	1.20	8.58	0.00	288
		Max	100.00	0.72	28.54	1.20	8.61	0.00	12893022
		Median	100.00	0.72	28.50	1.20	8.60	0.00	529882
		M.A.D.	0.00	0.00	0.00	0.00	0.00	0.00	710954
	Other solutions (N = 30763)	Min	90.55	0.01	1.19	0.01	1.93	11.17	457
		Max	116.97	5.00	50.00	5.00	50.00	3019.36	12771646
		Median	100.16	4.60	18.82	5.00	15.37	11.17	616682
		M.A.D.	0.00	0.00	0.00	0.00	0.00	0.00	845510
	Abnormal termination (N = 33)	Min	100.16	4.60	18.82	5.00	15.37	11.17	13268
		Max	100.16	4.60	18.82	5.00	15.37	11.17	7401559
		Median	100.16	4.60	18.82	5.00	15.37	11.17	655974
		M.A.D.	0.00	0.00	0.00	0.00	0.00	0.00	860297
Standard model (50% data)	True parameters (N = 69145)	Min	100.00	0.72	28.47	1.20	8.57	0.00	135
		Max	100.00	0.72	28.56	1.20	8.62	0.00	6470310
		Median	100.00	0.72	28.50	1.20	8.60	0.00	267974
		M.A.D.	0.00	0.00	0.00	0.00	0.00	0.00	359878
	Other solutions (N = 30823)	Min	90.55	0.01	1.19	0.01	1.93	0.00	457
		Max	116.97	5.00	50.00	5.00	50.00	3019.36	12858865
		Median	100.16	3.81	18.82	5.00	15.37	11.17	609208
		M.A.D.	0.24	1.18	14.35	0.00	10.04	16.56	830985
	Abnormal termination (N = 32)	Min	100.09	4.61	18.91	5.00	15.47	5.66	7159
		Max	100.09	4.61	18.91	5.00	15.47	5.66	4484160
		Median	100.09	4.61	18.91	5.00	15.47	5.66	480631
		M.A.D.	0.00	0.00	0.00	0.00	0.00	0.00	555066
Standard model (33% data)	True parameters (N = 70284)	Min	100.00	0.72	28.43	1.20	8.57	0.00	101
		Max	100.00	0.72	28.57	1.20	8.63	0.00	4266927
		Median	100.00	0.72	28.50	1.20	8.60	0.00	178693
		M.A.D.	0.00	0.00	0.00	0.00	0.00	0.00	239610
	Other solutions (N = 29676)	Min	91.75	0.01	1.00	0.01	1.68	2.89	306
		Max	116.14	5.00	50.00	5.00	50.00	1370.32	4018048
		Median	99.94	4.62	18.98	5.00	15.57	2.89	197519
		M.A.D.	0.00	0.00	0.00	0.00	0.00	0.00	269993
	Abnormal termination (N = 40)	Min	93.74	0.33	18.98	0.01	15.57	2.89	1920
		Max	99.94	5.00	50.00	5.00	49.64	154.04	2239363
		Median	99.94	4.62	18.98	5.00	15.57	2.89	415359
		M.A.D.	0.00	0.00	0.00	0.00	0.00	0.00	477229

M.A.D refers to the median absolute deviation. Data % refers to the proportion of data used in the fitting process (i.e., 100% corresponds to a measurement frequency of every day, 50% to every second day, 33% to every 3rd day). Other solutions include all non-true critical points (i.e., saddle and local minima). All parameter estimates rounded to 2.d.p, fitted RSS values to 3.d.p.

D-2 Parameter estimate distributions (fitness-delay model)

Table D-2: Parameter estimate and RSS distributions of solutions obtained for the fitness-delay model

Scenario	Convergence	Summary statistics	p^*	k_g	τ_{g_1}	τ_{g_2}	k_h	τ_h	$RSS_{solutions}$	$RSS_{initial}$
Fitness-delay model 100% data	**True parameters (N = 20588)**	**Min**	100.00	0.72	32.45	4.29	1.05	8.55	0.000	544
		Max	100.00	0.72	32.56	4.32	1.05	8.65	0.001	35463988
		Median	100.00	0.72	32.50	4.30	1.05	8.60	0.000	897537
		M.A.D.	0.00	0.00	0.00	0.00	0.00	0.00	0.000	1180419
	Other solutions (N = 96717)	**Min**	99.94	0.65	1.04	2.06	0.71	1.00	0.010	143
		Max	121.18	5.00	44.70	19.29	1.97	50.00	4380.976	41612914
		Median	100.04	1.53	32.16	6.17	1.05	21.87	0.121	1060962
		M.A.D.	0.04	0.17	5.94	0.53	0.00	25.09	0.164	1454207
	Abnormal termination (N = 344)	**Min**	100.01	0.74	27.86	5.65	1.04	4.84	0.011	2648
		Max	100.10	1.69	44.69	10.45	1.05	50.00	0.598	15290067
		Median	100.09	1.68	32.23	6.36	1.04	31.11	0.457	1090636
		M.A.D.	0.01	0.02	5.72	0.06	0.00	11.20	0.130	1479867
Fitness-delay model (50% data)	**True parameters (N = 20651)**	**Min**	100.00	0.72	32.43	4.28	1.05	8.54	0.000	273
		Max	100.00	0.72	32.57	4.32	1.05	8.66	0.000	18983885
		Median	100.00	0.72	32.50	4.30	1.05	8.60	0.000	451439
		M.A.D.	0.00	0.00	0.00	0.00	0.00	0.00	0.000	592054
	Other solutions (N = 96184)	**Min**	99.94	0.65	1.04	2.06	0.71	1.00	0.000	143
		Max	121.18	5.00	44.70	19.29	1.97	50.00	4380.976	41612914
		Median	100.04	1.53	32.16	5.81	1.05	21.86	0.121	1060179
		M.A.D.	0.04	0.25	5.94	0.90	0.00	19.65	0.164	1453494
	Abnormal termination (N = 157)	**Min**	99.96	0.73	19.34	5.64	0.80	4.88	0.006	539
		Max	100.48	5.00	41.40	14.23	1.06	45.27	37.537	8804668
		Median	100.06	1.65	30.65	6.29	1.04	26.70	0.181	540733
		M.A.D.	0.02	0.06	2.32	0.14	0.00	5.42	0.144	743087
Fitness-delay model (33% data)	**True parameters (N = 21065)**	**Min**	100.00	0.72	32.41	4.28	1.05	8.53	0.000	181
		Max	100.00	0.73	32.58	4.32	1.05	8.68	0.001	10781682
		Median	100.00	0.72	32.50	4.30	1.05	8.60	0.000	307321
		M.A.D.	0.00	0.00	0.00	0.00	0.00	0.00	0.000	403900
	Other solutions (N = 96486)	**Min**	94.31	0.65	1.00	1.78	0.01	1.78	0.004	46
		Max	119.93	5.00	44.71	30.11	1.94	50.00	1514.567	13746207
		Median	100.04	1.51	32.12	6.16	1.05	21.53	0.042	348259
		M.A.D.	0.04	0.26	6.02	0.59	0.00	24.62	0.058	476901
	Abnormal termination (N = 98)	**Min**	100.01	0.74	27.83	5.63	1.05	4.87	0.004	2255
		Max	100.11	1.69	38.48	10.33	1.05	40.99	0.227	7949920
		Median	100.09	1.66	30.37	6.24	1.05	27.20	0.154	429570
		M.A.D.	0.02	0.05	1.97	0.15	0.00	4.25	0.083	593110

M.A.D refers to the median absolute deviation. Data % refers to the proportion of data used in the fitting process (i.e., 100% corresponds to a measurement frequency of every day, 50% to every second day, 33% to every 3rd day). Other solutions include all non-true critical points (i.e., saddle and local minima). All parameter estimates rounded to 2.d.p, fitted RSS values to 3.d.p.

D-3 Unique solutions (standard model)

Below are tables of the top 10 highest frequency solutions (to 1.d.p for p^*, τ; 2.d.p for k) found across the searches applying the standard model. The complete set of unique solutions for each scenario are available in spreadsheet form at the following repository link, but to conserve space are not copied in entirety here.

Proportion of fitting data	Total unique solutions (N)	Link (to all solutions)
100%	353	github.com/bsh2/thesis/c5/SF-1.xlsx
50%	275	
33%	275	

100% Fitting Data

Table D-3A: Top 10 highest frequency solutions (standard model, 100% fitting data)

p^*	k_g	τ_{g_1}	k_h	τ_h	Type	Frequency	RSS
100.2	4.6	18.8	5	15.4	saddle	25838	21.42
117	3.8	1.2	5	1.9	minimum	1933	2896.74
94	5	50	4.7	49.7	saddle	164	395.04
93.4	0.34	50	0.01	50	saddle	88	406.15
93.4	0.34	50	0.01	48.7	saddle	77	405.69
93.4	0.34	50	0.01	48.6	saddle	76	405.65
93.4	0.34	50	0.01	48.9	saddle	75	405.76
93.4	0.34	50	0.01	49.9	saddle	64	406.12
93.4	0.34	50	0.01	49.8	saddle	59	406.08
93.4	0.34	50	0.01	48.8	saddle	58	405.72

50% Fitting Data

Table D-3B: Top 10 highest frequency solutions (standard model, 50% fitting data)

p^*	k_g	τ_{g_1}	k_h	τ_h	Type	Frequency	RSS
100.1	4.61	18.9	5.00	15.5	saddle	26359	8.52
116.5	3.65	1.3	5.00	1.9	minimum	1922	1487.52
94.1	5.00	50.0	4.70	49.6	saddle	177	191.12
93.4	0.34	50.0	0.01	50.0	saddle	94	194.69
93.4	0.34	50.0	0.01	48.7	saddle	73	194.41
93.4	0.34	50.0	0.01	49.8	saddle	71	194.65
93.4	0.34	50.0	0.01	48.8	saddle	69	194.43
93.4	0.34	50.0	0.01	48.9	saddle	65	194.45
93.4	0.34	50.0	0.01	48.5	saddle	56	194.37
93.4	0.34	50.0	0.01	49.9	saddle	55	194.67

33% Fitting Data

Table D-3C: Top 10 highest frequency solutions (standard model, 33% fitting data)

p^*	k_g	τ_{g_1}	k_h	τ_h	Type	Frequency	RSS
100.2	4.6	18.8	5	15.4	saddle	14394	6.24
100	0.72	28.5	1.2	8.6	minimum	12798	0.00
117	3.8	1.2	5	1.9	minimum	1834	1018.39
94	5	50	4.7	49.7	saddle	115	150.38
93.4	0.34	50	0.01	50	saddle	48	154.93
117	3.81	1.2	5	1.9	minimum	44	1019.40
93.4	0.34	50	0.01	48.9	saddle	43	154.83
93.4	0.34	50	0.01	48.6	saddle	43	154.81
93.4	0.34	50	0.01	48.7	saddle	41	154.82
93.4	0.34	50	0.01	49.9	saddle	40	154.92

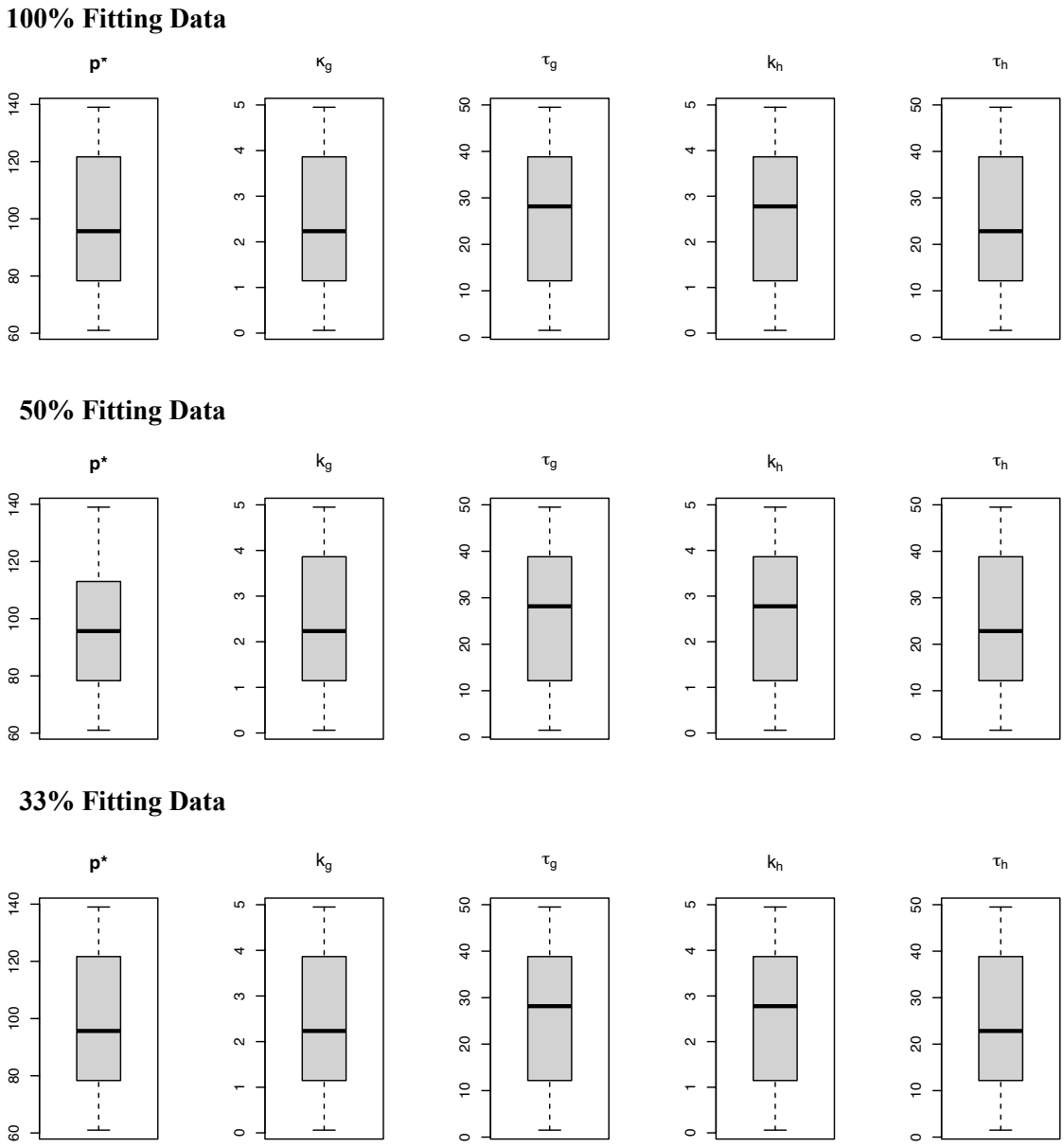


Figure D-3.1: Starting value (initial solution) distributions across iterations that successfully reached the true parameters in the standard model scenarios

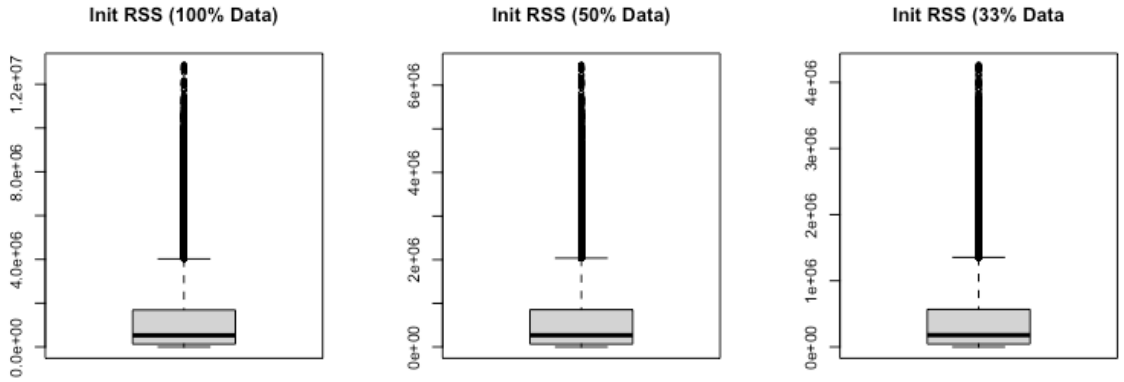


Figure D-3.2: RSS values associated with initial solution distributions across iterations that successfully reached the true parameters in the standard model scenarios.

D-4 Unique solutions (fitness-delay model)

Below are tables of the top 10 highest frequency solutions (to 1.d.p for p^*, τ; 2.d.p for k) found across the searches applying the fitness-delay model. The complete set of unique solutions for each scenario are available in spreadsheet form at the following repository link, but to conserve space are not copied in entirety here.

Proportion of fitting data	Total unique solutions (N)	Link (repository)
100%	383	github.com/bsh2/thesis/c5/SF-1.xlsx
50%	504	
33%	550	

100% Fitting Data

Table D-4A: Top 10 highest frequency solutions (fitness-delay model, 100% fitting data)

p^*	k_g	τ_{g_1}	τ_{g_2}	k_h	τ_h	Type	Frequency	RSS
100	0.74	32.2	10	1.05	4.9	minimum	39401	0.10
100	1.53	28.1	5.8	1.05	21.9	minimum	33266	1.03
100	1.53	28.2	5.8	1.05	21.9	minimum	5811	0.94
100.1	1.64	44	6.2	1.04	49	minimum	4380	2.03
121.2	5	1.2	2.1	1.97	2.1	minimum	1519	4417.29
100	1.52	28.1	5.8	1.05	21.8	minimum	1505	7.01
100	1.52	28.1	5.8	1.05	21.9	minimum	953	15.93
100.1	1.64	44.7	6.1	1.04	50	minimum	940	30.56
100	5	22.4	19.3	1.06	4.1	saddle	621	8.27
100.7	5	19.2	14	0.82	19.2	saddle	613	98.30

50% Fitting Data

Table D-4B: Top 10 highest frequency solutions (fitness-delay model, 50% fitting data)

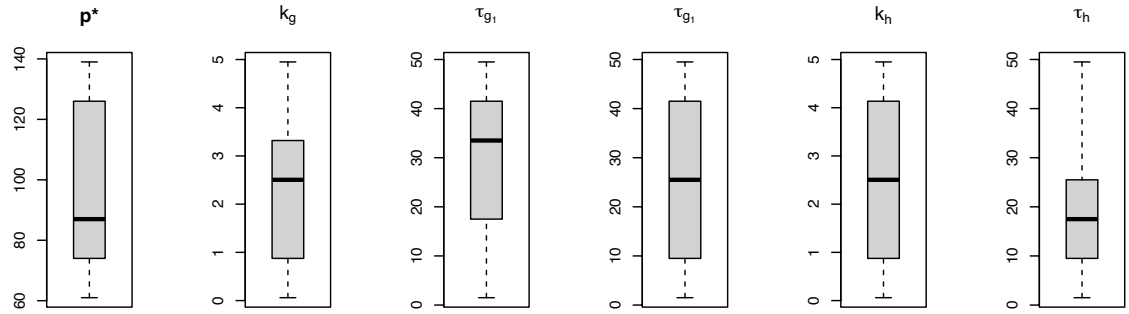
p^*	k_g	τ_{g_1}	τ_{g_2}	k_h	τ_h	Type	Frequency	RSS
100	0.74	32.2	10	1.05	4.9	minimum	34291	0.05
100	1.53	28.1	5.8	1.05	21.9	minimum	31140	0.52
100	0.72	32.5	4.3	1.05	8.6	minimum	8089	0.00
100	1.53	28.2	5.8	1.05	21.9	minimum	5434	0.47
100.1	1.64	44	6.2	1.04	49	minimum	4320	1.02
121.2	5	1.2	2.1	1.97	2.1	minimum	1510	2298.04
100	1.52	28.1	5.8	1.05	21.8	minimum	1411	3.53
100.1	1.64	44.7	6.1	1.04	50	minimum	924	15.31
100	1.52	28.1	5.8	1.05	21.9	minimum	894	8.01
100	5	22.4	19.3	1.06	4.1	saddle	604	4.16

33% Fitting Data

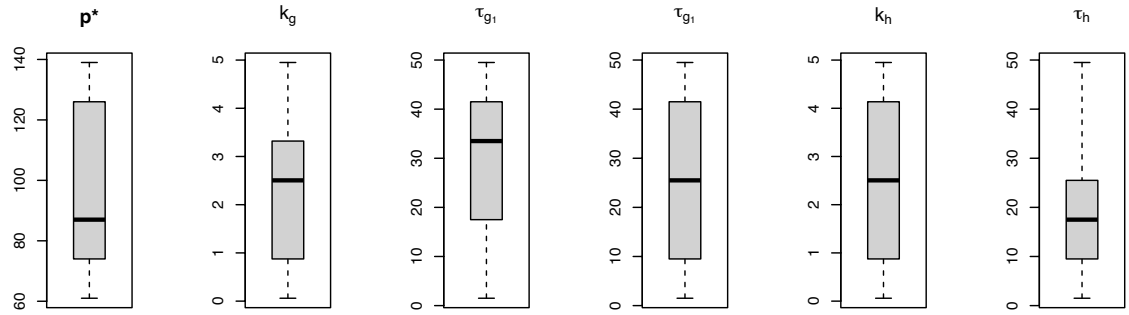
Table D-4C: Top 10 highest frequency solutions (fitness-delay model, 33% fitting data)

p^*	k_g	τ_{g_1}	τ_{g_2}	k_h	τ_h	Type	Frequency	RSS
100	0.74	32.1	10	1.05	4.9	minimum	38922	0.12
100	1.51	28.1	5.8	1.05	21.6	minimum	17384	2.95
100	1.51	28.1	5.8	1.05	21.5	minimum	12086	0.98
100	1.52	28.1	5.8	1.05	21.6	minimum	8634	0.11
100.1	1.64	44.7	6.1	1.05	50	minimum	4099	0.25
119.9	5	1	1.8	1.94	1.8	minimum	1636	1515.64
100	1.51	28	5.8	1.05	21.5	minimum	1332	3.75
100.1	5	19.4	14.4	0.76	19.4	saddle	913	19.93
99.9	5	22.3	19.2	1.03	4.2	saddle	587	2.33
100	1.52	28.1	5.8	1.05	21.7	minimum	409	0.59

100% Fitting Data



50% Fitting Data



33% Fitting Data

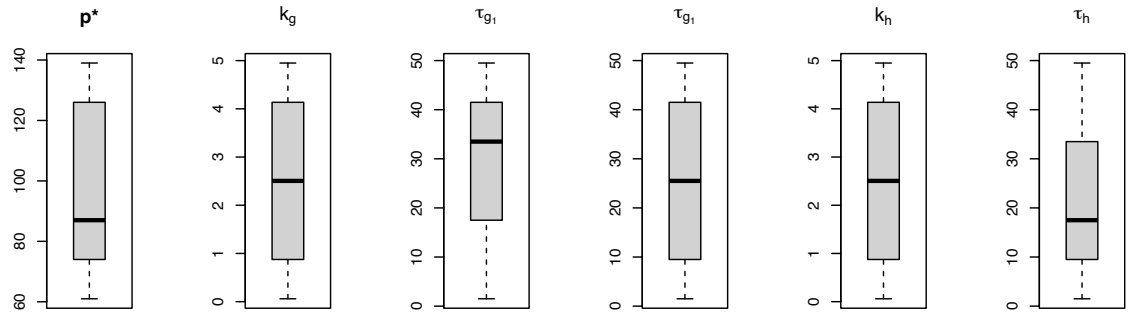


Figure D-4.1: Starting value (initial solution) distributions across iterations that successfully reached the true parameters in the fitness-delay model scenarios

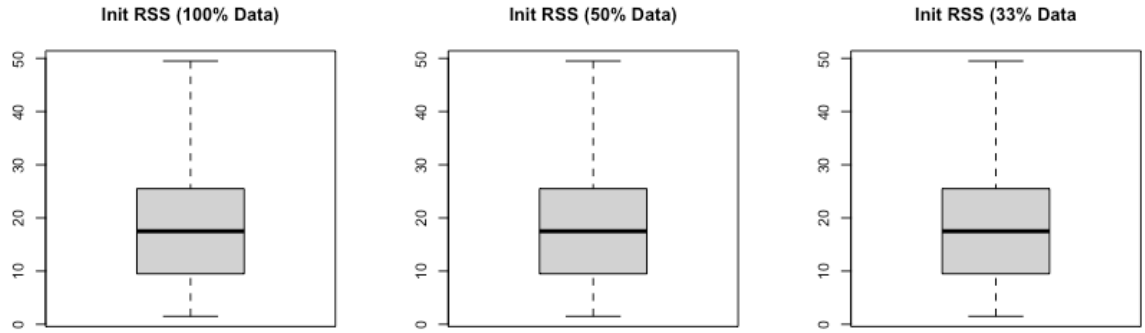


Figure D-4.2: RSS values associated with initial solution distributions across iterations that successfully reached the true parameters in the fitness-delay model scenarios.