Noname manuscript No.
(will be inserted by the editor)

A dual Newton strategy for scenario decomposition in robust multi-stage MPC

D. Kouzoupis · E. Klintberg · M. Diehl · S. Gros

Received: date / Accepted: date

Abstract This paper considers the solution of tree-structured Quadratic Programs (QPs) as they may arise in multi-stage Model Predictive Control (MPC). In this context, sampling the uncertainty on prescribed decision points gives rise to different scenarios that are linked to each other via the so-called non-anticipativity constraints. Previous work suggests to dualize these constraints and apply Newton's method on the dual problem in order to achieve a parallelizable scheme. However, it has been observed that the globalization strategy in such an approach can be expensive. To alleviate this problem, we propose to dualize both the non-anticipativity constraints and the dynamics to obtain a computationally cheap globalization. The dual Newton system is then reformulated into small, highly structured linear systems that can be solved in parallel to a large extent. The algorithm is complemented by an open-source software implementation that targets embedded optimal control applications.

Keywords Robust control · Multi-stage MPC · Quadratic Programming · Dual decomposition

1 Introduction

Model Predictive Control (MPC) has become the preferred control technique in a growing set of applications due to the natural way in which constraints can be incorporated in the control policy. However, constraint satisfaction

E. Klintberg
E-mail: kemil@chalmers.se / emil.klintberg@qamcom.se

D. Kouzoupis
E-mail: dimitris.kouzoupis@imtek.uni-freiburg.de

M. Diehl
E-mail: moritz.diehl@imtek.uni-freiburg.de

S. Gros
E-mail: grosse@chalmers.se

can in general not be guaranteed if uncertainties are present in the system. Several methods have been proposed to handle uncertainties in the employed model, e.g., min-max MPC [10] and tube-based MPC [30], [29]. Another common formulation, often called multi-stage MPC or stochastic programming, represents the uncertainty via a finite number of realizations at each decision point [33], [4], [25]. This formulation posses attractive properties such as recursive feasibility [27] and increased feasibility [26]. However, a major drawback is the size of the underlying optimization problem, which grows exponentially with the length of the control horizon.

Computational aspects of robust MPC have been considered e.g. in [18], where a primal-dual interior point method is proposed for min-max MPC, and in [37], where real-time feasibility is achieved via a tube-based robust MPC formulation together with the early termination of an interior point method. Moreover, several optimization methods have been proposed that are applicable to multi-stage MPC problems. The authors of [7], [19], [34], [32], [21] propose to use tailored interior point methods, the authors of [24], [20] use parallelizable active set methods, whereas the authors of [6], [31], [28] use various decomposition techniques in order to exploit the intrinsic structure of the problem.

The active set methods in [24], [20] are particularly well suited for multi-stage MPC, due to the natural way in which the similarity between subsequent Quadratic Programs (QPs) can be exploited. In particular, if the active set does not change between subsequent instances of the multi-stage MPC problem, the methods converge in one step. However, these methods suffer from two drawbacks stemming from the non-smoothness of dual problem. First, there is no practically useful upper bound on the number of iterations needed to solve a problem. This is a limitation in safety critical applications, although methods of this kind usually work well in practice [13], [14]. Secondly, globalization can be expensive since many backtracking steps may be required at each iteration to enforce convergence. This can be a significant drawback since every step involves solving QPs which often become the computational bottleneck of the method. In this paper, we overcome the second difficulty for the practically important class of multi-stage MPC problems with diagonal cost and simple bounds.

In contrast to the methods proposed in [24], [20], we dualize both the non-anticipativity constraints and the dynamic constraints. As a result, the solution of the the QPs becomes computationally negligible, whereas the resulting dual Newton system becomes larger. Moving computational effort from the solution of the QPs to the solution of the Newton system is an improvement since the Newton system is only solved once every iteration whereas the QPs may be solved multiple times. In order to further enhance performance, the Newton system is reformulated into several small, highly structured linear systems that can be solved in parallel to a large extent. For the two-stage MPC case, the structure of the dual Hessian is similar to the KKT matrix of the interior point method presented in [21], where the same reformulation is proposed. In

this paper, we generalize the parallelizable solution of the linear system to the multi-stage MPC case and use it within the context of a dual Newton method.

The existence of a distributed solution of the Newton system can be interpreted using the theory of chordal graphs [8], [35]. More precisely, the block sparsity pattern of the dual Hessian turns out to be chordal (being a block arrowhead matrix) and this implies that it has a Cholesky factorization with no fill-in. This property is crucial for many parallelizable factorization algorithms. Chordal sparsity patterns have lately attracted a lot of attention in the optimization community and several methods have been proposed that can solve such problems efficiently, see, e.g., [32].

To assess the performance of the proposed scheme, an open-source software implementation has been developed as part of the tree-sparse Quadratic Programming toolbox `treeQP` [3]. The focus of the implementation is on embedded optimization applications where small to medium scale problems are typically solved at high sampling frequencies. In our numerical experiments, the developed scheme is benchmarked against the interior point method of [16], which is to the best of our knowledge the only freely available software that targets multi-stage MPC problems of the same size range. The latter algorithm works directly on the tree-sparse QPs and resembles the methods proposed in [34], [32]. In addition, it uses the same linear algebra library, `BLASFEO` [17], which makes the comparison between the two algorithms more transparent.

The paper is organized as follows. In Section 2, we recall multi-stage MPC and the non-smooth dual Newton strategy. In Section 3, we detail the parallel calculation of the dual Hessian and gradient, whereas in Section 4, we propose a reformulation of the Newton system to exploit the problem structure and enhance parallel computations. The resulting method together with some further algorithmic aspects is presented in Section 5 and the software implementation in Section 6. In Section 7, the performance of the algorithm is assessed via numerical experiments. Section 8 concludes the paper.

2 Preliminaries

In this section, we recall multi-stage MPC and Newton strategies in the context of dual decomposition.

2.1 Multi-stage MPC

As a result of imperfect models and uncertain disturbances, constraint satisfaction can in general not be guaranteed for MPC schemes. A common remedy, often denoted as multi-stage MPC or stochastic programming, is to discretize the underlying stochastic process and describe the evolution of the uncertainty via a scenario tree [33]. To that end, we consider a discrete-time, constrained

system with uncertain parameters θ:

$$x_{i+1} = A_i(\theta)x_i + B_i(\theta)u_i \tag{1a}$$
$$x_{\min} \leq x_i \leq x_{\max} \tag{1b}$$
$$u_{\min} \leq u_i \leq u_{\max}, \tag{1c}$$

where $x_i \in \mathbb{R}^n$ and $u_i \in \mathbb{R}^m$ denote the state and control variables respectively. To account for the uncertain parameters, we consider m_d realizations of (1) at each time stage. The evolution of the system can then be described by a scenario tree as depicted in Figure 1.

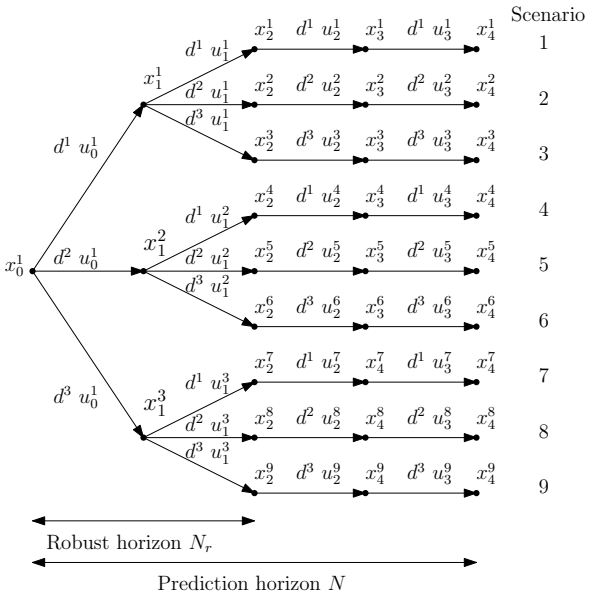

Fig. 1: The evolution of the system represented as a scenario tree. For nodes and branches that are shared between multiple scenarios, the variable corresponding to the scenario with the lowest index is visualized.

We define a scenario as a path from the root node to a leaf node of the scenario tree. The number of scenarios is thus growing exponentially with the length of the MPC horizon, yielding very large optimization problems. It is therefore often proposed to treat the uncertain parameters as constant after a certain period of time. We denote the time period where the parameters can change as the robust horizon N_r, in contrast to the prediction horizon N. Accordingly, we consider $M = m_d^{N_r}$ scenarios.

To enhance parallel computations, we introduce separate state and control variables for each scenario, i.e. we introduce $x_k = [x_{k,1}^\top \cdots x_{k,N}^\top]^\top \in \mathbb{R}^{\bar{n}}$, with $x_{k,i} \in \mathbb{R}^n$, and $u_k = [u_{k,0}^\top \cdots u_{k,N-1}^\top]^\top \in \mathbb{R}^{\bar{m}}$, with $u_{k,i} \in \mathbb{R}^m$ for $k = 1, \ldots, M$. However, because the uncertainty cannot be anticipated, control actions are

restricted to only depend on historical realizations of the uncertainty, such that the control variables of the scenarios are coupled at their shared nodes. More specifically, if the uncertainty realizations for scenario k and l are identical up to and including time stage i, their control inputs should be identical up to that time stage, i.e. $u_{k,j} = u_{l,j},\ \forall j = 0, \cdots i$. This restriction is commonly denoted as *non-anticipativity constraints*. The resulting MPC problem can be formulated as:

$$\min_{x,u} \quad \sum_{k=1}^{M} V_k(x_k, u_k) \tag{2a}$$

$$\text{s.t.} \quad \sum_{k=1}^{M} \bar{C}_k u_k = 0 \tag{2b}$$

$$\bar{A}_k x_k + \bar{B}_k u_k = b_k, \quad k = 1, \ldots, M \tag{2c}$$

$$\underline{x}_k \leq x_k \leq \overline{x}_k, \quad k = 1, \ldots, M \tag{2d}$$

$$\underline{u}_k \leq u_k \leq \overline{u}_k, \quad k = 1, \ldots, M, \tag{2e}$$

where we have introduced the notations $x = [x_1^\top \ldots x_M^\top]^\top$ and $u = [u_1^\top \ldots u_M^\top]^\top$ for the collection of variables over the scenarios. Additionally, we have defined $V_k(x_k, u_k) = \frac{1}{2} x_k^\top \bar{Q}_k x_k + \frac{1}{2} u_k^\top \bar{R}_k u_k + \bar{q}_k^\top x_k + \bar{r}_k^\top u_k$, $b_k = [-\bar{x}^\top A_{k,0}^\top \, 0 \ldots 0]^\top$, where $\bar{x}$ denotes the initial state estimate, and:

$$\bar{Q}_k = \begin{bmatrix} Q_{k,1} & & \\ & \ddots & \\ & & Q_{k,N} \end{bmatrix}, \ \bar{R}_k = \begin{bmatrix} R_{k,0} & & \\ & \ddots & \\ & & R_{k,N-1} \end{bmatrix} \tag{3a}$$

$$\bar{A}_k = \begin{bmatrix} -I & & & \\ A_{k,1} & -I & & \\ & \ddots & \ddots & \\ & & A_{k,N-1} & -I \end{bmatrix}, \ \bar{B}_k = \begin{bmatrix} B_{k,0} & & \\ & \ddots & \\ & & B_{k,N-1} \end{bmatrix}, \tag{3b}$$

where we require the matrices $\bar{Q}_k \in \mathbb{S}_{++}^{nN}$ and $\bar{R}_k \in \mathbb{S}_{++}^{mN}$ to be diagonal.

The formulation of the non-anticipativity constraints (2b) provides some freedom in constructing the sparsity structure of $\bar{C}$ via the ordering of the constraints. The structure of $\bar{C}$ has a direct impact on the sparsity structures of the dual Hessian, and ought to be exploited in order to facilitate the solution of (2). Due to the fact that the non-anticipativity constraints are enforced at nodes in the scenario tree, and that neighboring scenarios share nodes, we propose to enforce the non-anticipativity constraints in a chain structure. To that end we introduce the notation:

$$p = m \sum_{k=1}^{M-1} n_{c,(k,k+1)}, \tag{4}$$

where $n_{c,(k,k+1)}$ denotes the number of common nodes in the scenario tree for scenario k and $k+1$, and select the matrices $\bar{C}_k \in \mathbb{R}^{p\times \bar{m}}$ as follows:

$$\begin{aligned}\bar{C} &= \left[\begin{array}{c|c|c|c|c} C_{1,2} & -C_{1,2} & & & \\ & C_{2,3} & -C_{2,3} & & \\ & & \ddots & \ddots & \\ & & & C_{M-1,M} & -C_{M-1,M} \end{array}\right] = \\ &= \left[\, \bar{C}_1 \; \bar{C}_2 \; \dots \; \bar{C}_M \,\right], \end{aligned} \tag{5}$$

where we have introduced:

$$C_{k,k+1} = \left[\begin{array}{ccc|c} I_m & & & 0 \cdots 0 \\ & \ddots & & \vdots \;\ddots\; \vdots \\ & & I_m & 0 \cdots 0 \end{array}\right] \in \mathbb{R}^{mn_{c,(k,k+1)}\times \bar{m}}, \tag{6}$$

i.e., each block row of $C_{k,k+1}$ corresponds to a common node in the scenario tree for scenario k and scenario $k+1$.

2.2 Dual decomposition

We introduce the dual variables $\lambda \in \mathbb{R}^{m(M-1)}$ corresponding to the non-anticipativity constraints (2b), and the dual variables $\mu_k \in \mathbb{R}^{n(N-1)}$ corresponding to the dynamics (2c), and define the partial Lagrange function:

$$\mathcal{L}(x,u,\mu,\lambda) = \sum_{k=1}^{M} V_k(x_k,u_k) + \lambda^\top \sum_{k=1}^{M} \bar{C}_k u_k + \sum_{k=1}^{M} \mu_k^\top \left(\bar{A}_k x_k + \bar{B}_k u_k - b_k\right), \tag{7}$$

where we have defined $\mu = [\mu_1^\top \dots \mu_M^\top]^\top$ for notational convenience. Observe that the Lagrange function is seperable in the primal variables x and u and in the dual variables μ, i.e. we can express $\mathcal{L}(x,u,\lambda)$ as:

$$\mathcal{L}(x,u,\mu,\lambda) = \sum_{k=1}^{M} \mathcal{L}_k(x_k,u_k,\mu_k,\lambda), \tag{8}$$

where we have introduced:

$$\mathcal{L}_k(x_k,u_k,\mu_k,\lambda) = V_k(x_k,u_k) + \lambda^\top \bar{C}_k u_k + \mu_k^\top \left(\bar{A}_k x_k + \bar{B}_k u_k - b_k\right). \tag{9}$$

Due to the decomposable structure of $\mathcal{L}(x,u,\mu,\lambda)$, the Lagrange dual function $d(\lambda) = -\min_{(x,u)\in\mathcal{Z}} \mathcal{L}(x,u,\mu,\lambda)$, can be evaluated in parallel as:

$$d(\mu,\lambda) = -\sum_{k=1}^{M} \min_{(x_k,u_k)\in\mathcal{Z}_k} \mathcal{L}_k(x_k,u_k,\mu_k,\lambda), \tag{10}$$

where we have introduced the feasible sets $\mathcal{Z}_k = \{(x_k,u_k) : \underline{x}_k \le x_k \le \overline{x}_k,\ \underline{u}_k \le u_k \le \overline{u}_k\}$, and $\mathcal{Z} = \mathcal{Z}_1 \times \dots \times \mathcal{Z}_M$.

Due to strict convexity of (2), $d(\mu, \lambda)$ is convex and continuously differentiable. The Hessian of $d(\mu, \lambda)$ is a piecewise constant matrix and changes with the active set [14]. The non-smooth dual problem is then given by:

$$\min_{\mu,\lambda} d(\mu, \lambda), \tag{11}$$

from which the primal solution x^* and u^* to (2) can be recovered due to strong duality [9].

The gradient of the dual function is given by [5]:

$$\nabla d(\mu, \lambda) = \begin{bmatrix} r_1(x_1^*(\mu_1, \lambda), u_1^*(\mu_1, \lambda)) \\ \vdots \\ r_M(x_M^*(\mu_M, \lambda), u_M^*(\mu_1, \lambda)) \\ r(u^*(\mu, \lambda)) \end{bmatrix}, \tag{12}$$

where $x_k^*(\mu_k, \lambda)$ and $u_k^*(\mu_k, \lambda)$ are solutions to the subproblems:

$$\begin{bmatrix} x_k^*(\mu_k, \lambda) \\ u_k^*(\mu_k, \lambda) \end{bmatrix} = \arg \min_{(x_k, u_k) \in \mathcal{Z}_k} \mathcal{L}_k(x_k, u_k, \mu_k, \lambda), \tag{13}$$

while r_k and r represent the residual of the dualized constraints, i.e.:

$$r_k(x_k, u_k) = -\bar{A}_k x_k(\mu_k, \lambda) - \bar{B}_k u_k(\mu_k, \lambda) + b_k \tag{14a}$$

$$r(u) = -\sum_{k=1}^{M} \bar{C}_k u_k(\mu_k, \lambda). \tag{14b}$$

Following directly from (12), the Hessian of the dual function is a (permuted) block arrowhead matrix of the form:

$$\nabla^2 d(\mu, \lambda) = \begin{bmatrix} \frac{\partial r_1^*}{\partial \mu_1} & & & \frac{\partial r_1^*}{\partial \lambda} \\ & \ddots & & \vdots \\ & & \frac{\partial r_M^*}{\partial \mu_M} & \frac{\partial r_M^*}{\partial \lambda} \\ \frac{\partial r^*}{\partial \mu_1} & \cdots & \frac{\partial r^*}{\partial \mu_M} & \frac{\partial r^*}{\partial \lambda} \end{bmatrix}, \tag{15}$$

where we have omitted the arguments and defined $r_k^* = r_k(x_1^*(\mu_k, \lambda), u_k^*(\mu_k, \lambda))$ and $r^* = r(u^*(\mu, \lambda))$ for notational simplicity.

In this paper, we propose to solve (11) by updating the dual variables μ and λ according to:

$$\begin{bmatrix} \mu^+ \\ \lambda^+ \end{bmatrix} = \begin{bmatrix} \mu \\ \lambda \end{bmatrix} + t \begin{bmatrix} \Delta\mu \\ \Delta\lambda \end{bmatrix}, \tag{16}$$

where $\Delta\mu$ and $\Delta\lambda$ is a solution to the regularized Newton system:

$$\left(\nabla^2 d(\mu, \lambda) + \bar{F}\right) \begin{bmatrix} \Delta\mu \\ \Delta\lambda \end{bmatrix} = -\nabla d(\mu, \lambda) \tag{17}$$

and the step size $t \in (0, 1]$ is chosen to enforce convergence. The regularization $\bar{F}$ is selected to enforce $\nabla^2 d(\mu, \lambda) + \bar{F} \succ 0$, whenever the dual Hessian is rank

deficient. Specifically, this happens when the active local constraints together with the dualized constraints form linear dependencies [23]. In cases where the Linear Independence Constraint Qualification (LICQ) is fulfilled for all compatible active sets, $\bar{F}$ can be omitted.

The remaining of the paper is organized as follows. In Section 3, we describe an efficient way for calculating the dual gradient and the dual Hessian. In Section 4, we propose a parallelizable method for solving the Newton system and in Section 5 we summarize the resulting optimization scheme.

3 Calculating derivatives

In this section, we detail a method for calculating the dual gradient and the dual Hessian. Specifically, we propose a method for solving the subproblems (13), that are needed for the forming of the gradient, and a method for calculating the partial derivatives needed to form the Hessian.

3.1 Solving the subproblems

To evaluate the dual function, see (10), and its gradient, see (12), the optimal solutions $x_k^*(\mu_k, \lambda)$ and $u_k^*(\mu_k, \lambda)$ to the subproblems are needed. Because of the structure of the Lagrange function the calculations are separable and can be performed in parallel on a multiple core architecture. Additionally, as we shall see, the dualization of the non-anticipativity constraints and the dynamic equations results in subproblems with trivial solutions.

Note that $x_k^*(\mu_k, \lambda)$ and $u_k^*(\mu_k, \lambda)$ are given as solutions to a QP of the following form:

$$\min_{x_k, u_k} \; V_k(x_k, u_k) + \mu_k^\top \bar{A}_k x_k + \left(\lambda^\top \bar{C}_k + \mu_k^\top \bar{B}_k\right) u_k \tag{18a}$$

$$\text{s.t.} \; \underline{x}_k \leq x_k \leq \overline{x}_k \tag{18b}$$

$$\underline{u}_k \leq u_k \leq \overline{u}_k. \tag{18c}$$

Since the matrices $\bar{Q}_k$ and $\bar{R}_k$ are diagonal, their eigenvectors are aligned with the coordinate axes, i.e., aligned with the box constraints. This implies that the optimal solution can be found by a component-wise clipping of the unconstrained solution, as described in [13], i.e., $x_k^*(\mu_k, \lambda)$ and $u_k^*(\mu_k, \lambda)$ can be calculated as:

$$x_k^*(\mu_k, \lambda) = \text{mid}\left(\underline{x}_k, \overline{x}_k, -\bar{Q}_k^{-1}\left(\bar{q}_k + \bar{A}_k^\top \mu_k\right)\right) \tag{19a}$$

$$u_k^*(\mu_k, \lambda) = \text{mid}\left(\underline{u}_k, \overline{u}_k, -\bar{R}_k^{-1}\left(\bar{r}_k + \bar{B}_k^\top \mu_k + \bar{C}_k^\top \lambda\right)\right), \tag{19b}$$

where $\text{mid}(a, b, c)$ is a vector containing the component-wise median of its three arguments.

The dominating computational cost for finding $x_k^*(\mu, \lambda)$ and $u_k^*(\mu, \lambda)$ is therefore a banded matrix-vector multiplication which is negligible compared to the solution of the Newton system.

3.2 Calculating the dual Hessian

In this subsection, we detail the calculation of the partial derivatives in the dual Hessian (15), i.e., the calculation of:

$$\frac{\partial r_k^*}{\partial \mu_k} = -\bar{A}_k \frac{\partial x_k^*(\mu_k, \lambda)}{\partial \mu_k} - \bar{B}_k \frac{\partial u_k^*(\mu_k, \lambda)}{\partial \mu_k} \tag{20a}$$

$$\frac{\partial r_k^*}{\partial \lambda} = -\bar{A}_k \frac{\partial x_k^*(\mu_k, \lambda)}{\partial \lambda} - \bar{B}_k \frac{\partial u_k^*(\mu_k, \lambda)}{\partial \lambda} \tag{20b}$$

$$\frac{\partial r^*}{\partial \mu_k} = -\bar{C}_k \frac{\partial u_k^*(\mu_k, \lambda)}{\partial \mu_k} \tag{20c}$$

$$\frac{\partial r^*}{\partial \lambda} = -\sum_{k=1}^{M} \bar{C}_k \frac{\partial u_k^*(\mu_k, \lambda)}{\partial \lambda}. \tag{20d}$$

In the following, we denote the active constraints of (18b) and (18c) with:

$$\bar{D}_{\mathcal{A},k} x_k = d_{\mathcal{A},k} \tag{21a}$$

$$\bar{E}_{\mathcal{A},k} u_k = e_{\mathcal{A},k}. \tag{21b}$$

Note that every row in $\bar{D}_{\mathcal{A},k}$ and $\bar{E}_{\mathcal{A},k}$ has only one non-zero element, namely 1 if the row corresponds to an active upper bound or -1 if the row corresponds to an active lower bound.

By introducing the dual variables y_k and z_k corresponding to (21a) and (21b) respectively, it follows from the optimality conditions of (18) that the following holds:

$$0 = \bar{Q}_k \frac{\partial x_k^*(\mu_k, \lambda)}{\partial \lambda} + \bar{D}_{\mathcal{A},k}^\top \frac{\partial y_k^*(\mu_k, \lambda)}{\partial \lambda} \tag{22a}$$

$$0 = \bar{R}_k \frac{\partial u_k^*(\mu_k, \lambda)}{\partial \lambda} + \bar{C}_k^\top + \bar{E}_k^\top \frac{\partial z_k^*(\mu_k, \lambda)}{\partial \lambda} \tag{22b}$$

$$0 = \bar{D}_{\mathcal{A},k} \frac{\partial x_k^*(\mu_k, \lambda)}{\partial \lambda} \tag{22c}$$

$$0 = \bar{E}_{\mathcal{A},k} \frac{\partial u_k^*(\mu_k, \lambda)}{\partial \lambda}. \tag{22d}$$

By solving (22) for $\frac{\partial x_k^*(\mu_k,\lambda)}{\partial \lambda}$ and $\frac{\partial u_k^*(\mu_k,\lambda)}{\partial \lambda}$ we obtain:

$$\frac{\partial x_k^*(\mu_k, \lambda)}{\partial \lambda} = 0 \tag{23a}$$

$$\frac{\partial u_k^*(\mu_k, \lambda)}{\partial \lambda} = -\bar{R}_{\mathcal{I},k}^{-1} \bar{C}_k^\top, \tag{23b}$$

where we have introduced $\bar{R}_{\mathcal{I},k}^{-1} = \bar{R}_k^{-1}(I - \bar{E}_{\mathcal{A},k}^\top \bar{E}_{\mathcal{A},k})$. Observe that $\bar{R}_{\mathcal{I},k}^{-1}$ is identical to $\bar{R}_k^{-1}$, except that every diagonal element corresponding to an active constraint is replaced by a zero.

Let us now focus on the computation of $\frac{\partial x_k^*(\mu_k,\lambda)}{\partial \mu_k}$ and $\frac{\partial u_k^*(\mu_k,\lambda)}{\partial \mu_k}$. Similarly to (22), we obtain the following linear system:

$$0 = \bar{Q}_k \frac{\partial x_k^*(\mu_k,\lambda)}{\partial \mu_k} + \bar{A}_k^\top + \bar{D}_{\mathcal{A},k}^\top \frac{\partial y_k^*(\mu_k,\lambda)}{\partial \mu_k} \tag{24a}$$

$$0 = \bar{R}_k \frac{\partial u_k^*(\mu_k,\lambda)}{\partial \mu_k} + \bar{B}_k^\top + \bar{E}_k^\top \frac{\partial z_k^*(\mu_k,\lambda)}{\partial \mu_k} \tag{24b}$$

$$0 = \bar{D}_{\mathcal{A},k} \frac{\partial x_k^*(\mu_k,\lambda)}{\partial \mu_k} \tag{24c}$$

$$0 = \bar{E}_{\mathcal{A},k} \frac{\partial u_k^*(\mu_k,\lambda)}{\partial \mu_k}. \tag{24d}$$

Using block elimination, it can be verified that:

$$\frac{\partial x_k^*(\mu_k,\lambda)}{\partial \mu} = -\bar{Q}_{\mathcal{I},k}^{-1} \bar{A}_k^\top \tag{25a}$$

$$\frac{\partial u_k^*(\mu_k,\lambda)}{\partial \mu_k} = -\bar{R}_{\mathcal{I},k}^{-1} \bar{B}_k^\top, \tag{25b}$$

where we have introduced the notation $\bar{Q}_{\mathcal{I},k}^{-1} = \bar{Q}_k^{-1}(I - \bar{D}_{\mathcal{A},k}^\top \bar{D}_{\mathcal{A},k})$. Note again that $\bar{Q}_{\mathcal{I},k}^{-1}$ is identical to $\bar{Q}_k^{-1}$, except that every diagonal element corresponding to an active constraint is replaced by a zero.

Finally, by using (23) and (25) the expressions in (20) can be simplified as:

$$\frac{\partial r_k^*}{\partial \mu_k} = \bar{A}_k \bar{Q}_{\mathcal{I},k}^{-1} \bar{A}_k^\top + \bar{B}_k \bar{R}_{\mathcal{I},k}^{-1} \bar{B}_k^\top \tag{26a}$$

$$\frac{\partial r_k^*}{\partial \lambda} = \bar{B}_k \bar{R}_{\mathcal{I},k}^{-1} \bar{C}_k^\top \tag{26b}$$

$$\frac{\partial r^*}{\partial \mu_k} = \bar{C}_k \bar{R}_{\mathcal{I},k}^{-1} \bar{B}_k^\top \tag{26c}$$

$$\frac{\partial r^*}{\partial \lambda} = \sum_{k=1}^{M} \bar{C}_k \bar{R}_{\mathcal{I},k}^{-1} \bar{C}_k^\top. \tag{26d}$$

This implies that the dual Hessian can be formed cheaply at the cost of a banded matrix-matrix multiplication once the subproblems are solved and the active set is known. In [13], a similar expression is used for the dual Hessian in the context of standard MPC.

4 Efficient solution of the Newton system

To achieve an efficient implementation, it is crucial to exploit the structure of the Newton system in the computation of the search direction. Additionally, since the dual Hessian can be singular, a regularization strategy is needed in order to solve (17). In this section, we aim at addressing these issues.

4.1 Alternative Newton system

To exploit the sparsity structure of (15), block elimination can be used to obtain alternative formulations of the search direction computation. For problems of the form (2), we propose to exploit the sparsity structure and to enable parallel computations by eliminating $\Delta\mu_k$ from (17). In this case, the regularization matrix $\bar{F}$ can be calculated implicitly in a parallel fashion. The resulting formulation can be expressed as:

$$(J + F)\Delta\lambda = -r + \sum_{k=1}^{M} \frac{\partial r}{\partial \mu_k} \left(\frac{\partial r_k}{\partial \mu_k} + F_k \right)^{-1} r_k \tag{27a}$$

$$\left(\frac{\partial r_k}{\partial \mu_k} + F_k \right) \Delta\mu_k = -r_k - \frac{\partial r_k}{\partial \lambda}\Delta\lambda, \quad k = 1, \ldots, M, \tag{27b}$$

where we have introduced:

$$J = \frac{\partial r}{\partial \lambda} - \sum_{k=1}^{M} \frac{\partial r}{\partial \mu_k} \left(\frac{\partial r_k}{\partial \mu_k} + F_k \right)^{-1} \frac{\partial r_k}{\partial \lambda} \tag{28}$$

and $F_k \in \mathbb{S}_+^{nN}$ and $F \in \mathbb{S}_+^{p}$ are chosen to enforce that $\frac{\partial r_k}{\partial \mu_k} + F_k \succ 0$ and $J + F \succ 0$.

We can then first find $\Delta\lambda$ by solving (27a), and then use the result to find $\Delta\mu_k$. Observe that (27b) is trivially decomposable and can be solved in parallel on M CPUs.

4.2 Regularization

To guarantee that the search direction, $\Delta\mu$ and $\Delta\lambda$, is a descent direction, the regularized dual Hessian has to be positive definite. In the following lemma, we show that the formulation (27) results in search directions that are descent directions. This implies that the regularization matrix $\bar{F}$ can be calculated implicitly and partly in parallel without explicitly forming the dual Hessian.

Lemma 1 *If the conditions:*

1. $\frac{\partial r_k}{\partial \mu_k} + F_k \succ 0$, *for* $k = 1, \ldots, M$
2. $J + F \succ 0$

hold, then the regularized dual Hessian is positive definite.

Proof Notice that matrix J as defined in (28) is the Schur complement of the top-left block of the dual Hessian. The two conditions in Lemma 1 are equivalent to the Schur complement condition for positive definiteness of a symmetric matrix, which concludes the proof.

4.3 Finding the search directions $\Delta\mu_k$

In this subsection, we detail the calculation of the search directions $\Delta\mu_k$, i.e., a solution method for (27b).

First, let us introduce the following notation:

$$\Lambda_k = \frac{\partial r_k}{\partial \mu_k} + F_k = \bar{A}_k \bar{Q}_{\mathcal{I},k}^{-1} \bar{A}_k^\top + \bar{B}_k \bar{R}_{\mathcal{I},k}^{-1} \bar{B}_k^\top + F_k. \tag{29}$$

We observe that if F_k is restricted to have the same sparsity structure as $\frac{\partial r_k}{\partial \mu_k}$, then Λ_k has a block-tridiagonal structure , i.e.:

$$\Lambda_k = \begin{bmatrix} \Lambda_{k,(1,1)} & \Lambda_{k,(1,2)} & & \\ \Lambda_{k,(1,2)}^\top & \Lambda_{k,(2,2)} & \ddots & \\ & \ddots & \ddots & \Lambda_{k,(N-1,N)} \\ & & \Lambda_{k,(N-1,N)}^\top & \Lambda_{k,(N,N)} \end{bmatrix}, \tag{30}$$

with blocks $\Lambda_{k,(i,j)} \in \mathbb{R}^{n\times n}$. Consequently, the Cholesky factor L_k of Λ_k, such that $\Lambda_k = L_k L_k^\top$, is block-lower bidiagonal:

$$L_k = \begin{bmatrix} L_{k,(1,1)} & & & \\ L_{k,(2,1)} & L_{k,(2,2)} & & \\ & \ddots & \ddots & \\ & & L_{k,(N,N-1)} & L_{k,(N,N)} \end{bmatrix} \tag{31}$$

with blocks $L_{k,(i,j)} \in \mathbb{R}^{n\times n}$ that can be calculated using the following block-wise procedure:

$$\Lambda_{k,(1,1)} = L_{k,(1,1)} L_{k,(1,1)}^\top \tag{32a}$$

$$\Lambda_{k,(i,i+1)} = L_{k,(i,i)} L_{k,(i+1,i)}^\top, \quad i = 1, \dots, N \tag{32b}$$

$$\Lambda_{k,(i,i)} - L_{k,(i,i-1)} L_{k,(i,i-1)}^\top = L_{k,(i,i)} L_{k,(i,i)}^\top, \ i = 2, \dots, N. \tag{32c}$$

Using L_k, the search direction $\Delta\mu_k$ can be calculated from (27b) via a sparse forward and backward substitution. Note that the calculation and factorization of Λ_k and the calculation of $\Delta\mu_k$ for $k = 1, \dots, M$ can be performed in a parallel fashion on M CPUs.

4.4 Finding the search direction $\Delta\lambda$

The search direction $\Delta\lambda$ is provided by the linear system (27a). In this subsection, we establish the sparsity structure of J, and propose a method for solving (27a).

Let us start with detailing the sparsity structure of J. Using (26) and (28), we write J as:

$$J = \sum_{k=1}^{M} \bar{C}_k \left(\bar{R}_{\mathcal{I},k}^{-1} - \bar{R}_{\mathcal{I},k}^{-1} \bar{B}_k^\top \Lambda_k^{-1} \bar{B}_k \bar{R}_{\mathcal{I},k}^{-1} \right) \bar{C}_k^\top . \tag{33}$$

Using the structure of $\bar{C}_k$, detailed in (5), we note that J is also block-tridiagonal:

$$J = \begin{bmatrix} J_{1,1} & J_{1,2} & & \\ J_{1,2} & J_{2,2} & \ddots & \\ & \ddots & \ddots & J_{M-2,M-1} \\ & & J_{M-2,M-1} & J_{M-1,M-1} \end{bmatrix}, \tag{34}$$

with blocks $J_{k,k} \in \mathbb{S}^{mn_{c,(k,k+1)}}$ and $J_{k,k+1} \in \mathbb{R}^{mn_{c,(k+1,k+2)} \times mn_{c,(k,k+1)}}$ given by:

$$J_{k,k} = C_{k,k+1}(K_k + K_{k+1})C_{k,k+1}^\top \tag{35a}$$

$$J_{k-1,k} = -C_{k-1,k} K_k C_{k,k+1}^\top, \tag{35b}$$

where we have introduced:

$$K_k = \bar{R}_{\mathcal{I},k}^{-1} - \bar{R}_{\mathcal{I},k}^{-1} \bar{B}_k^\top \Lambda_k^{-1} \bar{B}_k \bar{R}_{\mathcal{I},k}^{-1}. \tag{36}$$

Note that the size of the blocks in the block-tridiagonal structure of J is inherited from the structure of the scenario tree. More specifically, the size of $J_{k,k}$ is determined by the number of common nodes between scenarios k and $k+1$, whereas the size of $J_{k,k+1}$ is determined by the number of common nodes between scenarios k, $k+1$ and scenarios $k+1$, $k+2$.

If F is restricted to have the same sparsity structure as J, the Cholesky factor L yielding $J + F = LL^\top$ is block lower bidiagonal:

$$L = \begin{bmatrix} L_{1,1} & & & \\ L_{2,1} & L_{2,2} & & \\ & \ddots & \ddots & \\ & & L_{M-1,M-2} & L_{M-1,M-1} \end{bmatrix} \tag{37}$$

and can be calculated efficiently according to:

$$J_{1,1} + F_{1,1} = L_{1,1} L_{1,1}^\top \tag{38a}$$

$$J_{k,k+1} + F_{k,k+1} = L_{k,k} L_{k+1,k}^\top, \quad k = 1, \ldots, M-1 \tag{38b}$$

$$J_{k,k} + F_{k,k} - L_{k,k-1} L_{k,k-1}^\top = L_{k,k} L_{k,k}^\top, \; k = 2, \ldots, M-1. \tag{38c}$$

Using L, the search direction $\Delta\lambda$ can be calculated from (27a) via a sparse forward and backward substitution.

4.5 Forming matrix J

In this subsection, we propose a parallelizable method for forming matrix J, where we put the emphasis on re-using components that are available from previous computations. First, let us introduce the following block partitioning of K_k:

$$K_k = \begin{bmatrix} K_{k,(1,1)} & \dots & K_{k,(N,1)}^\top \\ \vdots & \ddots & \vdots \\ K_{k,(N,1)} & \dots & K_{k,(N,N)} \end{bmatrix} \tag{39}$$

with blocks $K_{k,(i,j)} \in \mathbb{R}^{m\times m}$, and the notation:

$$K_k^{N_1,N_2} = \begin{bmatrix} K_{k,(1,1)} & \dots & K_{k,(N_2,1)}^\top \\ \vdots & \ddots & \vdots \\ K_{k,(N_1,1)} & \dots & K_{k,(N_1,N_2)} \end{bmatrix} \tag{40}$$

for the $N_1 \times N_2$ most top-left blocks of matrix K_k. Due to the structure of $C_{k,k+1}$, see (6), the expressions in (35) can now be simplified as:

$$J_{k,k} = K_k^{n_{c,(k,k+1)},n_{c,(k,k+1)}} + K_{k+1}^{n_{c,(k,k+1)},n_{c,(k,k+1)}} \tag{41a}$$

$$J_{k-1,k} = -K_k^{n_{c,(k-1,k)},n_{c,(k,k+1)}}. \tag{41b}$$

Our aim is to calculate a part of K_k for each scenario, where the size of the part that needs to be computed is governed by the number of common nodes in the scenario tree for scenario k and its neighboring scenarios. In the following, we propose a method for calculating top-left parts of K_k.

Let us first make the observation that we can express K_k as:

$$K_k = \bar{R}_{\mathcal{I},k}^{-1} - \bar{Z}_k^\top \Lambda_k^{-1} \bar{Z}_k, \tag{42}$$

where we have introduced:

$$\bar{Z}_k = \bar{B}_k \bar{R}_{\mathcal{I},k}^{-1}. \tag{43}$$

Let us now consider the calculation of the term $\bar{Z}_k^\top \Lambda_k^{-1} \bar{Z}_k$. Due to symmetry, we can find a rectangular factor U_k, such that $\bar{Z}_k^\top \Lambda_k^{-1} \bar{Z}_k = U_k^\top U_k$, where we introduce the following block partitioning of U_k:

$$U_k = \begin{bmatrix} U_{k,(1,1)} & \dots & U_{k,(1,N)} \\ \vdots & \ddots & \vdots \\ U_{k,(N,1)} & \dots & U_{k,(N,N)} \end{bmatrix} \tag{44}$$

with blocks $U_{k,(i,j)} \in \mathbb{R}^{m\times m}$. By using this partitioning, we can express $K_{k,(i,i)}$ and $K_{k,(i,j)}$ as:

$$K_{k,(i,i)} = R_{\mathcal{I},k,i}^{-1} - U_{k,(1,i)}^\top U_{k,(1,i)} - \dots - U_{k,(N,i)}^\top U_{k,(N,i)} \tag{45a}$$

$$K_{k,(i,j)} = -U_{k,(1,i)}^\top U_{k,(1,j)} - \dots - U_{k,(N,i)}^\top U_{k,(N,j)}, \tag{45b}$$

where $R^{-1}_{\mathcal{I},k,i} \in \mathbb{R}^{m\times m}$ denotes the block of $\bar{R}^{-1}_{\mathcal{I},k}$ corresponding to time stage $i+1$. This implies that we need to compute the first $n_k = \max\{n_{c,(k-1,k)}, n_{c,(k,k+1)}\}$ block columns[1] of U_k, in the following denoted as $U_k^{n_k}$, in order to assemble the required blocks $K_{k,(i,j)}$. Next, we propose to compute $U_k^{n_k}$ by re-using the Cholesky factor L_k in (31).

Observe that the factor U_k can be calculated according to:

$$L_k U_k = \bar{Z}_k. \tag{46}$$

This implies that $U_k^{n_k}$ can be computed by using only the corresponding part of $\bar{Z}_k$, i.e. by solving:

$$L_k U_k^{n_k} = \bar{Z}_k^{n_k}, \tag{47}$$

where $\bar{Z}_k^{n_k} \in \mathbb{R}^{nN\times mn_k}$ denotes the first mn_k columns of $\bar{Z}_k$.

The computational cost of building J consists then of M matrix forward substitutions, to calculate $U_k^{n_k}$, and M matrix-matrix multiplications, to calculate the required parts of K_k. Moreover, the computations are trivially decomposable and can be performed in parallel on M CPUs.

5 The dual Newton scheme

In this section, we provide a summary of the proposed dual Newton strategy and suggest some algorithmic details to enhance its performance.

The method is summarized in Algorithm 1. Note that each iteration consists of a possibly damped Newton step on the dual variables, i.e., the calculation of the search directions $\Delta\mu$ and $\Delta\lambda$ and an appropriately chosen step size t. The calculation of the search directions can be performed to a large extent in parallel as described in Section 3 and 4.

5.1 Choice of step size

Due to the non-smoothness of the dual function, a globalization strategy is needed to ensure convergence of the dual Newton strategy. Several line search strategies have been proposed in [14], with the common property that they require the evaluation of the dual function for candidate step sizes $t \in (0, 1]$, i.e. the evaluation of $d(\mu + t\Delta\mu, \lambda + t\Delta\lambda)$. Note that the dual function can be evaluated by performing only small matrix-vector multiplications, which are negligible compared to solving the large Newton system.

In this paper, we select t based on an Armijo line search with backtracking. More specifically, for every search direction $\Delta\mu$ and $\Delta\lambda$, we initialize the step size as $t = 1$, and backtrack according to:

$$t^+ = \beta t, \tag{48}$$

[1] Note that for the tree structures considered in this paper (see, e.g., Figure 1) it holds $n_k = N_r$ for all $k = 2, \dots M-1$.

Algorithm 1: Dual Newton strategy for multi-stage MPC

Input: μ, λ, τ
while $\|\nabla d(\mu,\lambda)\| > \tau$ **do**
  for *k=1,..., M* **do**
    Solve subsystems in parallel (19) to obtain $x_k^*(\mu_k,\lambda)$ and $u_k^*(\mu_k,\lambda)$
    Compute diagonal blocks of $\bar{Q}_{\mathcal{I},k}, \bar{R}_{\mathcal{I},k}$ based on the current active sets
    Compute r_k (14a) and summant of r (14b) to build $\nabla d(\mu,\lambda)$
    Build diagonal blocks of $\bar{Z}_k$ (43)
    Compute blocks of banded matrix Λ_k (29), re-using $\bar{Z}_k$
    Perform a banded reverse Cholesky factorization to form L_k (32)
    Perform a banded matrix substitution to form $U_k^{n_k}$ (47)
    Build top-left part of K_k (45)
  end
  Check termination condition and exit while loop if satisfied
  Form J (41) and perform a banded Cholesky factorization to calculate blocks of L (38)
  Perform banded forward-backward substitutions to build right hand side of (27a)
  Perform banded forward-backward substitutions with L to calculate $\Delta\lambda$ (27a)
  for *k=1,..., M* **do**
    Build right hand side of (27b)
    Perform banded forward-backward substitutions with L_k to calculate $\Delta\mu_k$ (27b)
  end
  Perform line search to calculate step size t (Section 5.1)
  Update dual multipliers to complete the Newton step (16)
end

until the following criterion is fulfilled:

$$d(\mu + t\Delta\mu, \lambda + t\Delta\lambda) \leq d(\mu,\lambda) + \sigma \nabla d(\mu,\lambda)^\top \begin{bmatrix} \Delta\mu \\ \Delta\lambda \end{bmatrix} \tag{49}$$

for constants $\sigma, \beta \in (0,1)$. Given the low computational cost of evaluating the dual function, we can afford a careful line search, i.e., large constants σ and β can be used. Additionally, observe that due to convexity of $d(\mu,\lambda)$, there exists always a step size $t \in (0,1]$ such that (49) is fulfilled.

Finally, note that the line search strategy is parallelizable since the evaluation of the dual function can be performed in parallel on M CPUs. In this context the communication overhead coming from the parallelization can be reduced by evaluating several candidate step sizes at once.

5.2 Elimination of redundant constraints

In the context of dual Newton strategies, the Hessian of the dual function is rank deficient if the active constraints (2d) and (2e) together with the dualized constraints (2b) and (2c) are linearly dependent, leading to an inconsistent Newton system [23]. This implies that the need for regularization can be reduced by eliminating redundant constraints.

For the multi-stage MPC problem (2), linear dependencies occur if inequality constraints become active for scenarios participating in the same node. In consequence, for every node in the scenario tree, we only enforce an inequality constraint on one scenario participating in that node. The inequality constraints are thus enforced implicitly on the other scenarios via the dynamics and the non-anticipativity constraints. For a detailed description see [20].

5.3 Warm-starting

One of the key advantages of using the dual Newton strategy in the context of MPC is its capability to exploit the similarity between two subsequent QPs. Specifically, if the method is provided with an initialization of the dual variables which corresponds to the correct active set, convergence is achieved in one step [14].

An often practically efficient strategy for standard MPC is to initialize the method by using a time shift of the dual solution of the previous problem instance. Due to the tree structure of a multi-stage MPC problem, a time shift of this kind can be performed in several different ways. However, we have not found a shifting strategy that consistently performs better than a warm-start where the problem is initialized at the non-shifted dual solution of the previous problem instance. For this reason, we only consider a non-shifted warm-start in the numerical experiments.

5.4 Re-use of intermediate results and reverse Cholesky factorizations

Another feature of the algorithm that can be exploited to improve its performance is the re-use of intermediate results between iterations. Specifically, the blocks of matrices $\bar{Q}_{\mathcal{I},k}, \bar{R}_{\mathcal{I},k}, \bar{Z}_k$ and Λ_k do not need to be recalculated if the subsystems they depend on have the same active set as in the previous iteration. These savings become even more prominent when combined with the elimination of redundant constraints described in Section 5.2, since blocks that depend only on unconstrained subsystems need to be calculated only once.

Furthermore, the banded Cholesky factorization of Λ_k only needs to start from the first block that has been updated since the last iteration. Since in MPC applications most active set changes happen at the beginning of the horizon, a reverse Cholesky factorization as proposed in [15] can typically reuse more factors than a forward one. Interestingly, this modification has also an impact on the computational cost of building K_k. Namely, the matrices U_k, which have many more block rows than columns if $N \gg N_r$, become upper (instead of lower) triangular and therefore the computational cost of the matrix substitution in (47) and the symmetric matrix multiplication in (45) is reduced.

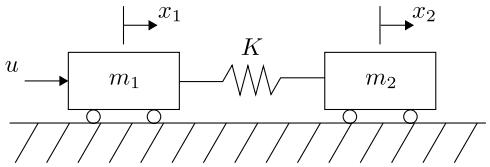

Fig. 2: Illustration of two-mass-spring system.

6 Software implementation

This section presents an open-source software implementation of Algorithm 1, which is tailored to embedded optimization. In the context of this paper, the term *embedded optimization* refers to MPC applications where small to medium scale optimization problems need to be solved at high sampling frequencies and occasionally on resource-constrained hardware.

Since computation times are critical to meet the often challenging real-time requirements and embedded hardware does not always support advanced programming languages, the algorithm is entirely written in C. The fact that the sparsity structure of the problem is exploited in the algorithmic level (see e.g. Section 4) implies that a key factor for high performance is the use of efficient, dense linear algebra routines. To this purpose, all matrix and vector operations are performed using the recently developed software `BLASFEO` [2]. `BLASFEO` is an open-source, highly optimized library that provides `BLAS`- and `LAPACK`-like routines tailored to embedded optimization algorithms. The expected speed-up with respect to existing `BLAS` and `LAPACK` implementations is about 20-30% for `BLAS` and a factor of 2 to 3 for `LAPACK` routines as observed in [17]. Finally, in order to parallelize the solution of the Newton system (see Section 4) as well as the rest of the operations that are trivially parallelizable, the `openMP` Application Programming Interface (API) is used [11]. The code is available on Github as part of the `treeQP` toolbox [3].

7 Numerical experiments

To study the properties of the presented dual Newton scheme we consider a benchmark example of a two-mass-spring system [36], [22], as illustrated in Figure 2. The two carts have the same mass $m_1 = m_2 = 1\,\mathrm{kg}$ while the spring constant K is uncertain, ranging between $K_{\min} = 1\,\mathrm{N/m}$ and $K_{\max} = 10\,\mathrm{N/m}$. The linear model that describes the system in continuous time is:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -K/m_1 & K/m_1 & 0 & 0 \\ K/m_2 & -K/m_2 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1/m_1 \\ 0 \end{bmatrix} u \tag{50}$$

where x_1 and x_2 denote the displacement of the two carts from their initial position, x_3 and x_4 denote the velocities of the carts and u is the force applied

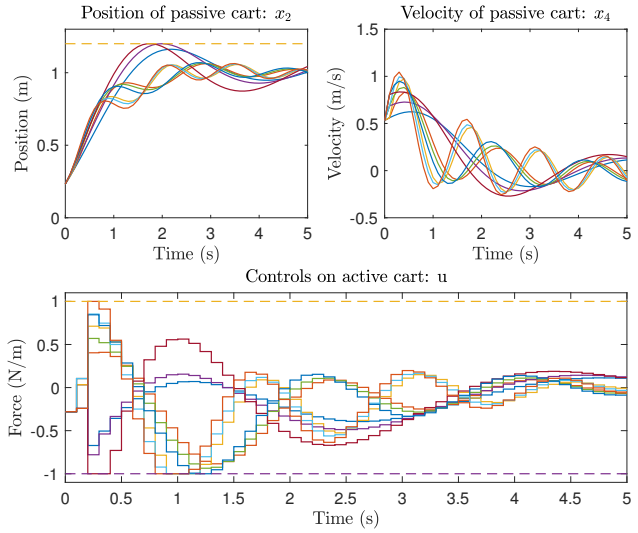

Fig. 3: Optimal solution of x_2, x_4 and u for one MPC problem.

on the active cart, i.e. on the left cart of Figure 2. The control objective is to track a unit step command on the position x_2 with a 20% maximum overshoot, while keeping the control actions in the range $-1\,\mathrm{N} \leq u \leq 1\,\mathrm{N}$.

The continuous time dynamics in (50) are discretized using matrix exponentials, assuming piecewise constant controls in intervals of $T_s = 0.1\,\mathrm{s}$. The initial condition is $x_0 = [0,\, 0,\, 0,\, 0]^\top$ and the steady state point in the tracking objective $x_{\mathrm{ref}} = [1,\, 1,\, 0,\, 0]^\top, u_{\mathrm{ref}} = 0$. Finally, the quadratic weights in the objective function are set to:

$$\begin{aligned} Q_{k,i} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{for } \; k = 1, \ldots, M, \; i = 1, \ldots, N-1, \\ R_{k,i} &= 1, \qquad\qquad\qquad \text{for } \; k = 1, \ldots, M, \; i = 0, \ldots, N-1, \end{aligned} \tag{51}$$

and $Q_{k,N} = 10\, Q_{k,i}$ for the terminal state of each scenario. All simulations were performed on a 3,1 GHz Intel Xeon processor (E5-2687W) with 8 cores and 32 GB of RAM, running Ubuntu 14.04.

An example of optimized state and control trajectories for one multi-stage MPC problem is shown in Figure 3, where $N = 50, N_r = 2$ and $m_d = 3$. Note that the non-anticipativity constraints impose one common control input for all scenarios at the first step (the value ultimately applied by the controller) and up to 3 different controls for the second step. The state and control bounds are satisfied for all trajectories.

To demonstrate the potential of the presented scheme from a numerical point of view, we compare its performance against the interior point method

of [16], which extends the QP solver `HPMPC` [1] to exploit the underlying tree sparsity. Note that the results in [16] suggest that `HPMPC` can outperform both tailored, code generated solvers and general purpose sparse solvers for the problems of interest.

For the purposes of this comparison, we run a series of closed loop simulations (each containing 50 MPC problems) that cover all possible combinations of: $K_{\text{nom}} \in \{2, 3, \ldots, 8\}$, $N \in \{20, 30, 40, 50\}$, $N_r \in \{1, 2, 3, 4\}$ and $m_d \in \{2, 3, 4\}$. The uncertainty realizations are linearly equally spaced between $K_{\min}$ and $K_{\max}$. Both cold- and warm-started versions of the dual Newton scheme are considered. Cold-start means that the solver is initialized at zero. Warm-start means that the solver is initialized at the previous solution.

To compare the execution times, we make use of performance profiles [12]. The y-axis of a performance profile is the cumulative distribution function of the performance ratio $r_{p,s}$, defined as

$$r_{\text{p,s}} = \frac{t_{\text{p,s}}}{\min\{t_{\text{p,s}} : s \in \mathcal{S}\}} \, .$$

Here, $t_{p,s}$ is the time required for solver s to solve problem p and $\mathcal{S}$ is the set of benchmarked solvers. Figure 4 contains three performance profiles. On the left, all 16800 MPC problems of the closed loop simulations take part in the plot. Since problems with no active constraints at the optimal solution may be considered trivial, the performance profile in the middle contains only those problems that have at least one constraint active. Finally, since the worst-case performance is often the most important metric in practical MPC applications, the rightmost profile takes into account only the highest execution time of each algorithm per closed loop simulation. The results indicate that the proposed scheme in combination with a simple warm-starting strategy is the best performing solver for this benchmark example. However, the performance gap is significantly smaller in the worst-case plot, where `HPMPC` is the faster solver for approximately 40% of the cases. In trivial problem instances where the dual Newton scheme may converge in one full step, the execution time is up to an order of magnitude faster than the interior point method. Note that the two algorithms are executed sequentially (in lack of a parallel implementation of `HPMPC`) although the presented dual Newton strategy can highly profit from a parallel architecture.

To assess the performance gains of parallelization, we run the algorithm using openMP and compare the worst-case CPU times for 6 closed loop simulations using up to 8 threads. The algorithm is warm-started in all simulations. The horizon length is fixed to $N = 20$ while N_r and m_d are varying. The spring constant of the simulation model is $K_{\text{nom}} = 5$. Table 1 reports the execution times in ms and in parenthesis the speed-up with respect to the sequential implementation, i.e., without using openMP. The table suggests that the performance of the algorithm can significantly benefit by parallelizing its key operations. Note that not all operations have the same degree of paralleliza-

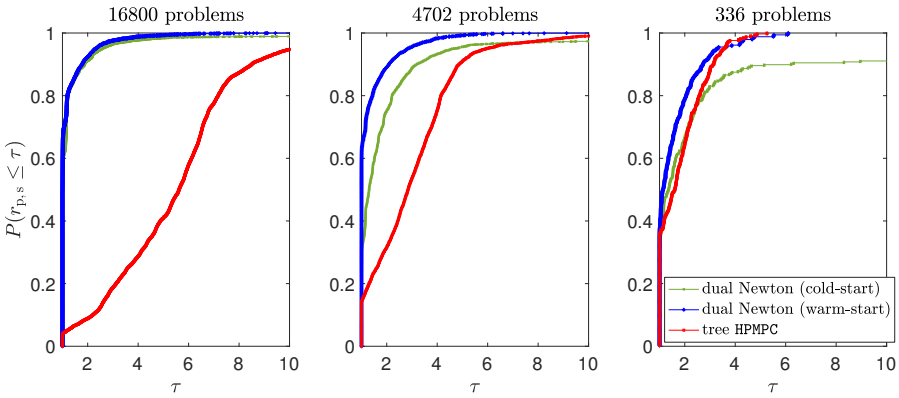

Fig. 4: Performance plot comparing the execution times of the presented dual Newton strategy and the interior point method of [16] on all closed loop simulations (left), on problem instances with at least one active constraint at the optimal solution (middle) and on worst-case performance of each simulation (right).

Table 1: Worst-case CPU time in closed loop (measured in ms) and associated speed-up (in parenthesis) with respect to sequential implementation for $N = 20$, $K_{\mathrm{nom}} = 5$ and different combinations of N_r and m_d.

# threads	$m_d = 2$			$m_d = 4$		
	$N_r = 2$	$N_r = 3$	$N_r = 4$	$N_r = 2$	$N_r = 3$	$N_r = 4$
1	1.01 (1.0)	1.66 (1.0)	2.72 (1.0)	5.38 (1.0)	46.99 (1.0)	295.00 (1.0)
2	0.72 (1.4)	1.18 (1.4)	2.01 (1.4)	3.40 (1.6)	28.69 (1.6)	179.08 (1.6)
4	0.42 (2.4)	0.84 (2.0)	1.45 (1.9)	2.26 (2.4)	15.57 (3.0)	118.87 (2.5)
8	0.27 (3.8)	0.49 (3.4)	1.00 (2.7)	1.84 (2.9)	10.68 (4.4)	60.88 (4.8)

tion and that differences in speed-up may also come from different number of Newton and/or line search iterations.

8 Conclusions

In this paper, we presented a dual Newton strategy for multi-stage MPC. To achieve a low computational complexity per iteration, we propose to dualize both the dynamic constraints and the non-anticipativity constraints. As a result, the dual function can be evaluated very cheaply, and the gradient and Hessian of the dual function can be formed at a low computational cost. Additionally, the dual Newton system can be reformulated into several small and highly structured linear systems that to a large extent can be solved in parallel. The presented scheme is complemented by an open-source software

implementation and benchmarked against a structure-exploiting interior point method that targets the same field of applications.

References

1. HPMPC. https://github.com/giaf/hpmpc (2014)
2. BLASFEO. https://github.com/giaf/blasfeo (2016)
3. treeQP: A toolbox for tree-sparse quadratic programming. https://github.com/dkouzoup/treeQP (2017)
4. Bernardini, D., Bemporad, A.: Scenario-based model predictive control of stochastic constrained linear systems. In: IEEE Conference on Decision and Control (2009)
5. Bertsekas, D., Tsitsiklis, J.N.: Parallel and distributed computation: Numerical methods. Prentice Hall (1989)
6. Birge, J.: Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs. Operations Research **33**, 989–1007 (1985)
7. Birge, J.R., Holmes, D.F.: Efficient Solution of Two-Stage Stochastic Linear Programs Using Interior Point Methods. Computational Optimization and Applications **1**, 245–276 (1992)
8. Blair, J.R.S., Peyton, B.: An Introduction to Chordal Graphs and Clique Trees, pp. 1–29. Springer New York (1993)
9. Boyd, S., Vandenberghe, L.: Convex Optimization. University Press, Cambridge (2004)
10. Campo, P.J., Morari, M.: Robust model predictive control. In: American Control Conference, pp. 1021–1026 (1987)
11. Dagum, L., Menon, R.: OpenMP: an industry standard API for shared-memory programming. IEEE Computational Science and Engineering **5**(1), 46–55 (1998)
12. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
13. Ferreau, H.J., Kozma, A., Diehl, M.: A parallel active-set strategy to solve sparse parametric quadratic programs arising in MPC. In: IFAC Nonlinear Model Predictive Control Conference (2012)
14. Frasch, J.V., Sager, S., Diehl, M.: A Parallel Quadratic Programming Method for Dynamic Optimization Problems. Mathematical Programming Computation **7**, 289–329 (2015)
15. Frasch, J.V., Vukov, M., Ferreau, H.J., Diehl, M.: A new quadratic programming strategy for efficient sparsity exploitation in SQP-based nonlinear MPC and MHE. In: Proceedings of the 19th IFAC World Congress (2014)
16. Frison, G., Kouzoupis, D., Diehl, M., Jørgensen, J.B.: A high-performance Riccati based solver for tree-structured quadratic programs. In: IFAC World Congress (2017)
17. Frison, G., Kouzoupis, D., Zanelli, A., Diehl, M.: BLASFEO: Basic linear algebra subroutines for embedded optimization. CoRR **abs/1704.02457** (2017). URL http://arxiv.org/abs/1704.02457
18. Hansson, A.: A primal-dual interior-point method for robust optimal control of linear discrete-time systems. IEEE Transactions on Automatic Control **45**, 1639–1655 (2000)
19. Jessup, E., Yang, D., Zenios, S.: Parallel factorization of structured matrices arising in stochastic programming. SIAM Journal on Optimization **4**, 833–846 (1994)
20. Klintberg., E., Dahl., J., Fredriksson., J., Gros., S.: An improved dual Newton strategy for scenario-tree MPC. In: IEEE Conference on Decision and Control (2016)
21. Klintberg, E., Gros, S.: A parallelizable interior-point method for two-stage robust MPC. IEEE Transactions on Control Systems Technology (2017)
22. Kothare, M., Balakrishnan, V., Morari, M.: Robust constrained model predictive control using linear matrix inequalities. Automatica **32**, 1361–1379 (1996)
23. Kozma, A., Klintberg, E., Gros, S., Diehl, M.: An improved distributed dual Newton-CG method for convex quadratic programming problems. In: American Control Conference (2014)
24. Leidereiter, C., Potschka, A., Bock, H.G.: Dual decomposition for QPs in scenario tree NMPC. In: European Control Conference (2015)

25. Lucia, S., Andersson, J., Brandt, H., Diehl, M., Engell, S.: Handling uncertainty in economic model predictive control: A comparative case study. Journal of Process Control **24**, 1247–1259 (2014)
26. Lucia, S., Finkler, T., Basak, B., Engell, S.: A new Robust NMPC Scheme and its Application to a Semi-batch Reactor Example. In: IFAC International Symposium on Advanced Control of Chemical Processes (2012)
27. Maiworm, M., Bathge, T., Findeisen, R.: Scenario-based model predictive control: Recursive feasibility and stability. In: IFAC Symposium on Advanced Control of Chemical Processes (2015)
28. Marti, R., Lucia, S., Sarabia, D., Paulen, R., Engell, S.: Improving scenario decomposition algorithms for robust nonlinear model predictive control. Computers and Chemical Engineering **79**, 30–45 (2015)
29. Mayne, D., Rakovic, S., Findeisen, R., Allgöwer, F.: Robust output feedback model predictive control of constrained linear systems. Automatica **42**, 1217–1222 (2006)
30. Mayne, D., Seron, M., Rakovic, S.: Robust model predictive control of constrained linear systems with bounded disturbances. Automatica **41**, 219–224 (2005)
31. Mulvey, J., Ruszczyński, A.: A New Scenario Decomposition Method for Large-Scale Stochastic Optimization. Operations Research **43**, 477–490 (1995)
32. Pakazad, S.K., Hansson, A., Andersen, M.S., Nielsen, I.: Distributed primal-dual interior-point methods for solving tree-structured coupled convex problems using message-passing. Optimization Methods & Software **32**(3), 401–435 (2016)
33. Munoz de la Pena, D., Bemporad, A., Alamo, T.: Stochastic programming applied to model predictive control. In: IEEE Conference on Decision and Control (CDC) (2005)
34. Steinbach, M.C.: Tree-Sparse Convex Programs. Mathematical Methods of Operations Research **56**(3), 347–376 (2002)
35. Vandenberghe, L., Andersen, M.S.: Chordal graphs and semidefinite optimization. Foundations and Trends in Optimization **1**(4), 241–433 (2015)
36. Wie., B., Bernstein., D.: Benchmark Problems for Robust Control Design. Journal of Guidance Control and Dynamics **15**, 1057–1059 (1992)
37. Zeilinger, M., Raimondo, D., Domahidi, A., Morari, M., Jones, C.: On real-time robust model predictive control. Automatica **50**, 683–694 (2014)