A modularization approach for nonlinear model predictive control of distributed fast systems.

DENTLER, J., KANNAN, S., MENDEZ, M.A.O. and VOOS, H.

2016

A modularization approach for nonlinear model predictive control of distributed fast systems *

Jan Dentler, Somasundar Kannan, Miguel Angel Olivares Mendez, Holger Voos[1]

***Abstract*— Distributed interconnected systems are omnipresent today. The development of advanced control methods for such systems are still challenging. Herein, the real-time applicability, flexibility, portability and ease of implementation are issues of the existing control solutions, especially for more advanced methods such as model predictive control. This paper is addressing these issues by presenting an efficient modular composition scheme for distributed fast nonlinear systems. The advantage of this modularization approach is the capability of changing control objectives, constraints, dynamics and system topology online while maintaining fast computation. This work analyzes the functions that have to be provided for a continuation generalized minimal residual method (CGMRES) model predictive controller based on the underlying control problem. The specific structure of these functions allows their decomposition into suitable fast modules. These modules are then used to recompose the functions which are required for the control of distributed systems in a computational efficient way, while maintaining the flexibility to dynamically exchange system parts. To validate this computational efficiency, the computation time of the proposed modular control approach is compared with a standard nonmodular implementation in a pursuit scenario of quadrotor unmanned aerial vehicles (*UAV*). Furthermore the real-time applicability is discussed for the given scenario.**

I. INTRODUCTION

The significance of distributed interacting systems is increasing steadily with the recent demand of interconnected smart devices. This development is present in almost all fields of technology affecting modern society, e.g. transport, energy systems, robotics, etc. which makes the control of such complex systems highly relevant. Distributed systems consist of single entities named agents. Their interaction is described by mutual couplings. This type of systems typically suffers from complexity, but offers advantages such as flexibility, efficiency and security by redundancy. The control of a distributed systems is therefore facing two main issues:

1) Maintaining inherent flexibility of distributed systems: Refers to the flexibility of changing single elements or control objectives at runtime, e.g. if a robot is defect, another robot can overtake its task. If the new robot has different physical dynamics, the control of the whole system has to be readapted to maintain an optimal result.
2) Computational efficiency for real-time applicability: As dynamics and couplings of each agent have to be represented within the control, the computational burden for large sets of agents e.g. robots is high.

A modern approach to control distributed systems is model predictive control (*MPC*), which allows to model couplings as mathematical constraints and to exploit system dynamics to achieve an optimal controlled system behavior. On the other hand *MPC* is computationally expensive, which becomes especially critical for real-time control of fast nonlinear systems and large scale problems. Typical examples of such distributed fast systems are cooperation scenarios in mobile robotics. In general a cooperation scenario describes agents that are cooperating to achieve a common goal, e.g. an unmanned aerial vehicles (UAV) pursuit scenario to reduce costs and weight for expensive localization equipment. The basic idea is: one drone has the task to geo-localize the transport caravan and other drones just follow with the load. Due to its simplicity, this example is used representatively as validation scenario within this work.

To address the problem of computational efficiency, a fast nonlinear model predictive control (*NMPC*) algorithm with constraint handling is required. A literature review on several such *NMPC* approaches is given in [7]. Some of these algorithms are implemented in the comprehensive fast model predictive control framework *ACADO* [4]. Here shall be mentioned the multiple shooting Gauß-Newton approach with approximated Hessian which is interesting for fast and stable computation. *ACADO* features nonlinear constraint handling, various solver interfaces, integrators, a code-generator and provides benchmark examples for fast nonlinear model predictive control e.g. a quadrotor *UAV*. The reason for not adapting this framework for the presented work is its complexity. As the adaptation of the framework requires profound knowledge of the inner structure, it might be interesting for future developments.

Most fast model predictive control algorithms are based on the exploitation of the Hamiltonian-Jacobi-Bellmann functions. One example is the *NMPC* package *GRAMPC* (accessible via [5]) which is based on a gradient descent method. The package offers transparent fast code in *C* and provides the example of a nonlinear quadrotor model as fast nonlinear system benchmark example similar to *ACADO*. One major draw-back of the gradient descent method is the strong dependancy of the computation time from the condition of

*This work was supported by FNR "Fonds national de la Recherche" (Luxembourg) through AFR "Aides à la Formation-Recherche" Ph.D. grant scheme No. 9312118.

[1]Jan Dentler, Dr. Somasundar Kannan, Dr. Miguel Angel Olivares Mendez and Prof. Dr.-Ing Holger Voos is with Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, L-1359 Luxembourg, Luxembourg jan.dentler@uni.lu, somasundar.kannan@uni.lu, miguel.olivaresmendez@uni.lu, holger.voos@uni.lu

the controlled system. Futhermore from an implementational point of view, there was no inequality constraint handling implemented at the time of access.

A *NMPC* algorithm directly related to distributed systems, is proposed by [1]. It presents a finite set predefined feasible control sequence method for multi *UAV* systems. Working with a limited control sequence set, also limits the computational burden. On the other hand the limited set can decrease the optimality of the solution and the set size has to be increased with the nonlinearity of the system.

This leads finally to the continuation generalized minimal residual (*CGMRES*) package by Ohtsuka (accessible via [8]) which offers an *NMPC* with exceptional low computation time. The basic *CGMRES* concept is presented in [8],[12],[13],[14],[15]. It is applicable on fast nonlinear systems and offers nonlinear inequality constraint handling with an auxiliary variable method. More detailed information on this constraint handling method is given in [11]. Besides these advantages, its compact and transparent code allows fast adaptations and debugging, which makes it the choice for this work. To be more specific, the experiments shown in this paper are using a condensed multiple shooting (*CMSCGMRES*) derivative [16],[17],[18] of *CGMRES*, which offers higher numerical stability than the standard *CGMRES* approach.

To address the flexibility issue of distributed system *NMPC*, this paper is presenting a composition scheme that provides the required function for a *CGMRES* solver in a modular way. For this purpose, the required functions for solving a typical optimal control problem (*OCP*) are first presented in II. In section III, these are extended to represent distributed systems. The resulting functions and their mathematical structure are analyzed in section IV and decomposed into single elemental functions. This decomposition exploits the mathematical structure to avoid computationally expensive matrix multiplications. The elemental functions are then provided in a compiled version. Out of these compiled functions, arbitrary optimal control problems can be recomposed at runtime. In standard nonmodular approaches, the system functions are determined fix in compilation time which leads to an efficient computation. The proposed method achieves a similar computational efficiency due to the choice of elemental functions, but furthermore allows runtime changes of system topology, dynamics, couplings and control objectives. The main contribution therefore allows the application of model predictive control on dynamically changing distributed fast nonlinear systems combined with the computational efficiency of compiled fast code. To validate this efficiency, the proposed method is applied in a quadrotor pursuit scenario in section V and compared with a nonmodular implementation. The final conclusion and future work is presented in section VI.

II. Optimal control problem

In terms of control engineering the n_x states x of a plant are controlled by n_u controls u, that are each concatenated to state $\mathbf{x} \in \mathbb{R}^{n_x}$ and control vector $\mathbf{u} \in \mathbb{R}^{n_u}$. The control of a plant with a model predictive controller is typically equivalent to the solving of an optimal control problem (OCP) like

$$\begin{aligned} \min_{\mathbf{u}(\cdot)} \quad & J(\mathbf{u}) = V\left(\mathbf{x}\left(t_f\right), t_f\right) + \int_{t_0}^{t_f} l\left(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau\right) \mathrm{d}\tau \qquad (1) \\ \text{u.c.} \quad & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \tau), \qquad \mathbf{0} \geq \mathbf{c}(\mathbf{x}, \mathbf{u}, \tau) \\ & \mathbf{x}(t_0) = \mathbf{x}_0, \qquad \tau \in \left[t_0, t_f\right], \end{aligned}$$

over a receding horizon. The desired system behavior is defined via a minimization problem of a cost function J and can be specified via terminal (V) and integral (l) cost functions. f is representing the dynamic of the plant expressed by differential equations. The in-/equality constraints of the system are defined via the constraint function c. This optimal control problem can be solved for the interval $\left[t_0, t_f\right]$ under use of boundary values e. g. initial states $\mathbf{x}_0$.

The fast *NPMC* algorithm *CGMRES* [8] is based on the analysis of the Hamiltonian-Jacobi-Bellmann function by using information about the *OCP* optimality conditions. The *CGMRES* package [8] comes with inequality constraint handling via the auxiliary variable transformation [11] which introduces slack variables $\alpha \in \mathbb{R}^{n_\alpha}$ to model the inequality:

$$0 \geq c(\mathbf{x}, \mathbf{u}, \tau) \quad \rightarrow \quad 0 = c(\mathbf{x}, \mathbf{u}, \tau) + \alpha^2, \; \alpha^2 > 0 \; \forall \alpha. \qquad (2)$$

Under use of the package inherent constraint handling (2), the Hamiltonian yields to

$$H(\mathbf{x}, \mathbf{u}, \lambda, t) = l(\mathbf{x}, \mathbf{u}, t) + \lambda^\top \mathbf{f}(\mathbf{x}, \mathbf{u}, t) + \mu^\top \mathbf{c}(\mathbf{x}, \mathbf{u}, \alpha, t) - \kappa^\top \alpha \qquad (3)$$

with the state Lagrange multipliers $\lambda(t) \in \mathbb{R}^{n_x}$ and constraint Lagrange multipliers $\mu(t) \in \mathbb{R}^{n_c}$. The slack variable as well as the constraint Lagrange multipliers are treated as additional optimization variables. The optimization variables are concatenated to vector

$$\mathbf{u}_c(t) = \left[\mathbf{u}^\top(t), \alpha^\top(t), \mu^\top(t)\right]^\top. \qquad (4)$$

This combined control vector is updated by solving the first order optimality condition derived from (3)

$$\begin{aligned} \mathbf{0} \quad & \stackrel{!}{=} \quad H_{u_c}(\mathbf{x}, \mathbf{u}, \lambda, \mu, \alpha, t) \\ & = \quad \begin{bmatrix} \frac{\partial H}{\partial \mathbf{u}} \\ \frac{\partial H}{\partial \alpha} \\ \frac{\partial H}{\partial \mu} \end{bmatrix} = \begin{bmatrix} \frac{\partial l}{\partial \mathbf{u}} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}^\top \lambda + \frac{\partial \mathbf{c}}{\partial \mathbf{u}}^\top \mu \\ \frac{\partial \mathbf{c}}{\partial \alpha}^\top \mu - \kappa \\ \mathbf{c} \end{bmatrix} \begin{matrix} \in \mathbb{R}^{n_u} \\ \in \mathbb{R}^{n_\alpha} \\ \in \mathbb{R}^{n_c} \end{matrix} \qquad (5) \end{aligned}$$

with the *CMSCGMRES* method. The optimality condition states that the descent of a smooth and convex function is zero at its minimum. Besides $H_{u_c}(\mathbf{x}, \mathbf{u}, \lambda, \mu, \alpha, t)$, the system dynamics have to be provided via $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ to be able to predict the system behavior by integrating the system dynamics through time. Finally, also the state derivative of the Hamiltonian

$$H_x(\mathbf{x}, \mathbf{u}, \lambda, \mu, \alpha, t) \quad = \quad \left[\frac{\partial l}{\partial \mathbf{x}} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^\top \lambda + \frac{\partial \mathbf{c}}{\partial \mathbf{x}}^\top \mu\right] \in \mathbb{R}^{n_x} \qquad (6)$$

is provided for updating the state Lagrange multipliers λ.

For executing a *NMPC* based on *OCP* (1), H_u, H_x, the initial states $\mathbf{x}_0$ and an initial guess $\mathbf{u}_c(0)$ have to be be provided to the algorithm. In the following section, we discuss how to exploit the structure of H_u (5) and H_x (6) to efficiently solve cooperation control problems.

III. Optimal control problem for distributed systems

An example of a central control of a distributed system of three coupled quadrotors is given in Fig. 1. In the presented

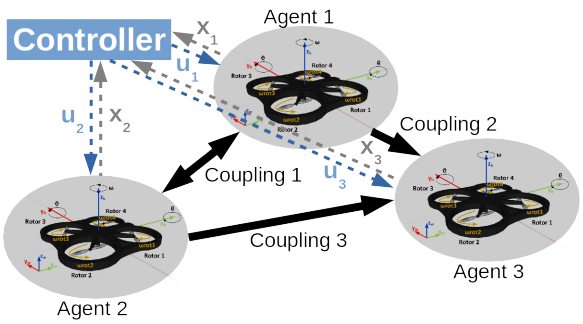

Fig. 1. Scheme of UAV cooperation

cooperation scenario every quadrotor represents an agent with its uncoupled dynamic f_i, constraints c_i and costs l_i. To describe the coupled systems behavior, the single agents are interconnected with coupling constraints c_{ij} and/or coupling costs l_{ij}. Coupling costs are additional cost function terms which are often referred to as a "weakened constraint". The couplings in Fig. 1 do not have to be physical couplings, but can also represent mathematical constraints. The constraints can act on several quadrotors e.g. "Coupling 1" or just one quadrotor. If the arrows in Fig. 1 are considered to represent the direction of information, for the given example "Agent 3" reacts on "Agent 2" but not in reverse.

For a set of agents $i \in \mathscr{V} = \{v_1, ..., v_N\}$, each coupled with neighbours $j \in \mathscr{N}^i = \{v_j \subseteq \mathscr{V} | \{v_i \neq v_j\}\}$ by couplings $\mathscr{E} = \{\varepsilon_{ij} = \{v_i, v_j\} | v_i, v_j \in \mathscr{N}^i\}$, the OCP results to (7).

$$
\begin{aligned}
\min_{\mathbf{u}(\cdot)} \quad & J(\mathbf{u}) = \sum_i V\left(\mathbf{x}_i(t_f), t_f\right) \\
& + \sum_i \int_{t_0}^{t_f} \left[l_i(\mathbf{x}_i(\tau), \mathbf{u}_i(\tau), \tau) \right. \\
& \left. + \sum_j l_{ij}\left(\mathbf{x}_i(\tau), \mathbf{x}_j(\tau), \mathbf{u}_i(\tau), \mathbf{u}_j(\tau), \tau\right) \right] \mathrm{d}\tau \\
\text{u.c.} \quad & \dot{\mathbf{x}}_i = \mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i, \tau) \qquad \mathbf{0} \geq \mathbf{c}_i(\mathbf{x}_i, \mathbf{u}_i, \tau) \\
& \mathbf{x}_i(t_0) = \mathbf{x}_{i,0} \qquad \mathbf{0} \geq \mathbf{c}_{ij}\left(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{u}_j, \tau\right) \\
& \tau \in \left[t_0, t_f\right], v_i \in \mathscr{V}, v_j \in \mathscr{N}^i.
\end{aligned} \tag{7}
$$

The cardinality $|\mathscr{V}| = N$ represents the amount of elements within set $\mathscr{V}$. Respectively $|\mathscr{E}|$ states the amount of Couplings in the system and $\left|\mathscr{N}^i\right|$ the amount of neighbours of Agent i. According to (4) the optimization variables for a distributed system are concatenated to

$$
\begin{aligned}
\mathbf{u}_c(t) = [& \mathbf{u}_1^\top(t), ..., \mathbf{u}_N^\top(t), \\
& \alpha_1^\top(t), ..., \alpha_N^\top(t), \alpha_{N+1}^\top(t), ..., \alpha_{N+|\mathscr{E}|}^\top(t), \\
& \mu_1^\top(t), ..., \mu_N^\top(t), \mu_{N+1}^\top(t), ..., \mu_{N+|\mathscr{E}|}^\top(t) \,]^\top
\end{aligned} \tag{8}
$$

The Hamiltonian (3) for a distributed system corresponds for the given concatenation (8) to

$$
\begin{aligned}
H = & \sum_i^N [\, l_i(\mathbf{x}_i, \mathbf{u}_i, t) + \lambda_i^\top \mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i, t) ... \\
& + \mu_i^\top \mathbf{c}_i(\mathbf{x}_i, \mathbf{u}_i, \alpha_i, t) - \kappa_i^\top \alpha_i \\
& + \sum_j^{|\mathscr{N}^i|} (\, \mu_{ij}^\top \mathbf{c}_{ij}(\, \mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{u}_1, ..., \mathbf{u}_N, \alpha_{ij}, t \,) \\
& - \kappa_{ij}^\top \alpha_{ij} \,) \,].
\end{aligned} \tag{9}
$$

Accordingly the first order optimality condition for the distributed system yields to

$$
H_{u_c} = \begin{bmatrix}
\frac{\partial l_1}{\partial \mathbf{u}_1} + \frac{\partial \mathbf{f}_1}{\partial \mathbf{u}_1}^\top \lambda_1 + \frac{\partial \mathbf{c}_1}{\partial \mathbf{u}_1}^\top \mu_1 ... \\
+ \sum_j^{|\mathscr{N}^i|} \left(\frac{\partial l_{1,j}}{\partial \mathbf{u}_1} + \frac{\partial c_{1,j}}{\partial \mathbf{u}_1}^\top \mu_{1j} \right) \\
\vdots \\
\frac{\partial l_N}{\partial \mathbf{u}_N} + \frac{\partial \mathbf{f}_N}{\partial \mathbf{u}_N}^\top \lambda_N + \frac{\partial \mathbf{c}_N}{\partial \mathbf{u}_N}^\top \mu_N ... \\
+ \sum_j^{|\mathscr{N}^N|} \left(\frac{\partial l_{N,j}}{\partial \mathbf{u}_N} + \frac{\partial c_{N,j}}{\partial \mathbf{u}_N}^\top \mu_{Nj} \right) \\
\frac{\partial \mathbf{c}_1}{\partial \alpha_1}^\top \mu_1 - \kappa_1 \\
\vdots \\
\frac{\partial \mathbf{c}_{N(N-1)}}{\partial \alpha_{N(N-1)}}^\top \mu_{N(N-1)} - \kappa_{N(N-1)} \\
\mathbf{c}_1 \\
\vdots \\
\mathbf{c}_{N(N-1)}
\end{bmatrix} \tag{10}
$$

with the state derivative of the Hamiltonian

$$
H_x = \begin{bmatrix}
\frac{\partial l_1}{\partial \mathbf{x}_1} + \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_1}^\top \lambda_1 + \frac{\partial \mathbf{c}_1}{\partial \mathbf{x}_1}^\top \mu_1 ... \\
+ \sum_j^{|\mathscr{N}^i|} \left(\frac{\partial l_{1,j}}{\partial \mathbf{x}_1} + \frac{\partial c_{1,j}}{\partial \mathbf{x}_1}^\top \mu_{1j} \right) \\
\vdots \\
\frac{\partial l_N}{\partial \mathbf{x}} + \frac{\partial \mathbf{f}_N}{\partial \mathbf{x}_N}^\top \lambda_N + \frac{\partial \mathbf{c}_N}{\partial \mathbf{x}_N}^\top \mu_N ... \\
+ \sum_j^{|\mathscr{N}^N|} \left(\frac{\partial l_{1,j}}{\partial \mathbf{x}_N} + \frac{\partial c_{1,j}}{\partial \mathbf{x}_N}^\top \lambda_{Nj} \right)
\end{bmatrix}. \tag{11}
$$

In the following, the structure of (11) and (10) is exploited for an efficient composition.

IV. Composition strategy for distributed systems

For fast model predictive control, the required equations are preferably given in a fast programming language e.g. *C*. Standard *MPC* control implementations form the central optimal control problem (7) in compilation time and then use the compiled fast functions within runtime. This has the advantage of fast running code, but with the draw-back, that for each adjustment of the system topology, control objective, etc., the functions have to be compiled again. The following composition idea, tries to combine the flexibility of runtime adjustments and computational speed of compiled functions. For this reason, we exploit the *OCP* structure to decompose the system functions into its elemental functions. These are compiled once and can than be recomposed to the desired *OCP* at runtime. The composed *OCP* can than be adapted at runtime according to the system topology and objectives by exchanging the corresponding elemental functions.

Recalling this, (11) and (10) are analyzed in respect to their modularity respective these elemental functions. Both equations are sums of vector-valued functions (e.g. $\frac{\partial l_1}{\partial \mathbf{x}_1}$) and multiplications of matrix-valued functions with vectors (e.g. $\frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_1}^\top \lambda_1$). Each summand is just depending on the corresponding agents data and/or the neighbours data. With this conclusion the computational expensive multiplications of the matrix-valued functions with vectors can be expressed by a vector valued function

$$\text{e.g. } \frac{\partial \mathbf{f}_1(\mathbf{x}_1,\mathbf{u}_1,t)}{\partial \mathbf{x}_1}^\top \lambda_1 \quad \equiv \quad \mathbf{s}(\mathbf{x}_1,\mathbf{u}_1,\lambda_1,t) \tag{12}$$

In reversal conclusion (11) and (10) can be composed by adding up the summands given in Table I. As a result, no matrix multiplication is needed within runtime, as each term in Table I can be expressed by a vector-valued function and the multiplication can therefore be executed before the compilation of the system functions.

TABLE I

COMPOSITION FUNCTIONS

Agent i with $l_i(\mathbf{x}_i,\mathbf{u}_i,\tau)$, $\mathbf{f}_i(\mathbf{x}_i,\mathbf{u}_i,\tau)$, $\mathbf{c}_i(\mathbf{x}_i,\mathbf{u}_i,\tau)$
$\frac{\partial l_i(\mathbf{x}_i,\mathbf{u}_i,\tau)}{\partial \mathbf{x}_i}$, $\frac{\partial l_i(\mathbf{x}_i,\mathbf{u}_i,\tau)}{\partial \mathbf{u}_i}$, $\frac{\partial \mathbf{f}_i(\mathbf{x}_i,\mathbf{u}_i,\tau)}{\partial \mathbf{x}_i}^\top \lambda_i$, $\frac{\partial \mathbf{f}_i(\mathbf{x}_i,\mathbf{u}_i,\tau)}{\partial \mathbf{u}_i}^\top \lambda_i$, $\frac{\partial \mathbf{c}_i(\mathbf{x}_i,\mathbf{u}_i,\tau)}{\partial \mathbf{x}_i}^\top \mu_i$, $\frac{\partial \mathbf{c}_i(\mathbf{x}_i,\mathbf{u}_i,\tau)}{\partial \mathbf{u}_i}^\top \mu_i$, $\frac{\partial \mathbf{c}_i(\mathbf{x}_i,\mathbf{u}_i,\tau)}{\partial \alpha_i}^\top \mu_i$
Coupling ij with $l_{ij}(\mathbf{x}_i,\mathbf{x}_j,\mathbf{u}_i,\mathbf{u}_j,\tau)$, $c_{ij}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_j,\mathbf{u}_j,\tau)$
$\frac{\partial l_{ij}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_j,\mathbf{u}_j,\tau)}{\partial \mathbf{x}_i}$, $\frac{\partial l_{ij}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_j,\mathbf{u}_j,\tau)}{\partial \mathbf{x}_j}$, $\frac{\partial l_{ij}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_j,\mathbf{u}_j,\tau)}{\partial \mathbf{u}_i}$, $\frac{\partial l_{ij}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_j,\mathbf{u}_j,\tau)}{\partial \mathbf{u}_j}$, $\frac{\partial \mathbf{c}_{ij}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_j,\mathbf{u}_j,\tau)}{\partial \mathbf{x}_i}^\top \mu_{ij}$, $\frac{\partial \mathbf{c}_{ij}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_j,\mathbf{u}_j,\tau)}{\partial \mathbf{x}_j}^\top \mu_{ij}$, $\frac{\partial \mathbf{c}_{ij}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_j,\mathbf{u}_j,\tau)}{\partial \mathbf{u}_i}^\top \mu_{ij}$, $\frac{\partial \mathbf{c}_{ij}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_j,\mathbf{u}_j,\tau)}{\partial \mathbf{u}_j}^\top \mu_{ij}$, $\frac{\partial \mathbf{c}_{ij}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{x}_j,\mathbf{u}_j,\tau)}{\partial \alpha_i}^\top \mu_{ij}$,

The idea of the composition is to predefine agents with its system functions, given in the upper part of Table I and couplings with the corresponding functions of the lower part of Table I. Out of these predefined modules the composer is than able to create the corresponding central optimal control problem for any arbitrary topology of the system. In this context also different control objectives can be formulated via different couplings, as the cost function l_{ij} can be adapted accordingly.

In combination with fast functor/pointer access, the computational overhead can be limited to the functor/pointer access times. An example for the access scheme is given in Listing 1. It shows the composition of the system dynamics, concatenated of the dynamics of all agents given by *agentlist*. The index arrays (e.g. *input_indices*) allow fast access to the variables of Agent i (e.g. $u[control\_indices[i]]$) in the concatenated vectors of the global states (e.g. $t, x, u, timevarpar$ in the snippet).

Listing 1. Composition of system dynamics

```
inline void function_f(double* out, double t,
   double* x, double* u, double* timevarpar){
  for(int i=0;i<this->agentlist.size();i++){
    agentlist[i]->function_f(&out[state_indices[i]],t,
        &x[state_indices[i]],
        &u[control_indices[i]],
        &timevarpar[timevarpar_indices[i]]);}
}
```

The code snippet in Listing 1 is also pointing out the modularity of the system. Agent i can be easily exchanged by referring the pointer to agent i in *agentlist* to another agent instance. This is also valid for control objectives, constraints, etc. As an example, the modularization is explained with a change of control objectives. Consider two different couplings have been predefined:

- follow drone with fix distance
- hover at position

Each of these objectives has a representation by a set of the functions given in Table I. An array of coupling pointers is storing the couplings used for the composition of the central optimal control problem. First the "hover at position" coupling is applied on a quadrotor by adding the pointer to the "hover at position" coupling to the list of couplings.

Afterwards the "follow drone with fix distance" objective shall be applied. Accordingly, the previous pointer to "hover at position" is substituted by a pointer to the "follow drone with fix distance" element. An advantage of the proposed modularization is, that without equation simplifications the composed functions are mathematically identical to the nonmodular formulation. Therefore the controllability and stability of the controlled system is not affected and directly determined by system and solver properties, if the additional calculation overhead can be neglected. In the following section the computational efficiency of the modular composition method is validated by a quadrotor *UAV* cooperation scenario.

V. EXPERIMENTAL VALIDATION

For ease of comprehension, the composition efficiency is validated with a simple persuit scenario of two quadrotors. The quadrotors are implemented as 3D-models in the simulation environment *V-REP*. According to (7) the scenario is described by the OCP

$$\begin{aligned}
\min_{\mathbf{u}_1,\mathbf{u}_2} J = & \int_{t_0}^{t_f} (\mathbf{x}_1^* - \mathbf{x}_1)^\top \mathbf{Q}_1 (\mathbf{x}_1^* - \mathbf{x}_1) + \mathbf{u}_1^\top \mathbf{R}_1 \mathbf{u}_1 \\
& + \int_{t_0}^{t_f} (\mathbf{x}_2^* - \mathbf{x}_2)^\top \mathbf{Q}_2 (\mathbf{x}_2^* - \mathbf{x}_2) + \mathbf{u}_2^\top \mathbf{R}_2 \mathbf{u}_2 \\
& + q_d \sqrt{(x_1 - x_2)^\top (x_1 - x_2) - d_{des}} \; \mathrm{d}\tau
\end{aligned} \tag{13}$$

$$\text{u.c.} \quad i \in [1,2]$$

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{z}_i \\ \dot{\psi}_i \\ \dot{v}_{f,i} \\ \dot{v}_{s,i} \end{bmatrix} = \begin{bmatrix} \dot{v}_{f,i}\cos(\psi_i) - \dot{v}_{s,i}\sin(\psi_i) \\ \dot{v}_{f,i}\sin(\psi_i) + \dot{v}_{s,i}\cos(\psi_i) \\ 0.8827 \cdot u_{z,i} \\ -0.005879 \cdot \psi_i + 1.265 \cdot u_{\psi,i} \\ -0.8799 \cdot v_{f,i} + 3.273 \cdot u_{f,i} \\ -0.5092 \cdot v_{s,i} - 1.458 \cdot u_{s,i} \end{bmatrix} \tag{14}$$

$$c: \quad -1 < u < 1 \tag{15}$$

$$\mathbf{x}_1(0) = [0,0,0,0,0,0], \qquad \mathbf{x}_2(0) = [0,1,0,0,0,0] \tag{16}$$

$$\mathbf{Q}_1 = [1,1,2,1,0,0], \qquad \mathbf{Q}_2 = [0,0,2,1,0,0] \tag{17}$$

$$\mathbf{R}_1 = [10,10,10,10], \qquad \mathbf{R}_2 = [10,10,10,10] \tag{18}$$

$$q_d = 10,\ d_{des} = 1, \qquad \mu_i(0) = [1,1,1,1] \cdot 10^{-4} \tag{19}$$

$$\alpha_i(0) = [0.9,0.9,0.9,0.9], \quad \kappa_i(0) = [1,1,1,1] \cdot 10^{-4} \tag{20}$$

$$\zeta_C = 1,\ \alpha_C = 1,\ \Delta t = 0.1, \tau_f = 1, \tag{21}$$

$$n_{hor} = 10,\ h = 0.001,\ \varepsilon = 10^{-8},\ it_{max} = 30 \tag{22}$$

with the position $\vec{x} = [x, y, z]$ the yaw angle ψ and the linear forward $\dot{v}_f$, respective sideward velocity $\dot{v}_s$. For ease of notation the time dependancy of variables and funtions are not explicitly shown in (13)-(15). The control objectiv (13) of UAV_1 is to track a given target state $\mathbf{x}_1^*$ (24) by penalizing the state error with $\mathbf{Q}_1$, whereas $\mathbf{Q}_2$ is used for tracking the z axis of UAV_2, to force it to stay in the xy-plane. The control penalties $\mathbf{R}_1, \mathbf{R}_2$ reduce the control action $\mathbf{u}_i = \left[\dot{v}_{f,i}, \dot{v}_{s,i}, u_{z,i}, u_{\psi,i}\right]$ and damp the system. The actual cooperation is introduced by the distance penalty term in (13). As both UAVs try to keep the same distance and UAV_1 is additionally tracking a target position, UAV_2 tries to follow UAV_1 with the given distance of $d_{des} = 1$. Here the advantage of the modular composition scheme can be seen. According to the composition algorithm, the only function that has to be provided for the coupling is

$$\frac{\partial l_{21}}{\partial \mathbf{x_1}} = -\frac{\partial l_{21}}{\partial \mathbf{x_2}} = \frac{(\vec{x}_1 - \vec{x}_2)\, q_d}{2\sqrt{(\vec{x}_1 - \vec{x}_2)^\top (\vec{x}_1 - \vec{x}_2) - d}}. \tag{23}$$

which can be used for $UAV1$ and in reverse for $UAV2$. As most of the desired constraints are either dependant on controls or states, most composition functions (Table I) are equal to zero and do not have to be considered. These can be dynamically excluded from the execution which saves computation time.

In the scenario both UAVs are constrained by their dynamics (14). Additionally the control limitation (15) is realized via the auxiliary variable method (2). The corresponding initialization of the Lagrange multipliers μ, the slack variables α and the slack penalty κ is given in (19)-(20). The system is initialized in an optimal state (16), where $d_{des} = 1$ is fulfilled. (17) is showing the state penalty and (18) respectively the control penalty. q_d in (18) is the penalty of the "keep fix distance" cost term, whereas d_{des} is representing the target distance between the UAVs. *CMSCGMRES* related parameters are given in (21)-(22) by continuation convergence ζ_C, horizon adaptation factor α_C, the simulation and control update timestep Δt, the horizon time τ_f with the horizon discretization n_{hor}, the forward difference step h, the precision ε and the maximal amount of iterations it_{max}.

To analyze the systems behavior, a step signal is applied on reference $\mathbf{x}_1^*$ of UAV_1 at time $t \approx 20\,\mathrm{s}$

$$\mathbf{x}_1^*(0) = [0,0,0,0,0,0]\,,\ \mathbf{x}_1^*(\approx 20) = [1,0,0,0,0,0]\,. \tag{24}$$

which is illustrated by Fig. 2. The figure shows the comparison of the modular composition technique (left hand side) with a nonmodular implementation (right hand side). Nonmodular implementation refers to providing the system functions for the complete system in a closed, compiled form directly to the Solver. In the plots, the z-axis is not shown for ease of visualization, as both UAVs show insignificant deviation from the tracked $z = 0$. The XY-Plane inserts show the system behavior in bird's eye view, where the UAVs are drawn as circles and their distance is shown as straight line. As expected UAV_1 is reaching the desired position, marked with a square. Both UAVs try to keep a constant distance and therefore UAV_2 is following UAV_1. The comparison between the plots of both approaches confirm identical behavior. This is expected, as the composed system functions and the nonmodular system functions are mathematically identical. Small derivations in the figures are caused by the simulator, the disturbance of the controller computation by other system tasks and because the trajectory change is applied manually and therefore not exactly triggered at $20s$.

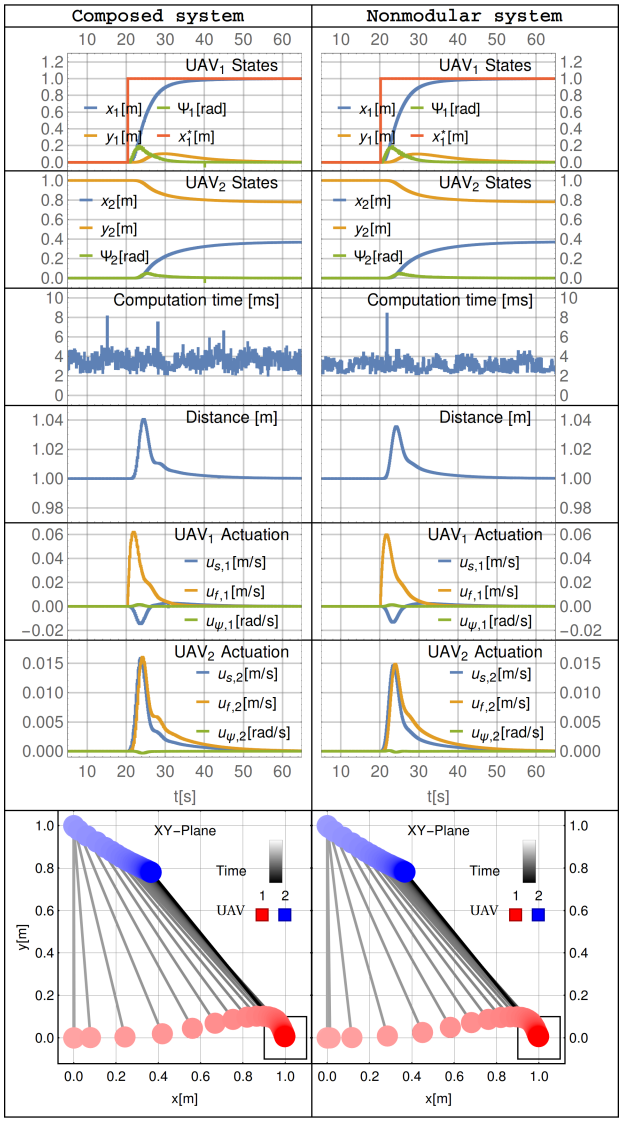

Fig. 2. Data of composed and nonmodular pursuit scenario: UAV_1 and UAV_2 keep distance $d = 1$ and UAV_1 tracking target

The slightly delayed reaction of UAV_2 in the actuation plots is caused by the trade-off between energy optimality (control penalty) and fulfilling the coupling constraint. This can also be seen by the small deviation in the distance (considering the scale of the image). To assess the computational efficiency of the proposed modularization, the computation time plots of Fig. 2 are compared. Both plots show peak values of $\approx$ 8ms which are assumed to be produced by external influences such as interruptions of the simulator or controller by other system processes, as they appear arbitrarily and without correlation to the system trajectory. Globally the

computation time does not exceed $t_{comp} = 10ms$ which validates the real-time applicability for the given control update frequency of $1/\Delta t = 10Hz$ and shows the potential of the proposed composition in combination with the chosen solver. In reverse conclusion this setup would allow the real-time control of up to 10 similar scenarios, controlled by a single computer. The mean computation time of the control with modular composition is $\bar{t}_{comp} = 3.551$ms, which represents a computational overhead of 16% to the $\bar{t}_{comp} = 3.057$ms of the nonmodular *OCP*. Here shall be stressed that the advantage of the composition scheme is, that the composed system can be adapted without recompilation at runtime, while the nonmodular system functions have to be recompiled for any changes in the dynamics, objectives or topology. Due to the exploitation of the *OCP* structure (choice of elemental functions) to avoid matrix multiplications in the composition, the computational overhead is remarkably low as 16% for this scenario. For more complex agents this ratio would further decline as the number of functor/pointer access stays the same, but the effective time within the functions would be higher. This low computational overhead confirms the computational efficiency of the proposed technique in connection with the *CMSCGMRES* method.

VI. Conclusion and future work

This work presents a modularization technique that targets efficient modular real-time control of distributed fast nonlinear systems. It shows how the required functions for solving an *OCP* of distributed systems with *CGMRES* can be mathematically decomposed into elemental functions. To control a distributed system, these elemental functions (Table I) are defined for each type of system entity (agent) and interconnection (coupling) and compiled once. Out of these compiled functions, the *OCP* for any arbitrary scenario based on the defined agents and couplings can be composed at runtime. This procedure in combination with the proposed composition has two major advantages:

- An *NMPC* control can by applied that dynamically adapts to a change in the distributed system topology, couplings, dynamics and control objectives under perpetuation of the low level language *C* performance without recompilation.
- The mathematical decomposition demonstrates, how runtime matrix multiplications in the composition of the Hamiltonian and its derivatives can be avoided by defining functions that directly contain these multiplications. This speeds up the computation to achieve a similar performance as solving a nonmodular *OCP*.

The computational efficiency of the composition approach is validated with the presented cooperation scenario of a drone pursuit scenario (section V). The comparison with a direct nonmodular implementation of the *OCP* shows, that the full modular composition requires a small computational overhead of 16%. In combination with the *CMSCGMRES OCP* solver, the real-time applicability could be confirmed for the given scenario.

The presented work represents the first stage of solving optimal control problems of distributed systems. Future work will address an adaptation of the composition scheme to distributed model predictive control algorithms. Further investigation will also address the system behavior under runtime switching of control objectives. Implementation of other solvers e.g. *ACADO* are also an interesting future field of studies.

References

[1] S. Bertrand, J. Marzat, H. Piet-Lahanier, A. Kahn, Y. Rochefort, MPC Strategies for Cooperative Guidance of Autonomous Vehicles, in: Aerospace Lab Journal 2014, Issue 8, pp. 1-18.

[2] R.V. Lopes and P. Santana, Model Predictive Control applied to tracking and attitude stabilization of a VTOL quadrotor aircraft, in 21st International Congress of Mechanical Engineering 2011, Brazil, pp. 176-185

[3] P. Bouffard, Thesis: On-board Model Predictive Control of a Quadrotor Helicopter: Design, Implementation, and Experiments EECS Department, University of California, Berkeley, 2012,December 13

[4] D. Ariens and M. Diehl and H. J. Ferreau and B. Houska and F. Logist and R. Quirynen and M. Vukov, ACADO Toolkit User's Manual, 2015 `http://acado.github.io`, Accessed: 2015-09-11

[5] K. Graichen and T. Utz, GRAMPC documentation, 2014, `https://www.uni-ulm.de/in/mrm/forschung/regelung-und-optimierung/grampc.html`, Accessed: 2015-04-20

[6] K. Graichen, Lecture notes: Methoden der Optimierung und optimalen Steuerung, institution Ulm University, 2013, pp. 84

[7] M. Diehl and H. Ferreau and N. Haverbeke, Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation, in Nonlinear Model Predictive Control, Magni, Lalo and Raimondo, DavideMartino and Allgöwer, Frank, volume 384, 2009, isbn 978-3-642-01093-4 pp. 391-417

[8] T. Ohtsuka, symlab: cgmres source code, `http://www.symlab.sys.i.kyoto-u.ac.jp/~ohtsuka/code/index.htm`, Accessed: 2015-09-4

[9] G. Antonelli and K. Baizid and F. Caccavale and G. Giglio and G. Muscio and F. Pierri, Control software architecture, Unmanned aerial vehicle manipulator systems, multi-robot systems, coordination, in Journal of Software Engineering for Robotics, 2014,

[10] Xiaotao Liu, Robust Model Predictive Control and Distributed Model Predictive Control Feasibility and Stability, Department of Mechanical Engineering, University of Victoria, 2014

[11] M. Huang and H. Nakada and Butts and R. Kenneth and I. Kolmanovsky, Nonlinear Model Predictive Control of a Diesel Engine Air Path: A Comparison of Constraint Handling and Computational Strategies, Proceedings - 5th IFAC Conference on Nonlinear Model Predictive Control - Seville, Spain, 2015

[12] T. Ohtsuka, A continuation/GMRES method for fast computation of nonlinear receding horizon control, in Automatica, volume 40, 2004, pp. 563-574

[13] H. Seguchi and T. Ohtsuka, Nonlinear receding horizon control of an RC hovercraft, in Proceedings of the International Conference on Control Applications, volume 2, 2002, pp. 1076-1081

[14] H. Seguchi and T. Ohtsuka, Nonlinear receding horizon control of an underactuated hovercraft, in International journal of robust and nonlinear control, volume 13, 2003, pp. 381-398

[15] Y. Soneda and T. Ohtsuka, Nonlinear moving horizon state estimation for a hovercraft with continuation/GMRES method, in Control Applications, Proceedings of the 2002 International Conference on, volume 2, 2002, pp. 1088-1093

[16] Y. Shimizu and T. Ohtsuka and M. Diehl, Nonlinear receding horizon control of an underactuated hovercraft with a multiple-shooting-based algorithm, in Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006

[17] Y. Shimizu and T. Ohtsuka and M. Diehl, A real-time algorithm for nonlinear receding horizon control using multiple shooting and continuation/Krylov method, in International Journal of Robust and Nonlinear Control, volume 19, 209, pp. 919-936

[18] Y. Shimizu, T. and Ohtsuka, A real-time algorithm for nonlinear receding horizon control of descriptor systems in Proceedings of SICE Annual Conference 2010, 2010, pp. 219-222