52nd IEEE Conference on Decision and Control
December 10-13, 2013. Florence, Italy

A Distributed Method for Convex Quadratic Programming Problems Arising in Optimal Control of Distributed Systems

Attila Kozma, Janick V. Frasch and Moritz Diehl

Abstract— **We propose a distributed algorithm for strictly convex quadratic programming (QP) problems with a generic coupling topology. The coupling constraints are dualized via Lagrangian relaxation. This allows for a distributed evaluation of the non-smooth dual function and its derivatives. We propose to use both the gradient and the curvature information within a non-smooth variant of Newton's method to find the optimal dual variables. Our novel approach is designed such that the large Newton system never needs to be formed. Instead, we employ an iterative method to solve the Newton system in a distributed manner. The effectiveness of the method is demonstrated on an academic optimal control problem. A comparison with state-of-the-art first order dual methods is given.**

I. Introduction

We are concerned with the solution of the separable convex quadratic program (QP), typically arising from optimal control of distributed systems, of the form

$$\min_{x} \sum_{k=1}^{n} \frac{1}{2} x_k^T H_k x_k + c_k^T x_k \tag{1a}$$

$$\text{s.t. } A_{i,j} x_i = B_{i,j} x_j \quad (i,j) \in \mathcal{G} \tag{1b}$$

$$x_k \in \mathcal{X}_k \ , \quad k = 1, \ldots, n \ . \tag{1c}$$

Here, $x_k \in \mathbb{R}^{m_k}$ summarizes the optimization variables of subproblem k, $H_k \succ 0$ is assumed to be positive definite, $\mathcal{X}_k := \{y \in \mathbb{R}^{m_k} : C_k y \leq d_k\}$ is a polyhedral set and matrices $A_{i,j} \in \mathbb{R}^{v_{i,j} \times m_i}$ and $B_{i,j} \in \mathbb{R}^{v_{i,j} \times m_j}$ couple variables x_i and x_j from subsystem i and j, respectively. The topology of the coupled subproblems is described by the graph $\mathcal{G} = (V, E)$ with vertex set $V = \{1, \ldots, n\}$ and edge set E.

Problems of form (1) arise typically in distributed control problems. Each subsystem aims for optimal operation (e.g., setpoint tracking) while respecting local physical operating conditions. The different subsystems may share some state or control variables via the coupling constraints. Distributed control problems with coupled cost function can be reformulated to (1) by introducing local copies of the shared variables, which are forced to be equal by a coupling constraint. Similarly, coupled state or control input constraints can be reformulated as local ones. Problems of form (1) particularly arise in linear and nonlinear model predictive control (MPC) after parametrization in time with *Bock's direct multiple shooting method* [4] as well as after parametrization in both time and space domains with the *distributed multiple shooting method* [21]. In the nonlinear case all subproblems arising in the SQP framework are of form (1).

A. Kozma, J. Frasch, and M. Diehl are with Department of Electrical Engineering (ESAT/SCD), KU Leuven, 3001, Leuven, Belgium {attila.kozma, janick.frasch, moritz.diehl}@esat.kuleuven.be

Our aim is to develop a distributed optimization algorithm for solving such strictly convex QP problems with separable structure. In distributed optimization, several computation nodes solve local problems, in our case n, while exchanging information locally or globally. There is a well known trade-off between the level of distribution and the convergence speed. We also keep in mind that the communication between subproblems requires resources. Thus, we propose an approach that converges to the optimal solution, which is unique due to convexity, while using minimal communication efforts. While designing the proposed method, we pay special attention to avoid forming large matrices.

The proposed algorithm is based on dual decomposition [11], which has already been succesfully employed in several areas such as in optimal control [14], [7], [1], [19], in estimation problems [20] or for problems from telecommunication [23], [5]. These methods typically utilize gradient or fast gradient methods to find the optimal dual variables. They exploit the fact that once the primal problem is strictly convex, the dual problem is concave and continuously differentiable, however, not twice differentiable (see, e.g., [9]). We propose to exploit curvature information in addition to the gradient in the dual decomposition framework for an improved convergence speed. For this purpose, we use a non-smooth inexact Newton's method in the dual space. Our method is inspired by [13], where the authors propose to use a Newton method to solve a convex quadratic spline reformulation of convex quadratic programs. In this work however, we regard QPs with generic coupling topology, and we use the conjugate gradient method to solve the Newton system.

The remainder of this paper proceeds as follows. In Section II, we introduce the concept of dual decomposition and detail the non-smooth Newton method together with a line-search procedure. Section III is devoted to the solution of the underlying linear systems; we discuss the conjugate gradient method and give an algorithmic description of how the gradients and Hessians can be computed addressing communication issues. In Section IV, we apply the proposed method to an example of a chain of masses connected by springs and compare its numerical performance with other state-of-the-art distributed QP methods. The conclusions of the paper are given in Section V.

II. Dual decomposition with second derivatives - The main algorithm in a nutshell

A. Dual decomposition

We introduce dual variables $\lambda_{i,j} \in \mathbb{R}^{v_{i,j}}$ for all $(i,j) \in \mathcal{G}$ corresponding to constraint (1b). We define the Lagrangian function as

$$\mathcal{L}(x,\lambda) := \sum_{k=1}^{n} \left(\frac{1}{2} x_k^T H_k x_k + c_k^T x_k \right) + \sum_{(i,j)\in\mathcal{G}} \lambda_{i,j}^T \left(A_{i,j} x_i - B_{i,j} x_j \right). \qquad (2)$$

Here, $x \in \mathbb{R}^m,\ x^T := [x_1^T, \dots, x_n^T],\ m := \sum_{k=1}^n m_k$, while $\lambda \in \mathbb{R}^v,\ \lambda^T = [\lambda_{i_1,j_1}^T, \dots, \lambda_{i_Q,j_Q}^T]$, with $v := \sum_{(i,j)\in\mathcal{G}} v_{i,j}$ and $(i_1,j_1), \dots, (i_Q,j_Q)$ is the enumeration of edges in V following a fixed order. Let $\mathrm{pre}(i)$, $\mathrm{suc}(i)$ denote the set of predecessor and successor vertices of subproblem i, respectively, i.e.,

$$\mathrm{pre}(i) := \{ j \in V | (j,i) \in \mathcal{G} \} \qquad (3a)$$

$$\mathrm{suc}(i) := \{ j \in V | (i,j) \in \mathcal{G} \}. \qquad (3b)$$

Furthermore, let us denote the set of edges including vertex i by $\mathcal{N}_i,\ i \in V$, i.e.,

$$\mathcal{N}_i := \{ (k,i) | k \in \mathrm{pre}(i) \} \cup \{ (i,j) | j \in \mathrm{suc}(i) \}, \qquad (4)$$

and let $\lambda_{\mathcal{N}_i},\ i \in V$ denote the dual variables that have effect on subproblem i, i.e.,

$$\lambda_{\mathcal{N}_i} := \mathrm{vec}(\{ \lambda_{k,j} | (k,j) \in \mathcal{N}_i \}), \qquad (5)$$

Note that $\mathcal{L}(x,\lambda)$ is separable in x as

$$\mathcal{L}(x,\lambda) = \sum_{i=1}^{n} \mathcal{L}_i(x_i, \lambda_{\mathcal{N}_i}) \qquad (6)$$

with

$$\mathcal{L}_i(x_i, \lambda_{\mathcal{N}_i}) := \frac{1}{2} x_i^T H_i x_i + c_i^T x_i + \sum_{j\in\mathrm{suc}(i)} \lambda_{i,j}^T (A_{i,j} x_i) - \sum_{j\in\mathrm{pre}(i)} \lambda_{j,i}^T B_{j,i} x_i. \qquad (7)$$

As a consequence, one can evaluate the dual function $d(\lambda) := -\min_{x\in\mathcal{X}} \mathcal{L}(x,\lambda)$ on n nodes concurrently as

$$d(\lambda) = -\sum_{i=1}^{n} \min_{x_i \in \mathcal{X}_i} \mathcal{L}_i(x_i, \lambda_{\mathcal{N}_i}). \qquad (8)$$

Note that since (1) is strictly convex, $d(\lambda)$ is convex and continuously differentiable, but not twice differentiable. However, the second derivative exists piecewise [13].

B. Nonsmooth Newton method in the dual space

The optimal dual variables of (1b) can be found as the solution of the unconstrained dual optimization problem

$$\min_{\lambda} d(\lambda). \qquad (9)$$

Several methods [14], [7], [1], [19], [20], [23], [5] tackle this problem with gradient based methods, thus not using second derivatives. In [2], the authors propose a quasi-Newton method to solve a non-smooth optimization problem, but in a different context. Our method is inspired by [13] and proposes to use curvature information to solve (9). The proposed approach consists of two essential ingredients. A *non-smooth Newton method* [18], [10] minimizes the approximatation of $d(\lambda)$ in the actual iterate $\lambda^{(k)}$ using the gradient and Hessian information. The linear Newton system is *solved inexactly* by a conjugate gradient (CG) method [22].

In each iteration of the non-smooth Newton method, a second-order model of $d(\lambda)$ in $\lambda^{(k)}$ is minimized that is

$$m(p) = d(\lambda^{(k)}) + \nabla d(\lambda^{(k)})^T p + \frac{1}{2} p^T \nabla^2 d(\lambda^{(k)}) p. \qquad (10)$$

Solving $p^{(k)} := \arg\min_p m(p)$ yields a descent direction in the space of λ and can be obtained as the solution of the linear system

$$\nabla^2 d(\lambda^{(k)}) p + \nabla d(\lambda^{(k)}) = 0. \qquad (11)$$

In [9], the authors propose a very similar method that solves optimal control problems obtained by Bock's direct multiple shooting method. If the connection graph has a linear topology, i.e., the actual dual Hessian $\nabla d(\lambda)^2$ has block banded structure, (11) can be efficiently solved by direct factorization. The approach of this paper addresses more generic problems, where $\mathcal{G}$ is arbitrary. We never form this Newton system, but only rely on matrix-vector products required by the CG method. A direct factorization is not tractable due to memory limitations, since $\nabla^2 d(\lambda)$ may have blocks at very different locations. Moreover, transmission of matrices through the communication network is costly.

C. Line search strategy

Once having obtained a search direction $p^{(k)}$, i.e., an approximate solution of (11), one can correct the dual variables as

$$\lambda^{(k+1)} := \lambda^{(k)} + t^{(k)} p^{(k)}, \qquad (12)$$

where $t^{(k)}$ is an appropriately chosen stepsize. As it is will be shown in Section III, $p^{(k)}$, the result of the CG method, is a descent direction and is in between the steepest descent and the Newton direction. In order to attain a proper stepsize $t^{(k)}$, we use an Armijo line search strategy with backtracking. In each iteration of (12), we would like to find a $t^{(k)}$ such that

$$d(\lambda^{(k)} + t^{(k)} p^{(k)}) \le d(\lambda^{(k)}) + \gamma t^{(k)} \nabla d(\lambda^{(k)})^T p^{(k)} \qquad (13)$$

holds. We summarize the approach in Algorithm 1, where the comments refer to the communication costs detailed in the next section.

III. Inexact Newton-CG method

In this section, we describe the CG method applied to (11) to obtain a descent search direction. Then we give a specific description of the operations required by the CG method, in particular of how to calculate the necessary matrix-vector products, residuals, etc. in a distributed fashion. We also discuss the communication aspects of the proposed method.

Algorithm 1: Line search procedure

```
   Input : $\lambda^{(k)}$, $t_0$, $p^{(k)}$, $\beta \in (0,1)$, $\gamma \in (0, \frac{1}{2})$
1  $d := d(\lambda^{(k)})$                          // global sum
2  $v := \nabla d(\lambda^{(k)})^T p^{(k)}$          // global sum
3  $t := t_0$
4  while True do
5      $\lambda^+ := \lambda^{(k)} + tp^{(k)}$        // local update
6      $d^+ := d(\lambda^+)$                        // global sum
7      if $d^+ \le d + \gamma t v$ then break
8      else $t := \beta \cdot t$
9  end
10 $t^{(k)} := t$
11 return $t^{(k)}$
```

A. Conjugate gradient method

For the sake of simplicity, in the following we denote the Newton system by $Ax = b$ with $A \succeq 0$. We describe a modified version of the CG algorithm in Algorithm 2. The modifications are necessary to treat singular directions of A and to detect whether a restart is necessary due to rounding errors.

Algorithm 2: Conjugate Gradient method

```
   Input : $A$, $x_0$, $b$, $k_{\max}$, $\epsilon_1$, $\epsilon_2$, $\epsilon_3$
1  $r_0 := Ax_0 - b$, $p_0 := -r_0$, $k := 0$
2  for $k = 0, \dots, k_{\max}$ do
3      $s_k := Ap_k$                                // local comm
4      $\begin{bmatrix}\nu_k \\ \mu_k \\ \tau_k\end{bmatrix} := \begin{bmatrix}p_k^T s_k \\ r_k^T s_k \\ r_k^T r_k\end{bmatrix}$    // global sum
5      if $\nu_k < \epsilon_1$ then
6          if $k = 0$ then return $p_0$
7          else return $x_{k-1}$
8      end
9      $\alpha_k := \frac{\tau_k}{\nu_k}$              // local update
10     $x_{k+1} := x_k + \alpha_k p_k$               // local update
11     $r_{k+1} := r_k + \alpha_k s_k$               // local update
12     if $\frac{\tau_k + \alpha_k \mu_k}{\tau_k} > \epsilon_2$ then
13         $\beta_{k+1} := 0$
14     else
15         $\beta_{k+1} := \frac{r_{k+1}^T r_{k+1}}{\tau_k}$    // global sum
16     end
17     $p_{k+1} := -r_{k+1} + \beta_{k+1} p_k$       // local update
18     if $\sqrt{\tau_k} < \epsilon_3$ then return $x_{k+1}$
19     $k := k + 1$
20 end
21 return $x_k$
```

First, since $A \succeq 0$ we might hit singular directions during the CG iterations, thus we check the definitness of A with respect to p_k in Step 5 and if this gets close to zero, we proceed with a steepest descent step, see Step 6, or with the most recent approximate solution, see Step 7. Second, due to the accumulated rounding and cancellation errors one might encounter non-orthogonal residuals, i.e., $r_{k+1}^T r_k \neq 0$. If such a situation occurs, see Step 12, β_k should be set to zero and a steepest descent step should be performed. Note that Algorithm 2 in the worst case returns with the steepest descent direction $p^{(k)} = -\nabla d(\lambda)$ and the iteration (12) boils down to a steepest descent step. In the best case, the CG subroutine manages to find the Newton direction $p^{(k)} = -\left(\nabla^2 d(\lambda^{(k)})\right)^{-1} \nabla d(\lambda^{(k)})$. If the maximal number of CG iterations $k_{\max}$ is reached and the residual r_k is not yet close to zero the returned search direction $p^{(k)}$ can be interpreted as an inexact Newton direction that improves the pure steepest descent direction. In any case, a descent direction is obtained. For this reason, the theoretical worst-case complexity is is not expected to exceed the one of a gradient method, namely sublinear convergence rate. However, we expect our method to perform better than a plain gradient method in practise. Again, the comments refer to the communication costs analyzed in the following subsection. We have added comments to several steps to give a hint about the nature of these operations, which are discussed in more detail in the following subsection.

B. Calculation of derivatives in a distributed manner

As we have seen in Algorithm 2, we need to calculate the gradient $\nabla d(\lambda)$, the Hessian times vector product $\nabla^2 d(\lambda)p$, the quadratic form $p^T \nabla^2 d(\lambda) p$ and the residual of (11). We now discuss how to calculate these in a distributed manner.

As we have mentioned earlier, the dual function $d(\lambda)$ is continuously differentiable [3, p. 100] and its derivative in the direction of $\lambda_{i,j}$ is given by

$$\frac{\partial d(\lambda)}{\partial \lambda_{i,j}}^T = -A_{i,j} x_i^*(\lambda_{\mathcal{N}_i}) + B_{i,j} x_j^*(\lambda_{\mathcal{N}_j}) \qquad (14)$$

where $x_i^*(\lambda_{\mathcal{N}_i}) := \arg\min_{x_i \in \mathcal{X}_i} \mathcal{L}_i(x_i, \lambda_{\mathcal{N}_i})$. It should be understood that only subproblems i and j need to be solved in order to calculate the dual gradient in the direction of $\lambda_{i,j}$. Moreover, this slice of the gradient can be computed in subproblems i and j by a simple local exchange of contributions. This is one of the reasons why first order methods are often used in distributed optimization algorithms.

Although the dual function $d(\lambda)$ is not twice differentiable everywhere, the second derivative can be given as a piece-wise quadratic function. Indeed, depending on λ, different constraints in (8) may get active or inactive and to each active set corresponds a well-defined Hessian $\nabla^2 d(\lambda)$. Let us show that the dual function is indeed piece-wise quadratic. The proof is constructive since it provides an algorithmic procedure to calculate the dual Hessian in the actual iterate. We assume that there exists an $\epsilon > 0$ neighbourhood of a fixed λ in which the set of active constraints does not change. Denote this set of constraints by $\tilde{C}_i x_i = \tilde{d}_i$. Now, $\min_{x_i \in \mathcal{X}_i} \mathcal{L}_i(x_i, \lambda_{\mathcal{N}_i})$ simplifies to an equality constrained convex QP of the form

$$\min_{x_i} \frac{1}{2}x_i^T H_i x_i + c_i^T x_i + \sum_{j\in \mathrm{suc}(i)} \lambda_{i,j}^T (A_{i,j}x_i) - \sum_{j\in \mathrm{pre}(i)} \lambda_{j,i}^T B_{j,i} x_i \tag{15a}$$

$$\text{s.t. } \tilde{C}_i x_i = \tilde{d}_i. \tag{15b}$$

One can eliminate the equality constraints (15b) and express $x_i = E_i\tilde{x}_i + g_i$ with some matrix E_i and vectors $\tilde{x}_i$ and g_i, see [16, p. 426] for details. The resulting equivalent QP is unconstrained, lower dimensional and is of the form

$$\min_{\tilde{x}_i} \frac{1}{2}\tilde{x}_i^T E_i^T H_i E_i \tilde{x}_i + \Bigg(g_i^T H_i + c_i^T + \sum_{j\in \mathrm{suc}(i)} \lambda_{i,j}^T A_{i,j} + \sum_{j\in \mathrm{pre}(i)} \lambda_{j,i}^T B_{j,i} \Bigg) E_i \tilde{x}_i + \Bigg(c_i^T + \frac{1}{2} g_i^T H_i + \sum_{j\in \mathrm{suc}(i)} \lambda_{i,j}^T A_{i,j} + \sum_{j\in \mathrm{pre}(i)} \lambda_{j,i}^T B_{j,i} \Bigg) g_i. \tag{16}$$

Its optimal solution $\tilde{x}_i^*$ is

$$\tilde{x}_i^* = -\left(E_i^T H_i E_i\right)^{-1} \left[\Bigg(g_i^T H_i + c_i^T + \sum_{j\in \mathrm{succ}(i)} \lambda_{i,j}^T A_{i,j} + \sum_{j\in \mathrm{pre}(i)} \lambda_{j,i}^T B_{j,i} \Bigg) E_i \right]^T \tag{17}$$

If we plug $\tilde{x}_i^*$ into (16), we get a quadratic function of the form

$$\min_{x_i\in\mathcal{X}_i} \mathcal{L}_i(x_i, \lambda_{\mathcal{N}_i}) = -\frac{1}{2}\left[(g_i^T H_i + c_i^T + q_i^T)E_i\right] Q_i \left[(g_i^T H_i + c_i^T + q_i^T)E_i\right]^T + \left(c_i^T + \frac{1}{2} g_i^T H_i + q_i^T\right) g_i, \tag{18a}$$

with

$$Q_i = (E_i^T H_i E_i)^{-1} \tag{18b}$$

$$q_i^T = \sum_{j\in \mathrm{succ}(i)} \lambda_{i,j}^T A_{i,j} + \sum_{j\in \mathrm{pre}(i)} \lambda_{j,i}^T B_{j,i}. \tag{18c}$$

Indeed, via this characterization we can conclude that $d(\lambda)$ is a piece-wise quadratic function and it has pure quadratic nature once the active set in the subproblems does not change. One possible way to calculate the dual Hessian $\nabla^2 d(\lambda)$ for a fixed λ is to use the result of (18). This approach necessitates that the underlying QP-solver solving $\min_{x_i\in\mathcal{X}_i} \mathcal{L}_i(x_i, \lambda_{\mathcal{N}_i})$ provides a basis of the nullspace of the active constraints, i.e., matrix E_i and vector g_i. Obtaining E_i and g_i is straightforward with no extra cost when using a QP solver based on an active-set strategy of nullspace type, such as [8]. In this case, the cost of calculating the dual Hessian is dominated by the expense of a local matrix inversion. Often, the direct factorization of $E_i^T H_i E_i$ is also available, which may make its inversion more efficient.

If matrix E_i and vector g_i are not available, for instance when using a black-box QP-solver, one can compute the dual Hessian with solution sensitivity information. We obtain $\nabla^2 d(\lambda)$ via differentiating (14). With this approach, it is important to observe that the dual Hessian has a well-defined sparsity structure, which is essentially determined by the interconnections of subsystems. The block corresponding to variables $\lambda_{i,j},\ (i,j)\in\mathcal{G}$ can be written as

$$\frac{\partial^2 d(\lambda)}{\partial\lambda_{i,j}\partial\lambda_{i,j}} = -A_{i,j}\frac{\partial x_i^*(\lambda_{\mathcal{N}_i})}{\partial\lambda_{i,j}} + B_{i,j}\frac{\partial x_j^*(\lambda_{\mathcal{N}_j})}{\partial\lambda_{i,j}}. \tag{19}$$

The off-diagonal blocks corresponding to the row of $\lambda_{i,j}$ and the column of $\lambda_{k,l},\ (i,j)\neq(k,l)$ are given by

$$\frac{\partial^2 d(\lambda)}{\partial\lambda_{i,j}\partial\lambda_{k,l}} = \begin{cases} -A_{i,j}\frac{\partial x_i^*(\lambda_{\mathcal{N}_i})}{\partial\lambda_{k,l}} & \text{if}(k,l)\in\mathcal{N}_i\setminus\mathcal{N}_j \\ B_{i,j}\frac{\partial x_j^*(\lambda_{\mathcal{N}_j})}{\partial\lambda_{k,l}} & \text{if}(k,l)\in\mathcal{N}_j\setminus\mathcal{N}_i \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

In other words, the second derivatives with respect to $\lambda_{i,j}$ and $\lambda_{k,l}$ are nonzero if and only if edge (k,l) is connected to node i or j.

In Algorithm 2, we need to calculate the quantities $\nabla^2 d(\lambda)p$ and $p^T\nabla^2 d(\lambda)p$. We discuss how to do this in a distributed fashion.

One can calculate the Hessian times a vector p product in slices as

$$\frac{\partial^2 d(\lambda)}{\partial\lambda_{i,j}\partial\lambda}p = \left(-A_{i,j}\frac{\partial x_i^*(\lambda_{\mathcal{N}_i})}{\partial\lambda_{i,j}} + B_{i,j}\frac{\partial x_j^*(\lambda_{\mathcal{N}_j})}{\partial\lambda_{i,j}}\right)p_{i,j} + \sum_{(k,l)\in\mathcal{N}_i} -A_{i,j}\frac{\partial x_i^*(\lambda_{\mathcal{N}_i})}{\partial\lambda_{k,l}}p_{k,l} + \sum_{(k,l)\in\mathcal{N}_j} B_{i,j}\frac{\partial x_j^*(\lambda_{\mathcal{N}_j})}{\partial\lambda_{k,l}}p_{k,l}. \tag{21}$$

Here, $p\in\mathbb{R}^l,\ p^T = [p_{i_1,j_1}^T,\ldots,p_{i_Q,j_Q}^T]$ and the order $(i_1,j_1),\ldots,(i_Q,j_Q)$ corresponds to the one defined in λ. It should be noted that the vector slice $\frac{\partial^2 d(\lambda)}{\partial\lambda_{i,j}\partial\lambda}p$ can be calculated on both nodes i and j via only local exchange of contributions to the sum in (21). We refer to Step 3 of Algorithm 2, where the matrix-vector product is calculated only in slices.

We use the result in (21) in order to calculate the quadratic form $p^T\frac{\partial^2 d(\lambda)}{\partial\lambda\partial\lambda}p$ as

$$p^T\frac{\partial^2 d(\lambda)}{\partial\lambda\partial\lambda}p = \sum_{(i,j)\in\mathcal{G}} p_{i,j}^T \frac{\partial^2 d(\lambda)}{\partial\lambda_{i,j}\partial\lambda}p = = \sum_{(i,j)\in\mathcal{G}} p_{i,j}^T \Bigg[\left(-A_{i,j}\frac{\partial x_i^*(\lambda_{\mathcal{N}_i})}{\partial\lambda_{i,j}} + B_{i,j}\frac{\partial x_j^*(\lambda_{\mathcal{N}_j})}{\partial\lambda_{i,j}}\right)p_{i,j} + \sum_{(k,l)\in\mathcal{N}_i} -A_{i,j}\frac{\partial x_i^*(\lambda_{\mathcal{N}_i})}{\partial\lambda_{k,l}}p_{k,l} + \sum_{(k,l)\in\mathcal{N}_j} B_{i,j}\frac{\partial x_j^*(\lambda_{\mathcal{N}_j})}{\partial\lambda_{k,l}}p_{k,l} \Bigg] \tag{22}$$

Note that in (22) there is a summation over all $(i,j) \in \mathcal{G}$. Each term of this sum can be calculated via local communication as we have seen earlier. However, the calculation of the sum itself needs a global operation, such that all local contributions are collected and the result is broadcasted. This operation is often referred to as *global summation* [12]. Based upon the previous discussion the residual $\nabla^2 d(\lambda)p + \nabla d(\lambda)$ is computable in slices with respect to $\lambda_{i,j}$ and is available in subproblems i and j.

One inexact Newton iteration consist of a distributed derivative computation step, a CG procedure, and a line search routine. In the derivative computation step, a local QP solution takes places on each node to obtain local contribution to the dual gradient. The computation of the local dual Hessian contribution requires an inversion of the reduced Hessian. No communication is needed at this point. Next, the Newton system is solved by several CG iterations, each of which involves an exchange of float vectors between direct neighbours, and two global summation operation. Once a search direction is obtained, the line search loop follows. In the beginning, a global summation of two numbers takes place. Furthermore, each line search trial requires one local QP solution, and a global summation. The gradient and Hessian obtained in the last trial can be used in the following Newton iteration.

The proposed method works well if the controlled system features a reasonable fast global communication structure (e.g., ethernet, internet or faster). Limitations of this approach are a sequential communication structure in systems of huge diameter (in a graph theoretical sense), when only neighbor-to-neighbor communication is possible.

IV. Numerical example

In this section, we demonstrate the performance of the method discussed in the previous section on an academic example.

We consider a linear model for a chain of connected masses in one dimension. Each mass $n \in \{1,\dots,N\}$ is described by its position $p_n \in \mathbb{R}$ and its velocity $v_n \in \mathbb{R}$ and may be controlled via a force F_n. For the sake of simplicity, we directly present the equations obtained by using distributed multiple shooting method, i.e., after spatial and time discretization. We discretize in time by one step of an explicit Euler method. Let us denote the length of time intervals and the number of time intervals by Δt and M, respectively. The continuity of the state profile is ensured by the constraint

$$\begin{bmatrix} p_n^{(m+1)} \\ v_n^{(m+1)} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 \\ -2\Delta t & 1 & \Delta t & \Delta t & \Delta t \end{bmatrix} \begin{bmatrix} p_n^{(m)} \\ v_n^{(m)} \\ z_{n,n-1}^{(m)} \\ z_{n,n+1}^{(m)} \\ F_n^{(m)} \end{bmatrix}. \tag{23}$$

Here, $m = 1,\dots,M-1$, $n = 1,\dots,N$ and the variables $z_{n,n-1}^{(m)}$ and $z_{n,n+1}^{(m)}$ belong to the masspoint n on time interval m and represent a finite discretization, e.g. polynomial coefficients, of the positions of masspoint $n-1$ and $n+1$, respectively. The spatial coupling between the mass points is ensured by

$$z_{n,n+1}^{(m)} = p_{n+1}^{(m)} \quad n = 1,\dots,N-1 \tag{24}$$

$$z_{n,n-1}^{(m)} = p_{n-1}^{(m)} \quad n = 2,\dots,N \tag{25}$$

The objective function penalizes the deviation from the steady state 0 in an L_2-norm. The bounds on the actuated forces are given by

$$\underline{F} \le F_n^{(m)} \le \overline{F}, \quad m = 1,\dots,M, \; n = 1,\dots,N, \tag{26}$$

and the first and last mass points are fixed

$$p_1^m = p_{\text{first}}, \; p_n^m = p_{\text{last}}, \; m = 1,\dots,M \tag{27}$$

We inroduce for all $n = 1,\dots,N$, and $m = 1,\dots,M$

$$y_{(m-1)N+n}^T := (p_n^{(m)}, v_n^{(m)}, z_{n,n-1}^{(m)}, z_{n,n+1}^{(m)}, F_n^{(m)})^T, \tag{28}$$

$S := NM$ and summarize the discretized optimal control problem as a convex QP of the form

$$\min_{y_1,\dots y_S} \frac{1}{2}\sum_{k=1}^{S} y_k^T R_k y_k \tag{29a}$$

$$\text{s.t } C_{i,j} y_i = D_{i,j} y_j, \; (i,j) \in \overline{\mathcal{G}} \tag{29b}$$

$$y_l \in \mathcal{Y}_l, \; l = 1,\dots,S. \tag{29c}$$

Here, $(i,j) \in \overline{\mathcal{G}}$ if and only if $j = i+N$ or ($j = i+1$ and $j \mod N \neq 0$). Moreover, $C_{i,j}$ and $D_{i,j}$ are directly computable from (23)-(25) and $\mathcal{Y}_l$ can be derived from (26). Note that (29) has the same structure as (1) and thus the proposed algorithm of this paper can applied.

We solved an instance of (29) with $m = 20$ mass points and $n = 15$ time intervals resulting in a decomposable QP with 321 subproblems. The full QP has a total of 1540 primal variables, 1170 equalities, which is the dimension of the dual space as well, and 2340 inequalities. As a stopping criterion, we used $\frac{\|\nabla d(\lambda^k)\|}{\|\nabla d(\lambda^0)\|} < \epsilon$ to measure the infeasibility of the primal problem. To obtain a solution with an accuracy $\epsilon = 10^{-6}$, the proposed approach took 67 inexact Newton iterations, see Figure 1. Throughout the iterations, 264 line search tries were carried out, which is the number of necessary local QP solutions using a white-box local QP solver. In each Newton iteration, the maximal number of CG iterations was limited by 120. Altogether, 5812 CG iterations were necessary. We note that in our prototype implementation finite differences were used to calculate the local dual Hessian blocks.

For comparison, we solved the same problem instance with a fast gradient method [15, p. 80] applied in the dual space to solve (9). A solution with 10^{-4} accuracy was obtained after 16007 iterations, thus the same number of QP solutions were necessary on each node plus the local communication expenses. We also solved this problem with the restarted variant of the fast gradient method [17], which according to our experience is one of the most efficient first order methods for dual decomposition. This method required 13185 steps

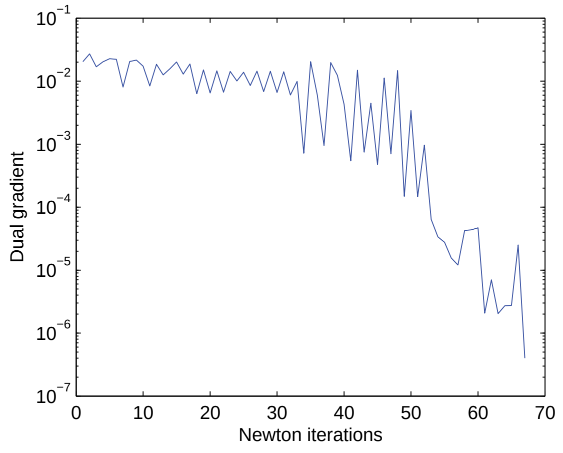

Fig. 1. The decrease of the dual gradient norm throughout the inexact Newton iterations.

in the dual space (i.e., sequential local QP solutions) to find multipliers up to 10^{-4} accuracy.

V. Conclusions and future work

In this paper, we presented a novel method for the solution of strictly convex quadratic programs with decomposable structure using dual decomposition. In each iteration, matrix-vector products are communicated locally, while only scalars need to be summed up globally. Preliminary experiments with an optimal control problem of a chain of masses show that using both first and second derivatives in the dual space may lead to a significant speed-up compared to purely gradient based methods. Exploiting second order information comes at little extra cost when using an appropriate white-box QP solver.

Establishing a proof of local as well as of global convergence properties is subject of ongoing research. The generalization of the method to convex nonlinear problems is also desired.

Acknowledgments

This work received funding from the EU Seventh Framework Programme FP7/2007-2013 under grant agreement n^o FP7-ICT-2009-4 248940. This research was supported by Research Council KUL: PFV/10/002 Optimization in Engineering Center OPTEC, GOA/10/09 MaNet and GOA/10/11 Global real-time optimal control of autonomous robots and mechatronic systems. Flemish Government: IOF/KP/SCORES4CHEM, FWO: PhD/postdoc grants and projects: G.0320.08 (convex MPC), G.0377.09 (Mechatronics MPC); IWT: PhD Grants, projects: SBO LeCoPro; Belgian Federal Science Policy Office: IUAP P7 (DYSCO, Dynamical systems, control and optimization, 2012-2017); EU: FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC ST HIGHWIND (259 166), Eurostars SMART, ACCM.

References

[1] A.G. Beccuti and M. Morari. A distributed solution approach to centralized emergency voltage control. In *American Control Conference, 2006*, page 6, june 2006.

[2] Stephen Becker and M Jalal Fadili. A quasi-newton proximal splitting method. *arXiv preprint arXiv:1206.1156*, 2012.

[3] D.P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: Numerical methods*. Prentice Hall, 1989.

[4] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 242–247. Pergamon Press, 1984.

[5] R. Cendrillon, Wei Yu, M. Moonen, J. Verlinden, and T. Bostoen. Optimal multiuser spectrum balancing for digital subscriber lines. *Communications, IEEE Transactions on*, 54(5):922 – 933, may 2006.

[6] M. Diehl, H.G. Bock, and J.P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.

[7] M. Fardad, Fu Lin, and M.R. Jovanovic. On the dual decomposition of linear quadratic optimal control problems for vehicular formations. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 6287 –6292, dec. 2010.

[8] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 2013. (under review).

[9] H.J. Ferreau, A. Kozma, and M. Diehl. A parallel active-set strategy to solve sparse parametric quadratic programs arising in MPC. In *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference, Noordwijkerhout, The Netherlands*, 2012.

[10] Matthias Gerdts and Martin Kunkel. A nonsmooth newton's method for discretized optimal control problems with state and control constraints. *Journal of Industrial and Management Optimization*, 4:247–270, 2008.

[11] C. Lemaréchal. Lagrangian relaxation. In Michael Jünger and Denis Naddef, editors, *Computational Combinatorial Optimization*, volume 2241 of *Lecture Notes in Computer Science*, pages 112–156. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-45586-8_4.

[12] J.G. Lewis and R.A. van de Geijn. Distributed memory matrix-vector multiplication and conjugate gradient algorithms. In *Supercomputing '93. Proceedings*, pages 484 – 492, nov. 1993.

[13] W. Li and J. Swetits. A new algorithm for solving strictly convex quadratic programs. *SIAM Journal of Optimization*, 7(3):595–619, 1997.

[14] I. Necoara, D. Doan, and J.A.K. Suykens. Application of the proximal center decomposition method to distributed model predictive control. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 2900 –2905, dec. 2008.

[15] Y. Nesterov. *Introductory lectures on convex optimization: a basic course*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, 2004.

[16] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.

[17] B. O'Donoghue and E. Candes. Adaptive Restart for Accelerated Gradient Schemes. (submitted for publication), April 2012.

[18] Liqun Qi and Jie Sun. A nonsmooth version of newton's method. *Mathematical Programming*, 58:353–367, 1993.

[19] S. Richter, M. Morari, and C.N. Jones. Towards computational complexity certification for constrained mpc based on lagrange relaxation and the fast gradient method. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 5223 –5229, dec. 2011.

[20] S. Samar, S. Boyd, and D. Gorinevsky. Distributed estimation via dual decomposition. In *Proceedings European Control Conference (ECC)*, pages 1511–1516, Kos, Greece, 2007.

[21] C. Savorgnan, C. Romani, A. Kozma, and M. Diehl. Multiple shooting for distributed systems with applications in hydro electricity production. *Journal of Process Control*, 21:738–745, 2011.

[22] Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA, 1994.

[23] P. Tsiaflakis, I. Necoara, J.A.K. Suykens, and M. Moonen. Improved Dual Decomposition Based Optimization for DSL Dynamic Spectrum Management. *IEEE Transactions on Signal Processing*, 58(4):2230–2245, 2010.