**Abstract:**
The performance of fast forward-backward splitting, or equivalently fast proximal gradient methods, is susceptible to conditioning of the optimization problem data. This conditioning is related to a metric that is defined by the space on which the optimization problem is stated. By selecting a space on which the optimization data is more well-conditioned, the performance of the fast forward-backward splitting method is likely to improve. In this paper, we propose several methods, with different computational complexity, to find a space on which the algorithm performs well when applied to the dual of a strongly convex optimization problem. We evaluate the proposed metric selection procedures by comparing the performance to when the fast dual forward-backward splitting method is applied in the Euclidean space. For the most ill-conditioned problem we consider, the computational complexity is improved by two to three orders of magnitude.

1. **INTRODUCTION**

Fast gradient methods have been around since the early 80’s when the seminal paper Nesterov (1983) was published. The fast gradient method in Nesterov (1983) is applicable to unconstrained smooth optimization problems in the Euclidean space. This method was not well recognized until the mid 00’s, after which several extensions and generalizations of the fast gradient method have been proposed. In Nesterov (2003), new acceleration schemes where presented as well as methods for constrained optimization, i.e. projected fast gradient methods. In Nesterov (2005), fast gradient methods were applied to problems defined on general real finite-dimensional linear normed vector-spaces with an inner product, and smoothing techniques were presented. A fast proximal gradient method, or equivalently a fast forward-backward splitting method, was presented in Beck and Teboulle (2009). This method is applicable to composite optimization problems consisting of one smooth and one non-smooth part, defined on the Euclidean space. In Tseng (2008), generalizations and unifications of many fast forward-backward splitting methods were presented, e.g. it was shown that the method in Beck and Teboulle (2009) can be applied to composite optimization problems defined on any Hilbert space.

The smooth part, $f$, of the composite objective function is in fast forward-backward splitting approximated by the r.h.s. of

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{\beta}{2} \| x - y \|^2 \tag{1}$$

where the norm and inner-product are given by the space on which $f$ is defined. The condition that (1) holds for all $x$ and $y$ is referred to as $f$ being $\beta$-smooth w.r.t. the space on which $f$ is defined. Since the r.h.s. of the smoothness condition (1) is the only information the algorithm has about the smooth function, the smaller the gap in (1) (i.e. the better the r.h.s. of (1) approximates $f$), the better the performance of the algorithm is likely to be. In this paper, we show how to select a space on which the fast forward-backward splitting method performs well, when solving the dual of strongly convex composite optimization problems.

The spaces we consider when solving these problems are Euclidean spaces with inner product $(x,y) = x^T y$ and a scaled norm $\| x \|_K = \sqrt{x^TKx}$. These spaces are denoted by $E_K$. We also consider real finite-dimensional Hilbert spaces with inner product $(x,y)_K = x^T Ky$ and induced norm $\| x \|_K = \sqrt{(x,x)_K}$. These spaces are denoted by $H_K$. We show that applying fast dual forward-backward splitting on $E_{K^{-1}}$ is equivalent to applying fast dual forward-backward splitting on $H_K$. We also show that the smooth part of the dual problem, which is a composition of the conjugate of the strongly convex function with a linear bounded operator, defined on $E_{K^{-1}}$, is $\beta$-smooth (w.r.t. $E_{K^{-1}}$) if and only if the dual problem defined on $H_K$ is $\beta$-smooth (w.r.t. $H_K$). This verifies that fast dual forward-backward splitting on $E_{K^{-1}}$ and $H_K$ indeed converge under the same assumptions. This implies that the space selection reduces to find a matrix $K$ that defines the spaces $E_{K^{-1}}$ and $H_K$ on which fast forward-backward splitting performs well. We show how to select this $K$ to get the optimal approximation of the form (1) of the smooth part of the dual function. Here, by optimal approximation, we mean that no other space $E_{L^{-1}}$ (or equivalently $H_L$) exists with $L \geq K$ such that (1) holds for all $x,y$. Since the smooth part of the dual function is in fast forward-backward splitting approximated by the

---


corresponding author

Pontus Giselsson * Stephen Boyd *

* Electrical Engineering Department, Stanford University.

Email: {pontusg,boyd}@stanford.edu

* During the preparation of this paper, the first author was a member of the LCCC Linnaeus Center at Lund University. Financial support from the Swedish Research Council for the first author’s Postdoctoral studies at Stanford University is gratefully acknowledged. Further, Alexander Domahidi is gratefully acknowledged for suggesting the AFTI-16 control problem as a benchmark and providing FORCES code for the same. Finally, the authors acknowledge the anonymous reviewers for constructive feedback.
r.h.s. of (1), applying fast dual forward backward splitting on \( E_{K^{-1}} \) (or \( H_K \)), where \( K \) is chosen to get the optimal approximation, is likely to give good iteration counts.

The optimal approximation makes the algorithm as iteration count efficient as possible. This is often achieved when the space \( E_{K^{-1}} \) (or \( H_K \)) is defined by a matrix \( K \) with no specific structure. To get a computationally efficient algorithm, the \( K \)-matrix should have a structure that makes the backward part, or the prox step, of the algorithm efficient. For instance, if the non-smooth part of the dual problem is separable down to the component, we need the \( K \)-matrix to be diagonal to not increase the computational burden when computing the prox-step. We propose several methods to compute a \( K \), with a given structure, that gives both good iteration counts and cheap iteration complexity. Such \( K \) are ideally computed by solving a condition number minimization problem, subject to structural constraints. We show how to solve such problems exactly using semi-definite programming. Condition number minimization is computationally expensive and is therefore only applicable for code-generation for multi-parametric programs or model predictive control (MPC), where considerable computational effort can be spent on optimizing the optimization code offline. We also present several heuristic approximation methods for reducing the condition number that are (much) less computationally expensive than the condition number minimization. These heuristics can, of course, not be expected to give the same performance of the resulting algorithm as if minimizing the condition number.

Recently, Richter et al. (2013); Patrinos and Bemporad (2014) proposed fast dual forward-backward splitting as appropriate optimization algorithms for embedded model predictive control. They propose two different dual formulations and apply fast forward-backward splitting in the Euclidean space on their respective dual problems. Using the results in this paper, we show how to select an appropriate space on which to apply these methods. The performance improvement is evaluated by applying the methods to a pitch control problem in an AFTI-16 aircraft that has previously been studied in Kapasouris et al. (1990); Bemporad et al. (1997). This is a challenging problem for first order methods since it is quite ill-conditioned. The numerical evaluation shows that for such ill-conditioned problems, the computation time can be improved by two to three orders of magnitude by selecting the space on which to apply the algorithm appropriately. We also compare the performance with the algorithm in O’Donoghue et al. (2013); Jerez et al. (2013) that use an ADMM-based (see Boyd et al. (2011) for more on ADMM – the alternating direction method of multipliers) algorithm to solve the embedded model predictive control optimization problems. Further, we compare to FORCES, Domahidi et al. (2012), which is a C code-generator for MPC problems using a tailored interior point method, and to the general commercial QP-solver MOSEK. The fast dual forward-backward splitting when applied on a suitable space performs little to much better than all other considered methods on the considered example.

Fast dual forward-backward splitting can also be used for distributed optimization when the objective to be minimized is separable. In the context of gradient methods, this has been known since Everett (1963); Danzig and Wolfe (1961); Benders (1962). Recently such approaches have been proposed for distributed model predictive control (DMPC) Negenborn (2007); Doan et al. (2011); Giselsson et al. (2013); Giselsson (2013), and resource optimization over networks Ghadimi et al. (2013); Beck et al. (2013). Often, centralized coordination is needed when selecting the step-size for the gradient-step. In Beck et al. (2013), it was noted that the smooth part of the dual problem consists of a sum of local dual terms. Each of these can compute its own step-size, share with its neighbors, and sum to get a fully distributed step-size selection. This procedure can be augmented by the results of this paper to not only select a local step-size, but local matrices \( K_i \), that, after neighborhood exchange and summation, give a space on which to apply the fast distributed dual forward-backward splitting. When solving a set of randomly generated separable optimization problems, the iteration count of the algorithm when applied on a space as computed by the methods presented in this paper is two to ten times lower than the iteration count when the local step-sizes are computed as in Beck et al. (2013). We also compare the performance to the recently proposed dual Newton conjugate gradient method in Kromka et al. (2014). The proposed methods outperform this method by at least a factor of 10 on the considered examples.

This paper unifies and extends the conference publications Giselsson (2014a,b); Giselsson and Boyd (2014).

2. NOTATION AND PRELIMINARIES

2.1 Notation

We denote by \( \mathbb{R} \), \( \mathbb{R}^n \), \( \mathbb{R}^{n \times n} \), the sets of real numbers, column vectors, and matrices. We use notation \((x, y, z) := [x^T \ y^T \ z^T]^T \) for stacked real column vectors. We also use notation \( \mathbb{R} = \mathbb{R} \cup \{ \infty \} \) for the extended real line. \( S^n \subset \mathbb{R}^{n \times n} \) is the set of symmetric matrices, and \( S^n_+ \subset S^n \), \( S^n_\ast \subset S^n \), are the sets of positive [semi] definite matrices. We consider real finite-dimensional normed vector spaces equipped with inner products. Especially, we consider the Euclidean space \( \mathbb{R}^n \) with inner product \((x, y) = x^T y \) and induced norm \( \| \cdot \| = \sqrt{(\cdot, \cdot)} \). We also consider \( \mathbb{E}_H \) with Euclidean inner product \((\cdot, \cdot)_H \) and with a scaled norm \( \| \cdot \|_H = \sqrt{(\cdot, H \cdot)}_2 \), where \( H \in S^n_+ \). Finally, we consider spaces \( \mathbb{E}_H \) with inner product \((x, y) = (x, H y)_2 = x^T H y \) and induced norm \( \| \cdot \|_H = \sqrt{(\cdot, H \cdot)}_2 \), where again \( H \in S^n_+ \). For clarity, we sometimes use notation \( \| \cdot \|_{\mathbb{E}_H} \) and \( \| \cdot \|_{\mathbb{E}_H} \) to distinguish the norms on \( \mathbb{R}^n \) and \( \mathbb{E}_H \) when needed. If any of these spaces is considered, we denote this by \( \mathbb{X} \) or \( \mathbb{Y} \) and the corresponding inner product and norm are denoted by \((\cdot, \cdot)_{\mathbb{X}} \) and \( \| \cdot \|_{\mathbb{X}} \) respectively. We denote the dual space of \( \mathbb{X} \) by \( \mathbb{X}^* \) and the corresponding dual norm by \( \| \cdot \|_{\mathbb{X}} \). The adjoint operator to \( A \) is denoted by \( A^* \). Further, the class of closed, proper, and convex functions \( f : \mathbb{X} \rightarrow \mathbb{R} \) is denoted by \( \Gamma_0(\mathbb{X}) \). Finally, \( I_X \) denotes the indicator function for the set \( \mathbb{X} = \{ x \mid g(x) \leq 0 \} \).

2.2 Preliminaries

Definition 1. The dual norm is defined as
\[ \|y\| = \max_x \{ \langle y, x \rangle : \|x\| = 1 \}. \]

**Example 2.** The dual norm to \( H \), i.e. the norm on \( H^* \), is given by
\[ \|y\|_{H^*} = \max_x \{ \langle y, x \rangle_H : \|x\|_H = 1 \} = \max_y \{ \langle H^1/2 y, v \rangle_2 : \|v\|_2 = 1 \} = \frac{\|y\|}{\|H\|}. \]

Thus, \( H^*_\mathcal{H} = H \). This holds since \( H \) is a Hilbert space.

The norm on \( E_H \) is given by
\[ \|y\|_{E_H} = \max_x \{ \langle y, x \rangle_{E_H} : \|x\|_{E_H} = 1 \} = \max_y \{ \langle H^{-1/2} y, v \rangle_2 : \|v\|_2 = 1 \} = \frac{\|y\|}{\|H\|}. \]

Thus, \( E_H = E_{H^{-1}} \).

**Definition 3.** For a linear operator \( A : \mathcal{X} \rightarrow \mathcal{Y}^* \), the adjoint operator \( A^* : \mathcal{Y} \rightarrow \mathcal{X}^* \) is defined to satisfy
\[ \langle Ax, y \rangle = \langle A^* y, x \rangle \]
for all \( x \in \mathcal{X} \) and all \( y \in \mathcal{Y} \).

**Example 4.** Let \( A \in \mathbb{R}^{m \times n} \) be a matrix and \( A \in \mathbb{E}_H \rightarrow \mathbb{E}_K^* \) be a bounded linear operator such that \( Ax = Ax \) for all \( x \). Then
\[ \langle Ax, y \rangle_{E_K} = \langle Ax, y \rangle_{E_H} = \langle A^T y, x \rangle_{E_K} = \langle A^* y, x \rangle_{E_H}. \]

That is, \( A^* : \mathbb{E}_K \rightarrow \mathbb{E}_H \) satisfies \( A^* y = A^T y \) for all \( y \) of \( \mathcal{Y} \).

Let \( A \in \mathbb{R}^{m \times n} \) be a matrix and \( A : \mathbb{H}_H \rightarrow \mathbb{H}_K \) be a bounded linear operator such that \( Ax = Ax \) for all \( x \). Then
\[ \langle Ax, y \rangle_{E_K} = \langle KAx, y \rangle_{E_K} = \langle (x, A^T K y)_{E_K} = \langle (H^{-1} A^T K y, x)_{E_K} \]

That is, \( A^* : \mathbb{E}_K \rightarrow \mathbb{E}_H \) satisfies \( A^* y = H^{-1} A^T K y \) for all \( y \) of \( \mathcal{Y} \).

**Definition 5.** The operator norm of \( A : \mathcal{X} \rightarrow \mathcal{Y}^* \) is defined as
\[ \|A\|_{\mathcal{X}, \mathcal{Y}^*} = \max \{ \langle Ax, y \rangle_{\mathcal{Y}^*} : \|x\|_\mathcal{X} = 1, \|y\|_{\mathcal{Y}^*} = 1 \}. \]

**Example 6.** From Definitions 1 and 3, we conclude that the operator norm satisfies
\[ \|A\|_{\mathcal{X}, \mathcal{Y}^*} = \|A^*\|_{\mathcal{Y}^*, \mathcal{X}} = \max_x \{ \|Ax\|_{\mathcal{Y}^*} : \|x\|_\mathcal{X} = 1 \} = \max_y \{ \|A^* y\|_{\mathcal{X}} : \|y\|_{\mathcal{Y}^*} = 1 \}. \]

**Definition 7.** Given \( f : \mathcal{X} \rightarrow \mathbb{R} \), the subdifferential \( \partial f(x) \) at \( x \in \mathcal{X} \) is given by the set of \( u \in \mathcal{X}^* \) such that
\[ f(x) \geq f(y) + \langle u, x - y \rangle. \]

**Definition 8.** For the function \( f : \mathcal{X} \rightarrow \mathbb{R} \) to be everywhere single-valued, then \( f \) is differentiable and we set \( \partial f = \nabla f \) where \( \nabla f \) denotes the gradient.

Since the inner product is part of the subdifferential definition, the subdifferential may differ for different spaces. Next, we state a result that quantifies this difference for the spaces of interest to this paper.

**Proposition 9.** Suppose that \( f_H \in \Gamma_0(\mathbb{H}_H) \) and \( f \in \Gamma_0(\mathbb{E}_H) \) satisfy \( f_H(x) = f(x) \) for all \( x \in \mathbb{R}^n \). Then \( \partial f_H(x) = H^{-1} \partial f(x) \) for all \( x \in \mathbb{R}^n \).

**Proof.** From the definition of a subgradient, the following holds for all \( u \in \partial f_H(y) \) and any \( x, y \in \mathbb{R}^n \)

\[ f(x) = f_H(x) \geq f_H(y) + \langle u, x - y \rangle H \geq f(y) + \langle Hu, x - y \rangle_2. \]

This concludes the proof.

**Remark 10.** The inner product for \( E_H \) is independent of \( H \). This implies that for two functions \( f \in \Gamma_0(\mathbb{E}_H) \) and \( f \in \Gamma_0(\mathbb{E}_L) \) with \( f(x) = f(x) \) for all \( x \in \mathbb{R}^n \), we have that \( \partial f(x) = \partial f(x) \).

**Definition 11.** Given a function \( f : \mathcal{X} \rightarrow \mathbb{R} \), the conjugate function \( f^* : \mathcal{X}^* \rightarrow \mathbb{R} \) is defined as
\[ f^*(y) = \sup_x \{ \langle y, x \rangle - f(x) \}. \]

The conjugate function is different when \( f \) is defined on different spaces. Next, we state a result that quantifies this difference for spaces of interest to this paper.

**Proposition 12.** Suppose that \( f_H \in \Gamma_0(\mathbb{H}_H) \) and \( f \in \Gamma_0(\mathbb{E}_H) \) satisfies \( f_H(x) = f(x) \) for all \( x \in \mathbb{R}^n \). Then \( f_H^*(x) = f^*(Hx) \) for all \( x \in \mathbb{R}^n \).

**Proof.** From the definition of the conjugate function we have
\[ f_H^*(x) = \sup_y \{ \langle y, x \rangle_H - f_H(y) \} = \sup_y \{ \langle Hx, y \rangle_2 - f(y) \} = f^*(Hx). \]

This concludes the proof.

**Remark 13.** The inner product for \( E_H \) is independent of \( H \). This implies that for two functions \( f \in \Gamma_0(\mathbb{E}_H) \) and \( f \in \Gamma_0(\mathbb{E}_L) \) with \( f(x) = f(x) \) for all \( x \in \mathbb{R}^n \), we have that \( f^* = f^* \).

Next, we introduce the concepts of strong convexity and strong smoothness and present a result on how these concepts relate to each other through the conjugate function.

**Definition 14.** A function \( f : \mathcal{X} \rightarrow \mathbb{R} \) is \beta-strongly convex (w.r.t. \( \mathcal{X} \)) if
\[ f(x) \geq f(y) + \langle u, x - y \rangle + \frac{\beta}{2} \|x - y\|^2 \]
holds for all \( x, y \in \mathcal{X} \) and all \( u \in \partial f(y) \).

**Definition 15.** An equivalent characterization of \( \beta \)-strongly convex w.r.t. \( \mathcal{X} \) is that
\[ f(x) \geq f(y) + \langle u, x - y \rangle + \frac{\beta}{2} \|x - y\|^2 \]
holds for all \( x, y \in \mathcal{X} \).

Next, we state a result that links the notions of strong convexity and strong smoothness though the conjugate function. A proof to this can be found in Kakade et al. (2009) and the direction (i) to (ii) can be deduced from (Nesterov, 2005, Theorem 1).

**Proposition 18.** Suppose that \( f \in \Gamma_0(\mathcal{X}) \). Then the following are equivalent:

(i) \( f \) is \( \beta \)-strongly convex (w.r.t. \( \mathcal{X} \))

(ii) \( f^* \) is \( \frac{1}{\beta} \)-smooth (w.r.t. \( \mathcal{X}^* \))

**Remark 19.** The converse statement also holds since \( f = (f^*)^* \) for \( f \in \Gamma_0(\mathcal{X}) \).
Finally, before we proceed, we introduce the prox operation Moreau (1965) and show how it can be evaluated when applied to the conjugate function $f_H^*$.

**Definition 20.** For $f \in \Gamma_0(\mathcal{X})$, the prox operation is defined as

$$\text{prox}_f(y) := \arg\min_x \left\{ f(x) + \frac{1}{2\gamma} \|x - y\|^2 \right\}.$$  

**Remark 21.** For $f \in \Gamma_0(\mathcal{H}_H)$, the prox operation on the conjugate function $f_H^*$ can be computed as:

$$\text{prox}_{f_H^*}(y) = \arg\min_x \left\{ f_H^*(x) + \frac{1}{2\gamma} \|x - y\|^2 \right\} = \arg\min_v \left\{ f^*(v) + \frac{1}{2\gamma} \|v - H y\|^2 \right\} = H^{-1} \text{prox}_{f^*}(H y).$$

where $f^* \in \Gamma_0(\mathcal{E}_H^{-1})$.

### 3. PROBLEM FORMULATION

We consider optimization problems of the form

$$\begin{align*}
\text{minimize} & \quad f(x) + g(y) \\
\text{subject to} & \quad Ax = y
\end{align*} \tag{4}$$

We assume that the following assumption holds throughout the paper:

**Assumption 22.**

(a) The extended valued function $f \in \Gamma_0(\mathcal{E}_H)$ is 1-strongly convex (w.r.t. $\mathcal{E}_H$) with $H \in \mathcal{S}_++^{m \times n}$.

(b) The extended valued function $g \in \Gamma_0(\mathcal{E}_K)$ with $K \in \mathcal{S}_+^{n \times n}$.

(c) $A : \mathcal{E}_H \rightarrow \mathcal{E}_K^{-1}$ is a bounded linear operator.

**Remark 23.** Examples of functions that satisfy Assumption 22(b) are: $I_X$ and $I_{\chi_X}$, where $X$ is a nonempty, closed, and convex set, the 1-norm $\| \cdot \|_1$, and linear functions. Due to Example 4, we can treat $A$ as a matrix in $\mathbb{R}^{m \times n}$ since $A^* = A^T$.

We will also consider an equivalent optimization problem stated on $\mathcal{H}_H$. To this end, we define $f_H \in \Gamma_0(\mathcal{H}_H)$ to satisfy $f_H(x) = f(x)$ for all $x \in \mathbb{R}^m$. Define $g_K(y) = g(y)$ for all $y \in \mathbb{R}^n$, and the linear operator $A : \mathcal{H}_H \rightarrow \mathcal{H}_K$ to satisfy $Ax = Ay$ for all $x \in \mathbb{R}^m$. These definitions give rise to the following equivalent formulation of (4):

$$\begin{align*}
\text{minimize} & \quad f_H(x) + g_K(y) \\
\text{subject to} & \quad Ax = y
\end{align*} \tag{5}$$

The definitions of $f_H, g_H$ and $A$ implies, together with the definition of strong convexity and Proposition 9, that the following assumptions for (5) are equivalent to Assumption 22 for (4).

**Assumption 24.**

(a) The extended valued function $f_H \in \Gamma_0(\mathcal{H}_H)$ is 1-strongly convex (w.r.t. $\mathcal{H}_H$).

(b) The extended valued function $g_K \in \Gamma_0(\mathcal{H}_K)$

(c) $A : \mathcal{H}_H \rightarrow \mathcal{H}_K$ is a bounded linear operator.

The (negative) dual problem to (4) is

$$\begin{align*}
\text{minimize} & \quad d(\mu) + g^*(\mu) \\
\text{subject to} & \quad x^*(\mu) = \arg\min_x \left\{ \langle A^T \mu, x \rangle + f(x) \right\}
\end{align*} \tag{6}$$

where

$$d(\mu) := f^*(-A^T \mu). \tag{7}$$

Note that $d, g^* \in \Gamma_0(\mathcal{E}_K^{-1})$ for some $K \in \mathcal{S}_+^{n \times n}$. The (negative) dual to (5) is

$$\begin{align*}
\text{minimize} & \quad d_K(\mu) + g_K^*(\mu) \\
\text{subject to} & \quad x^*(\mu) = \arg\min_x \left\{ \langle A^* \mu, x \rangle + f_H(x) \right\}
\end{align*} \tag{8}$$

where

$$g_K^*(\mu) = \sup_y \left\{ \langle y, K - g_K(y) \rangle \right\}$$

$$f_H^*(\lambda) = \sup_x \left\{ \langle \lambda, x \rangle - f_H(x) \right\}$$

$$d_K(\mu) = f_H^*(-A^* \mu) \tag{9}$$

and $d_K, g_K^* \in \Gamma_0(\mathcal{H}_K)$ and $f_H^* \in \Gamma_0(\mathcal{H}_H)$. Note that, due to Example 4 and Definition 11, the shape of $d$ does not depend on which spaces $f, g$, and $A$ are defined, while the shape of $d_K$ does depend on which space $f_H, g_K$ and $A$ are defined. Thus, also the optimal dual solutions differ. Specifically, we have the following relation between optimal solutions of (6) and (8).

**Proposition 25.** Suppose that the set of optimal dual points to (6) defined on $\mathcal{E}_K^{-1}$ is given by $\Lambda^*$ and that the set of optimal dual points to (8) defined on $\mathcal{H}_K$ is given by $\Lambda^*_K$. Then $\Lambda^* = K \Lambda^*_K$.

**Proof.** We have

$$\begin{align*}
d(\mu) &= f^*(-A^T \mu) = f_H^*(H^{-1}A^T \mu) = f_H^*(-H^{-1}A^T K v) = f_H^*(A^*v) = d_K(v)
\end{align*}$$

where the first equality is by definition of $d$ in (7), the second is from Proposition 12, the third by setting $\mu = K v$.

The performance of fast dual forward backward splitting is highly sensitive to on which space the algorithm is applied. This paper is about computing $H$ and $K$, based on problem data, that define spaces on which fast dual forward backward splitting performs well. We also show that applying fast dual forward backward splitting to solve (6) on $\mathcal{E}_K^{-1}$ is equivalent to applying fast dual forward backward splitting to solve (8) on $\mathcal{H}_K$.

### 4. DUAL PROBLEM PROPERTIES

From, e.g., (Nesterov, 2005, Theorem 1), we know that $d$ and $d_K$ are differentiable with

$$\begin{align*}
\nabla d(\mu) &= -A^T x^*(\mu) \tag{10} \\
\nabla d_K(\mu) &= -A^* x_H^*(\mu) \tag{11}
\end{align*}$$

where

$$x^*(\mu) = \arg\min_x \left\{ \langle A^T \mu, x \rangle + f(x) \right\} \tag{12}$$

$$x_H^*(\mu) = \arg\min_x \left\{ \langle A^* \mu, x \rangle + f_H(x) \right\}. \tag{13}$$

It is also well known that $\nabla d$ and $\nabla d_K$ are Lipschitz continuous. This is shown in the following proposition, which is an application of (Nesterov, 2005, Theorem 1).

**Proposition 26.** Suppose that Assumptions 22 and 24 hold and that $d$ and $d_K$ are defined as in (7) and (9) respectively. Then $d$ is $\|K^{1/2}A H^{-1/2}\|_2$-smooth w.r.t. $\mathcal{E}_K^{-1}$ and $d_K$ is $\|K^{1/2}A H^{-1/2}\|_2$-smooth w.r.t. $\mathcal{H}_K$. 

Proof. From (Nesterov, 2005, Theorem 1), we have that $d$ is $\|A^*\|_{K^{-1},E_H}$-smooth (w.r.t. $\mathbb{E}_{K^{-1}}$). Example 6 implies that

$$
\|A^*\|_{K^{-1},E_H} = \max_x \left\{ \|Ax\|_{K^{-1}} : \|x\|_{E_H} = 1 \right\} = \max_x \left\{ \|Ax\|_K : \|x\|_H = 1 \right\} = \max_v \left\{ \|K^{1/2}AH^{-1/2}v\|_2 : \|v\|_2 = 1 \right\} = \|K^{1/2}AH^{-1/2}\|_2.
$$

From (Nesterov, 2005, Theorem 1) we also conclude that $d_K$ is $\|A^*\|_{K^{-1},E_H}$-smooth (w.r.t. $\mathbb{H}_K$). Example 6 implies that

$$
\|A^*\|_{K^{-1},E_H} = \max_x \left\{ \|Ax\|_{K^{-1}} : \|x\|_{E_H} = 1 \right\} = \max_x \left\{ \|Ax\|_K : \|x\|_H = 1 \right\} = \max_v \left\{ \|K^{1/2}AH^{-1/2}v\|_2 : \|v\|_2 = 1 \right\} = \|K^{1/2}AH^{-1/2}\|_2.
$$

This concludes the proof.

By letting $E_H = E_L = \mathbb{R}^m$, we get that the Lipschitz constant to $\nabla d$ is $\|AH^{-1}A^T\|_2$. This is exactly the result provided in Richter et al. (2013) which implies that

$$
d(\mu) \geq d(\nu) + \langle \nabla d(\nu) - \nabla d(\mu), \mu - \nu \rangle + \frac{\|AH^{-1}A^T\|_2}{2} \|\mu - \nu\|_2^2
$$

holds for all $\mu, \nu \in \mathbb{R}^m$. By assuming that $A$ has full row rank and letting $K = (AH^{-1}A^T)^{-1}$, Proposition 26 implies that $d$ is 1-smooth w.r.t. $\mathbb{E}_{AH^{-1}A^T}$, i.e. that

$$
d(\mu) \geq d(\nu) + \langle \nabla d(\nu) - \nabla d(\mu), \mu - \nu \rangle + \frac{\|AH^{-1}A^T\|_2}{2} \|\mu - \nu\|_2^2
$$

holds for all $\mu, \nu \in \mathbb{R}^{AH^{-1}A^T}$. This is obviously a tighter characterization of $d$ than (14). In this section, we will extend this result to when $A$ does not have full row rank. We will also show that for many functions $f$, no better quadratic upper bound than (15) can be found. We also show that for $f$ being a quadratic function plus the indicator function of an affine subspace, the bound can improved.

Similarly to the above, we have $d_K = f_{p_k} \circ -A^* = (-A^T f_{H^*})$. Denote by $p_K = -A^T f_H$ (which implies $d_K = p_K$) which is defined by

$$
p_K(y) := \inf_{x \in A^{-1}y} f_H(x).
$$

Again, Proposition 18 gives that $\beta$-smoothness (w.r.t. $\mathbb{H}_{K}$) of $p_K$ is equivalent to $\frac{1}{\beta}$-strong convexity of $p$ (w.r.t. $\mathbb{H}_{K}$).

This proves (i). Point (ii) follows from (i) by applying Proposition 27.

These results allow us to focus on smoothness properties of either $d$ or $d_K$ and translate these properties to the other function when needed. In the remainder of this section, we will focus on smoothness properties of $d$ and translate these to $d_K$ when needed. Next, we show that (15) holds also for the case of $A$ not having full row rank.

Proposition 29. Suppose that $f : \mathbb{E}_H \rightarrow \mathbb{R}$ is 1-strongly convex. Then $d : \mathbb{E}_{K^{-1}} \rightarrow \mathbb{R}$ as defined in (7) satisfies

$$
d(\mu) \leq d(\nu) + \langle \nabla d(\nu) - \nabla d(\mu), \mu - \nu \rangle + \frac{1}{2} \|\mu - \nu\|_L^2
$$

with $L = AH^{-1}A^T$ for all $\mu, \nu \in \mathbb{E}_{K^{-1}}$.

Proof. From Proposition 18, we know that $f^* = 1$-smooth w.r.t. $\mathbb{E}_{\nu^{-1}}$. Thus, (3) holds for $f^*$ for any $x, y \in \mathbb{E}_{H^{-1}}$. Especially, let $x = -A^T \mu$ and $y = -A^T \nu$ to get

$$
d(\mu) \leq f^*(-A^T \mu) \leq d(\nu) + \langle \nabla f^*(-A^T \nu), -A^T(\mu - \nu) \rangle + \frac{1}{2} \|\mu - \nu\|_{AH^{-1}A^T}^2
$$

This concludes the proof.

Remar 30. Proposition 29 implies through Proposition 27 and Corollary 28 that for any $K > 0$ such that $K^{-1} \succeq AH^{-1}A^T$, we have that $d : \mathbb{E}_{K^{-1}} \rightarrow \mathbb{R}$ is 1-smooth (w.r.t. $\mathbb{E}_{K}$) and $d_K$ is 1-smooth (w.r.t. $\mathbb{H}_{K}$).

Next, we show that for many interesting functions $f$, the bound provided in Proposition 29 on $d$ is indeed tight. Essentially, we show that if the strong convexity bound on the primal (i.e. the quadratic lower bound to the primal) is tight, so is the smoothness bound on the dual (i.e. the quadratic upper bound to the dual).
Proposition 33. Suppose that \( f : \mathbb{E}_H \to \mathbb{R} \) is 1-strongly convex (w.r.t. \( \mathbb{E}_H \)) and that there exists a full-dimensional ball \( B^n_\infty(x^*(\bar{\mu})) \) centered around \( x^*(\bar{\mu}) \) for some \( \bar{\mu} \in \mathbb{E}_{K^-1} \) on which (2) is tight. Then no matrix \( L \not\preceq AH^{-1}A^T \) exists such that \( d : \mathbb{E}_{K^-1} \to \mathbb{R} \) as defined in (7) satisfies (16) for all \( \mu, \nu \in \mathbb{E}_{K^-1} \).

Proof. The strong convexity definition (2) for \( \mathbb{E}_H \) is tight on \( B^n_\infty(x^*(\bar{\mu})) \) if and only if \( f(x) = \frac{1}{2}\|x\|^2_H + \langle \xi, x \rangle + \theta \) on \( B^n_\infty(x^*(\bar{\mu})) \). Since \( x^*(\bar{\mu}) \in B^n_\infty(x^*(\bar{\mu})) \), we have

\[
x^*(\bar{\mu}) = \text{argmin}_x \left\{ \frac{1}{2}\|x\|^2_H + \langle \xi, x \rangle + \theta + \langle A^T \bar{\mu}, x \rangle \right\}
\]

\[
= -H^{-1}(\xi + A^T \bar{\mu}).
\]

Due to continuity of \( H^{-1/2}A^T \), we also have \( x^*(\bar{\mu}) + \mu = -H^{-1}(\xi + A^T(\mu + \bar{\mu})) \in B^n_\infty(x^*(\bar{\mu})) \) for sufficiently small \( \mu \in \mathbb{R}^m \). Thus, there exists a ball \( B^n_\infty(\mu) \) such that for each \( \mu \in B^n_\infty(\mu) \),

\[
d(\mu) = -\frac{1}{2}\|A^T \mu + \xi\|^2_H - \theta.
\]

That is, \( d \) is a quadratic function with Hessians \( AH^{-1}A^T \) on \( B^n_\infty(\mu) \). This implies that for any \( L \not\preceq AH^{-1}A^T \), where exist \( \nu, \mu \in B^n_\infty(\mu) \) (since \( B^n_\infty(\mu) \) has non-empty interior) such that (16) does not hold. This concludes the proof.

Remark 34. Under the same assumptions, we get through Proposition 27 and Corollary 28 that for any \( K > 0 \) such that \( K^{-1} \not\preceq AH^{-1}A^T \), we have that \( d : \mathbb{E}_{K^-1} \to \mathbb{R} \) is 1-smooth (w.r.t. \( \mathbb{E}_{K^-1} \)) and \( d_K \) is not 1-smooth (w.r.t. \( \mathbb{E}_{K^-1} \)).

The assumptions in Proposition 33 are met, for instance, if \( A \) has full column rank and \( f(x) = \frac{1}{2}\|x\|^2_H + h(x) \) where \( h \) is the indicator function for a closed and convex constraint set with nonempty interior, the 1-norm, a linear function, or any other function that is linear on a convex subset with nonempty interior. For these examples, Proposition 31 indeed gives the best obtainable quadratic majorizer of \( d \). However, for \( f \) being a quadratic plus the indicator functions for an affine subspace \( I_{R_\beta=0} \), the assumptions in Proposition 33 are not met. The reason is that the interior of \( I_{R_\beta=0} \) is empty (except in the trivial case where \( B = 0 \) and \( \beta = 0 \)). In the following proposition we present a result that shows how the quadratic bound in Proposition 33 can be improved in this case.

Proposition 33. Assume that \( f(x) = \frac{1}{2}\|x\|^2_H + \langle \xi, x \rangle + I_{R_\beta=0} \) with \( H \in \mathbb{S}^n_+ \), \( \xi \in \mathbb{R}^n \), \( B \in \mathbb{R}^{n \times n} \), and \( \beta \in \mathbb{R}_+ \). Further assume that \( x^T H x > 0 \) whenever \( x \neq 0 \) and \( B x = 0 \), i.e., that \( H \) is positive definite on the null-space of \( B \). Then \( d : \mathbb{E}_{K^-1} \to \mathbb{R} \) satisfies

\[
d(\mu) \leq d(\nu) + \langle \nabla d(\nu), \mu - \nu \rangle + \frac{1}{2}\|\mu - \nu\|^2_L \leq L d(\mu) \tag{17}
\]

for all \( \mu, \nu \in \mathbb{E}_{K^-1} \), where

\[
\begin{bmatrix}
M_{11} & M_{12} \\
M_{21} & M_{22}
\end{bmatrix} = \begin{bmatrix}
H & B^T \\
B & 0
\end{bmatrix}^{-1}.
\]

Further, no matrix \( L \not\preceq AH^{-1}A^T \) exists such that (17) holds for all \( \mu, \nu \in \mathbb{E}_{K^-1} \).

Proof. Since \( H \) is positive definite on the null-space of \( A \), the inverse in (18) exists, see (Boyd and Vandenberghe, 2004, p. 523). Thus \( x^*(\mu) = -M_{11}(A^T \mu + \xi) + M_{12}b \), where \( x^* \) is defined in (12). Further

\[
d(\mu) = \frac{1}{2} \mu^T A(2M_{11} - M_{11}HM_{11})A^T \mu + \zeta^T \mu + \theta
\]

where \( \zeta \in \mathbb{R}^m \) and \( \theta \in \mathbb{R} \) collect the linear and constant terms, and where \( M_{11}HM_{11} = M_{11} \) is used in the second equality. This identity follows from the upper left block of \( \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \), \( B = 0 \) and \( \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \) and using \( M_{11}B = M_{11}B = BM_{11} = 0 \), where \( BM_{11} = 0 \) follows from the lower left block of \( \begin{bmatrix} H & 0 \\ B & 0 \end{bmatrix} \). This implies that (17) holds with \( L = AM_{11}A^T \). Further, since \( d \) is a quadratic function with Hessian \( AM_{11}A^T \), (17) is tight for \( L = AM_{11}A^T \) for all \( \mu, \nu \in \mathbb{E}_{K^-1} \). Thus,

\[
L \not\preceq AM_{11}A^T \text{ exists such that (17) holds for all } \mu, \nu \in \mathbb{E}_{K^-1} \text{. This concludes the proof.}
\]

5. FAST DUAL FORWARD-BACKWARD SPLITTING

Fast forward-backward splitting can be applied to solve problems of the form

\[
\text{minimize } \ell(x) + \psi(x) \tag{19}
\]

where \( \ell \in \Gamma_0(\mathbb{X}) \) is \( \beta \)-smooth and \( \psi \in \Gamma_0(\mathbb{X}) \). Fast forward-backward splitting is given by the iterations

\[
y^{k+1} = x^k + \alpha_k (x^k - x^{k-1}) \tag{20}
\]

\[
x^{k+1} = \text{argmin}_x \left\{ \ell(y^k) + \langle \nabla \ell(y^k), x - y^k \rangle + \frac{\beta}{2}\|x - y^k\|^2_H + \psi(x) \right\} \tag{21}
\]

where \( \alpha_k = \theta_k \beta^{-1}(k-1) - 1 \) and the \( \theta_k \)-sequence satisfies (see Tseng (2008))

\[
\frac{1}{\theta_k^2} = 1 - \frac{\theta_k}{\theta_{k+1}}. \tag{22}
\]

When the functions in (19) are defined on \( \psi \in \Gamma_0(\mathbb{E}_H) \) and \( \ell \in \Gamma_0(\mathbb{E}_H) \), the forward backward step in (21) becomes

\[
\text{argmin}_x \left\{ \ell(y^k) + \langle \nabla \ell(y^k), x - y^k \rangle + \frac{\beta}{2}\|x - y^k\|^2_H + \psi(x) \right\}
\]

\[
\approx \text{prox}_{\frac{\beta}{2}\psi}(y^k - \frac{\beta}{2}H^{-1}\nabla \ell(y^k)). \tag{23}
\]

In that case, (20)-(21) is the generalized fast proximal gradient method used in Zuo and Lin (2011). This algorithm is in Zuo and Lin (2011) shown to converge as

\[
\ell(x^k) - \ell(x^*) \leq \frac{2\beta\|x^* - x^0\|^2_H}{(k+1)^2} \tag{24}
\]

where \( \ell : \mathbb{E}_H \to \mathbb{R} \) and \( x^* \) is an optimal solution to (19). When instead applying fast forward-backward splitting on (19) with \( \psi \in \Gamma_0(\mathbb{E}_H) \) and \( \ell \in \Gamma_0(\mathbb{E}_H) \) with \( \ell \) being \( \beta \)-smooth (w.r.t. \( \mathbb{E}_H \)), the forward backward step in (21) becomes
Further, $\nu$ the chain-rule in the third, Remark 21 in the fourth, that $\ell$ of the preconditioned problem

$$Dx = Dy$$

scheme is applied, the primal iterates coincide. The primal iterates are obtained when computing the gradient of $\nabla d$ and $\nabla d_K$ in (10) and (11) respectively, i.e., when computing $x'(u^k)$ and $x_H^K(v^k)$ defined in (12) and (13) respectively. That they coincide can be shown as follows:

$$x^*(u^k) = \arg\min_x \{ f(x) + \langle A^T u^k, x \rangle \}$$

$$= \arg\min_x \{ f(x) + \langle H^{-1} A^T u^k, x \rangle \}$$

$$= \arg\min_x \{ f(x) + \langle H^{-1} A^T K u^k, x \rangle \}$$

$$= \arg\min_x \{ f(x) + \langle A^* v^k, x \rangle \}$$

where Example 4 is used in the fourth equality.

Next, we state a proposition that shows that the convergence rates coincide for fast dual forward backward splitting on $E_{K^{-1}}$ and $H_K$, when applied to solve (6) and (8) respectively.

**Proposition 38.** Suppose that Assumptions 22 and 24 hold and that the dual problems (6) and (8) are defined on $E_{K^{-1}}$ and $H_K$ respectively, where $K > 0$ satisfies $K^{-1} \succeq AH^{-1}A^T$. Further suppose that $\{u^k\}$ and $\{v^k\}$ are generated by (20)-(21) with $\beta = 1$ applied to solving (6). Then

$$D(v^k) - D(v^*) \leq \frac{2\|v^k - v^0\|_{K^{-1}}}{(k+1)^2}$$

where $D = d + g^*$ and $v^*$ is an optimal solution to (6). Further suppose that $\{v^k\}$ and $\{u^k\}$ are generated by (20)-(21) with $\beta = 1$ applied to solving (8). Then

$$D_K(K^{-1}u^k) - D_K(K^{-1}v^*) \leq \frac{2\|\mu^k - \mu^0\|_{K^{-1}}}{(k+1)^2}$$

where $D_K = d_K + g_K^*$ and $\mu^*$ is an optimal solution to (8). In addition $K\mu^0 = v^0$ and $K\mu^{-1} = v^{-1}$, these convergence estimates are equivalent.

**Proof.** From Remark 30, we have that $d : E_{K^{-1}} \to \mathbb{R}$ is 1-smooth (w.r.t. $E_{K^{-1}}$) for any $K^{-1} \succeq 0$ such that $K^{-1} \succeq AH^{-1}A^T$ and that $d_K : H_K \to \mathbb{R}$ is 1-smooth (w.r.t. $H_K$). This implies that the convergence estimates (25) and (26) follow from Zuo and Lin (2011) and Tseng (2008) respectively.

Next, we show equivalence between the convergence rates. Proposition 25 and Proposition 35 imply that $\mu^k = K^{-1}u^k$ for all $k$ and $\mu^* = K^{-1}v^*$. Thus, (26) is equivalent to

$$D_K(K^{-1}u^k) - D_K(K^{-1}v^*) \leq \frac{2\|K^{-1}(v^* - v^0)\|_{K^{-1}}^2}{(k+1)^2}$$

where $D_K = d_K + g_K^*$ and $\mu^*$ is an optimal solution to (8). In addition $K\mu^0 = v^0$ and $K\mu^{-1} = v^{-1}$, these convergence estimates are equivalent.

**Remark 36.** From Giselsson and Boyd (2014) it follows that these algorithms are equivalent to applying forward-backward splitting on the Euclidean space $\mathbb{R}^m$ to the dual of the preconditioned problem

$$\min_{x \in \mathbb{R}^m} f(x) + g(y) \text{ subject to } Dx = Dy$$

where $K = D^TD$.

**Remark 37.** Although the dual iterates differ depending on which space the fast dual forward-backward splitting
From (21), we see that the smooth function in fast forward-backward splitting when applied on $E_H$ or $\|E_H$ is approximated by a quadratic function that majorizes it. More specifically, it is approximated by the r.h.s. of (3) that describes the smoothness properties of the function. Obviously, the tighter this quadratic majorizer is, the more accurate the function approximation used in the algorithm is, and consequently the faster the convergence of the algorithm is likely to be. In the case of fast dual forward-backward splitting, Proposition 29 suggests that $E_L$ with $L = AH^{-1}A^T$ would be a good space on which to apply the algorithm, since the r.h.s. of (3) would (in many cases) be the best quadratic majorizer of $d$. In most cases, however, it is not advisable to let $L = AH^{-1}A^T$ since the prox operation of $g^*$ could become computationally too expensive. For instance, if $g^*$ is separable, using a non-diagonal $L$ would typically increase the computational burden in each iteration more than what is gained by the reduced number of iterations. This discussion suggests that to get good performance, we should choose $L \approx AH^{-1}A^T$ where the structure of $L$ should be such that the computational complexity of each iteration is kept low. We also need $L \succeq AH^{-1}A^T$ to guarantee convergence of the algorithm.

By letting an invertible matrix $E$ satisfy $L = (E^T E)^{-1}$, these objectives can be formulated as choosing $E$ such that $I \approx EQE^T$ and $I \succeq EQE^T$. A natural choice is then to minimize $\kappa(Q)$ subject to $I \succeq EQE^T$, where $\kappa$ denotes the condition number. However, if $A$ does not have full row rank, or if the objective is to satisfy $L \approx AM_{11}A^T$ (when the assumptions in Proposition 33 hold), the matrix $APAT$ (with $P = H^{-1}$ or $P = M_{11}$) does not have a well defined condition number. In such cases, the ratio between the largest and smallest non-zero eigenvalues of $EAPAT^TE^T$ could be minimized instead. This is reasonable, since the zero eigenvalues of $APAT$ cannot be affected or made positive by pre- and post-multiplying $APAT$ with $E$ and $E^T$ respectively.

In the following section, we show how to find an $E$ that minimizes the ratio between the largest and smallest non-zero eigenvalues of $EAPAT^TE^T$ (with $P = H^{-1}$ or $P = M_{11}$). Finding such $E$ is often computationally expensive and only possible for small-scale problems. We also present computationally less expensive methods that are heuristics for reducing the (pseudo) condition number of $EAPAT^TE^T$.

### 6. COMPUTING THE METRIC MATRIX

In this section, we show how to solve

$$\min \frac{\lambda_1(EQE^T)}{\lambda_r(EQE^T)} \quad (28)$$

where $\lambda_1$ denotes the largest (non-zero) eigenvalue, $\lambda_r$ denotes the smallest non-zero eigenvalue, and $Q \in S_+^n$ is $Q = AH^{-1}A^T$ or $AM_{11}A^T$. We restrict the matrix $E$ to be full, block-diagonal, or diagonal, since this implies that also $L = (E^T E)^{-1}$ is full, block-diagonal, or diagonal respectively. We denote by $E$ any of these structural constraints. Besides showing how to solve (28) exactly, we also present a heuristic optimization formulation that gives $E QE^T \approx I$ and is often less computationally demanding than solving (28). We also present equilibration methods that are computationally very cheap methods that often improve the conditioning.

#### 6.1 Exact condition number minimization

In this section, we show how to solve (28) exactly. We consider two different cases; $Q$ positive definite, and $Q$ positive semi-definite. Before we show how to solve (28), we state the following well known lemma that follows directly from the definition of singular values.

**Lemma 39.** For any matrix $\Phi \in \mathbb{R}^{m \times n}$, the non-zero eigenvalues of $\Phi^T \Phi$ equals the non-zero eigenvalues of $\Phi \Phi^T$.

*The positive definite case.* Here, we assume that $Q$ is positive definite, which occurs, for instance, if $Q = AH^{-1}A^T$, where $H \in S_+^n$ and $A \in \mathbb{R}^{m \times n}$ has full row rank.

**Proposition 40.** Assume that $Q \in S_+^m$. Then a matrix $E \in E$ that minimizes the ratio (28) can be computed by solving the semi-definite program

$$\min t \quad \text{subject to} \quad tQ \succeq E = E \quad (29)$$

where $L = (E^T E)^{-1}$. Further, $L \succeq Q$.

**Proof.** Since $Q$ has full rank, (28) is the condition number. Thus, according to (Boyd et al., 1994, Section 3.1), (29) can be solved in order to minimize (28). Further, the second constraint implies that $L \succeq Q$.

*The positive semi-definite case.* Here, we assume that $Q$ is positive semi-definite. This situation occurs, e.g., if $Q = AH^{-1}A^T$ and $A \in \mathbb{R}^{m \times n}$ with $m > n$, or if $Q = M_{11}A^T$.

**Proposition 41.** Assume that $Q \in S_+^n$ is factorized as $Q = R^T R$, where $R \in \mathbb{R}^{n \times k}$ has rank $k$. Then a matrix $E \in E$ that minimizes the ratio (28) can be computed by solving the semi-definite program

$$\min -t \quad \text{subject to} \quad RMR^T \preceq I \quad RMR^T \succeq tI$$

$$M \in E \quad (30)$$

where $M = (E^T E)$. Further $L = M^{-1} = (E^T E)^{-1} \succeq Q$.

**Proof.** Since $RMR^T$ has full rank, we get from Lemma 39 and equalities $M = E^T E$ and $Q = R^T R$ that minimizing the condition number of $RMR^T$ is equivalent to minimizing the ratio between the largest and smallest non-zero eigenvalues of $EQE^T$, i.e. equivalent to solving (28). From (Boyd et al., 1994, Section 3.1), we get that (30) minimizes the condition number of $RMR^T$, i.e., it minimizes (28). Further, the first inequality in (30) implies through Lemma 39 that $EQE^T \preceq I$, which is equivalent to that $L = (E^T E)^{-1} \succeq Q$. This concludes the proof.

#### 6.2 Heuristic 1 - Trace minimization

To solve (28) using (29), and especially (30), is computationally expensive. Here, we propose the following alternative optimization problem that gives an $L$ that well
approximates $Q$, where $Q$ is positive semi-definite (e.g. $Q = AH^{-1}AT$ or $Q = AM_{11}A^T$):

\[
\begin{align*}
\text{minimize} & \quad \text{trace } L \\
\text{subject to} & \quad Q \preceq L \\
& \quad L \in \mathcal{E}
\end{align*}
\]

This is a semi-definite program that is often less computationally expensive than (29) and far less computationally expensive than (30). However, being a semi-definite program, it is still restricted to fairly small-scale problems.

6.3 Heuristic 2 – Equilibration

In equilibration, given a matrix $T \in \mathbb{R}^{m \times n}$, the objective is to find positive and diagonal matrices $E \in \mathbb{R}^{m \times m}$ and $F \in \mathbb{R}^{n \times n}$ such that all rows and columns of the scaled matrix $ETF$ have the same length in some norm. This is a heuristic to reduce the condition number of the scaled matrix $ETF$ compared to the original matrix $T$, see Bradley (2010) for an overview of equilibration and further references. There are no guarantees that the condition number is reduced, but in practice, this is most often the case. In our setting, we will use this heuristic to improve the conditioning of $EQET$, which is the objective stated in (28). To this end, we consider symmetric equilibration where the pre- and post-multiplied matrices are the same, i.e., where $E = F$. Below, we present different methods to achieve symmetric equilibration of symmetric matrices in the 1-norm, 2-norm and $\infty$-norm. None of these methods guarantee that $I \succeq EQET$, which is required to get convergence of the fast dual forward-backward splitting method. However, this is achieved by appropriately scaling $E$.

**Equilibration in 1-norm and 2-norm.** For a symmetric matrix, the $i$th row and column are the same, hence also their norms. Thus, in symmetric equilibration of symmetric matrices, we need only equilibrate either the rows or the columns. The 1-norm or row-wise $\|EQET\|_1$ is given by

\[
\|EQET\|_1 = \sum_{i=1}^{m} |E_{ii}| \sum_{j=1}^{m} |Q_{ij}| E_{jj}
\]

since $E$ is diagonal with $E_{ii} > 0$. Similarly, the squared 2-norm is given by

\[
\|EQET\|_2^2 = \sum_{i=1}^{m} (E_{ii} |Q_{ij}| E_{jj})^2 = E_{ii}^2 \sum_{j=1}^{m} |Q_{ij}|^2 E_{jj}
\]

Thus, by introducing the matrices $T_1 = |Q|$ (where $\cdot$ denotes element-wise absolute value) and $T_2 = (Q)^{\frac{1}{2}}$ (where $\cdot^{\frac{1}{2}}$ denotes element-wise square), and by letting $E = \text{diag}(\tilde{e})$, symmetric equilibration can be stated as finding $E$ (and $\tilde{e}$) such that

\[
ET_1 e = 1
\]

in the 1-norm case and

\[
E^{\frac{1}{2}} T_2 e^{\frac{1}{2}} = 1
\]

in the 2-norm case. We treat these cases simultaneously by introducing $\tilde{E} = \text{diag}(\tilde{e})$ and $T$ that satisfies $\tilde{E} = E$ and $T = T_1$ in the 1-norm case, and $\tilde{E} = E^{\frac{1}{2}}$ and $T = T_2$ in the 2-norm case. The conditions (31) and (32) can then be written as

\[
0 = TE - \tilde{E}^{-1}1.
\]

This is indeed the gradient of the function

\[
\phi(\tilde{e}) = \frac{1}{2} \tilde{e}^T T \tilde{e} - \sum_{i=1}^{n} \ln(\tilde{e}_i).
\]

Since $\tilde{e}^T T \tilde{e} \geq \sum_{i=1}^{n} (T_{ii} \tilde{e}_i^2) \geq (\min T_{ii}) |\tilde{e}|^2$ for all $\tilde{e} \in \text{dom} \phi$, i.e. for all $\tilde{e} > 0$, and since $-\ln$ is convex, $\phi$ is convex on its domain. If in addition $\min T_{ii} > 0$ (which is the case we are interested in), then $\phi$ is strongly convex. Since $\phi(\tilde{e}) < \infty$ for all $\tilde{e} \in \text{int} \text{dom}(\phi)$ and since $\phi(\tilde{e}) \to \infty$ as $\tilde{e}$ approaches the boundary of the domain, we conclude that $\phi$ has a unique minimizer $\tilde{e}^* \in \text{int}(\text{dom}(\phi))$. This unique minimizer can be found in various ways.

One approach is to perform element-wise optimization and cycle through the elements until convergence. Optimality conditions for optimizing (33) w.r.t. $\tilde{e}_i$, while the other $\tilde{e}_j$ are fixed, are given by

\[
0 = T_{ii} \tilde{e}_i + \sum_{j \neq i} T_{ij} \tilde{e}_j + 1/\tilde{e}_i
\]

Each such element-wise optimization is very cheap since it requires the solution of a second-order equation and a vector-vector multiplication of size $m - 1$.

A classic method to perform 1-norm and 2-norm equilibration is the (symmetric) Sinkhorn-Knopp algorithm, Sinkhorn and Knopp (1967), which was originally developed to generate doubly stochastic matrices from positive matrices. The symmetric Sinkhorn-Knopp algorithm is given by the iteration

\[
\tilde{e}^{k+1} = (T \tilde{e}^k)^{-1}
\]

where $(\cdot)^{-1}$ denotes element-wise reciprocal. This is known to converge, see Sinkhorn and Knopp (1967), to an alternating sequence that satisfies $\tilde{e}^k = \beta \tilde{e}^{k+1}$ for some $\beta \in \mathbb{R}_{++}$ under the conditions that $T$ is positive, symmetric, and fully indecomposable. Full indecomposability means that no sub-matrix with only zeros of size $l \times p$ exists where $l + p \geq m$, where $m$ is the dimension of $T$. This excludes, e.g., block-diagonal matrices where each block instead have to be equilibrated separately.

Also other equilibration methods exist, see Ruiz (2001); Bradley (2010); Knight et al. (2011). Common for all these methods is that usually two to five passes over the data are sufficient to obtain a close to equilibrated matrix. This implies that they are computationally very cheap, and can be used both for offline and online metric selection.

**Equilibration in $\infty$-norm.** In $\infty$-norm equilibration of general symmetric matrices, the magnitude of the largest element in each row (or column) is set to 1. For positive semi-definite matrices $S \in S^+_n$, we have that $S_{ii} \geq 0$ and max $S_{ii} \leq \max_{i \neq j} |S_{ij}|$, see (Horn and Johnson, 1990, p. 398). Thus, for positive semi-definite matrices with $S_{ii} > 0$ for all $i$, having $S_{ii} = 1$ for all $i$ gives equilibration in the $\infty$-norm. For $S = EQET$ with $Q$ positive semi-definite with positive diagonal, this scaling, which is also referred to as Jacobi scaling, is obtained by letting $E_{ii} = 1/\sqrt{Q_{ii}}$. This is less computationally expensive than equilibration in the 1-norm or 2-norm as discussed above.
7. APPLICATIONS

In this section, we show how to apply the results in this paper on two quadratic programming formulations. We consider one formulation with no specific structure that we solve in two different ways, and one that has a separable structure that allows for distributed implementation. We state the algorithms on $\mathbb{E}_K$ since they are equivalent to the algorithms on $\mathbb{H}_K$ and let $L = K^{-1}$ to simplify notation. The reason why $\mathbb{E}_K$ is chosen rather than $\mathbb{H}_K$ is that the gradient on $\mathbb{E}_K$ coincides with the gradient in the Euclidean space.

7.1 Two QP splittings

Here, we consider the following quadratic program

$$
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|x\|^2_H + \langle \xi, x \rangle_2 \\
\text{subject to} & \quad Bx = b \\
& \quad d \leq Cx \leq d
\end{align*}
$$

where $H \in \mathbb{S}^n_+$, $\xi \in \mathbb{R}^n$, $B \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $C \in \mathbb{R}^{p \times n}$, and $d, d \in \mathbb{R}^p$. We form two different dual problems to this problem by employing two different splittings that has previously been used in the literature. We present the resulting fast dual forward-backward splitting method and discuss how to choose the space on which to apply the algorithm.

**QP splitting 1.** In the first splitting, which has been used in Patrinos and Bemporad (2014) in the context of fast dual forward-backward splitting and in O’Donoghue et al. (2013) in the context of ADMM, we set $f$ and $g$ in (4) to

$$
\begin{align*}
f(x) &= \frac{1}{2} \|x\|^2_H + \langle \xi, x \rangle_2 + I_{Bx=b}(x) \\
g(y) &= I_{d \leq y \leq d}(y)
\end{align*}
$$

which gives equality constraint $Cx = y$. (Here, it is enough to assume that $H$ is positive definite on the null-space of $B$.) We form the dual as in (6) and restrict $L = K^{-1}$ in the algorithm to be diagonal. After some simplification, the resulting fast dual forward-backward splitting algorithm on $\mathbb{E}_L = \mathbb{E}_K$ becomes

$$
\begin{align*}
\nu^{k+1} &= \mu^k + \alpha_k(\mu^k - \mu^{k-1}) \\
x^k &= \text{argmin}_x \left\{ \frac{1}{2} \|x\|^2_H + \langle \xi, x \rangle_2 + I_{Bx=b}(x) + \langle \nu, Cx \rangle_2 \right\} \\
\mu^k &= \min \left( \nu^k + L^{-1}(Cx^k - d), \max \left( \nu^k + L^{-1}(Cx^k - d), 0 \right) \right)
\end{align*}
$$

where $\alpha_k = \theta_k(\theta_k - 1)$ and $\theta_k$ satisfies (22). The restriction that $L$ is diagonal implies that the prox-operation becomes a min-max operation only, hence very cheap. The matrix $L$ should be computed in accordance with the suggestions in Section 6. Specifically, according to Proposition 33, we need $L \geq CM_{11}CT$, where $M_{11}$ is defined in (18). Obviously also $L \geq CH^{-1}CT$ holds since $M_{11} \preceq H^{-1}$.

Equation (35) can be efficiently implemented since it is an equality constrained quadratic problem. It can be solved by forming and storing $M_{11}$ and $M_{12}$ in (18), and reuse these in all iterations. Another option, that might be beneficial if $\begin{bmatrix} H & B^T \\ B & 0 \end{bmatrix}$ (from (18)) is sparse, is to compute and store a sparse LDL factorization of $\begin{bmatrix} H & B^T \\ B & 0 \end{bmatrix}$ and perform forward and backward substitution on the factorization in each subsequent iteration.

**QP splitting 2.** The second splitting, which has been used in Richter et al. (2013), is obtained by letting $f$ and $g$ in (4) be

$$
\begin{align*}
f(x) &= \frac{1}{2} \|x\|^2_H + \langle \xi, x \rangle_2 + I_{\|x\| \leq d}(x) \\
g(y) &= I_{y = b}(y)
\end{align*}
$$

which gives equality constraint $Bx = y$. Without structural restrictions on $L = K^{-1}$, the resulting fast dual forward backward splitting method on $\mathbb{E}_L = \mathbb{E}_K$ becomes after some simplification:

$$
\begin{align*}
\nu^{k+1} &= \mu^k + \theta_k(\mu^k - \mu^{k-1}) \\
x^k &= \text{argmin}_x \left\{ \frac{1}{2} \|x\|^2_H + \langle \xi, x \rangle_2 + I_{\|x\| \leq d}(x) + \langle \nu, Bx \rangle_2 \right\} \\
\mu^k &= \nu^k + L^{-1}(Bx^k - b)
\end{align*}
$$

Since we have no structural constraints on $L$, we can choose any $L \succeq BH^{-1}B^T$. If $BH^{-1}B^T$ is sparse, an efficient choice is to let $L = BH^{-1}B^T$ and compute and store a sparse Cholesky factorization of $BH^{-1}B^T$. Updating $\mu^k$ then reduces to a forward and backward solve in each subsequent iteration.

The complexity of solving (38) depends highly on the structures of $H$ and $C$. If $H$ and $C$ are block-diagonal with sufficiently small blocks, then (38) can be solved efficiently in parallel. In the limiting case where $H$ and $C$ are diagonal, solving (38) reduces to one max and one min operation for each variable. For problems where solving (38) is computationally expensive, QP splitting 1 often gives better performance.

7.2 The distributed case

Here, we consider separable optimization problems of the form

$$
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{M} (f_i(x_i) + g_i(y_i)) \\
\text{subject to} & \quad Ax = y
\end{align*}
$$

where $f_i : \mathbb{E}_{H_i} \rightarrow \mathbb{R}$ is 1-strongly convex (w.r.t. $\mathbb{E}_{H_i}$), $g_i \in \Gamma_0(\mathbb{E}_{K_i})$, $x = (x_1, \ldots, x_M)$, $y = (y_1, \ldots, y_M)$, and

$$
A = \begin{bmatrix} A_{11} & \ldots & A_{1M} \\ \vdots & \ddots & \vdots \\ A_{M1} & \ldots & A_{MM} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_M \end{bmatrix}
$$

We further assume that many $A_{ij} = 0$. The non-zeros block entries of $A$ are indexed by the sets $N_i = \{j \in \{1, \ldots, M\} \mid A_{ij} \neq 0\}$, $M_i = \{i \in \{1, \ldots, M\} \mid A_{ij} \neq 0\}$.

We introduce the notation $x_{N_i} = (\ldots, x_i, \ldots)$ that stacks all $x_j$ with $j \in N_i$, and $A_{N_i} = [\ldots, A_{ij}, \ldots]$ that collects all $A_{ij} \neq 0$ in block-row $i$. This implies that (40) can equivalently be written as

$$
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{M} (f_i(x_i) + g_i(y_i)) \\
\text{subject to} & \quad A_{N_i}x_{N_i} = y_i, \text{ for all } i \in \{1, \ldots, M\}
\end{align*}
$$
We introduce dual variables $\mu_i$ for all equality constraints $A_N x_N = y_i$ and define the vectors $\mu_{M,i} = (\ldots, \mu_i, \ldots)$ that stacks all $\mu_i$ with $i \in M_j$. We also define the matrix $A_{M,j} = [\ldots A_{j,i}^T, \ldots]^T$ that collects all $A_j \neq 0$ in block-column $j$. This implies that the dual problem can be written as

$$\text{minimize} \ \sum_{i=1}^{M} (d_i(\mu_{M,i}) + g^T_i(\mu_i))$$

where

$$d_i(\mu_{M,i}) := f^*(\nabla_{A_{M,j}}^T \mu_{M,i}).$$

In Beck et al. (2013), it is shown how to compute a matrix $L$ that defines a space $\mathbb{E}_L$, using distributed computations only, on which

$$d(\mu) := \sum_{i=1}^{m} d_i(\mu_{M,i})$$

is 1-smooth. The procedure from Beck et al. (2013) is presented below:

$$L_{M,j} = \|A_{M,j}\|^2_{\min(H_j)}$$

$$L_i = \sum_{j \in N_i} L_{M,j}$$

$$L = \text{blkdiag}(L_1, \ldots, L_M)$$

This metric selection procedure relies on that $d_i$ is $\|A_{M,j}\|^2_{\min(H_j)}$ smooth w.r.t $\mathbb{E}_L$. From Proposition 29, we know that $d_i$ is majorized by a quadratic with Hessian $A_{M,j} H_j^{-1} A_{M,j}^T$, i.e.,

$$d_i(\mu_{M,i}) \leq d_i(\mu_{M,i}) + (\nabla d_i(\mu_{M,i}), \mu_{M,i} - \mu_{M,i})_2 + \frac{1}{2} \|\mu_{M,i} - \mu_{M,i}\|^2_{A_{M,j} H_j^{-1} A_{M,j}^T}.$$ 

This is obviously a tighter characterization of $d_i$ than the one used in Beck et al. (2013), i.e., that $d_i$ is $\|A_{M,j}\|^2_{\min(H_j)}$ smooth w.r.t $\mathbb{E}_L$. Using Proposition 29, the distributed metric selection procedure proposed in Beck et al. (2013) can be modified to yield a less conservative set from which $L$ may be chosen. The modified procedure is presented below:

$$L_{M,j} = \text{blkdiag}(\ldots, L_{M,j,i}, \ldots) \geq A_{M,j} H_j^{-1} A_{M,j}^T$$

$$L_i = \sum_{j \in N_i} L_{M,j,i}$$

$$L = \text{blkdiag}(L_1, \ldots, L_M)$$

where $L_{M,j,i}$ are sub-blocks of the same dimension as $\mu_i$. With this restriction, it is straightforward to verify that the fast dual forward backward splitting algorithm on $\mathbb{E}_L$ in distributed fashion with the same communication structure as the algorithm in Beck et al. (2013). Also, the proof in Beck et al. (2013) to show that $d$ is 1-smooth w.r.t $\mathbb{E}_L$, with $L$ computed as in (41)-(43) is straightforwardly generalized to show that $d$ is 1-smooth w.r.t $\mathbb{E}_L$ with $L$ as computed in (44)-(46). This is needed to guarantee convergence of the distributed fast dual forward-backward splitting algorithm.

By choosing $L_{M,j}$ appropriately (using some method from Section 6) an $L$ computed by (44)-(46) probably gives a better approximation of $d$ (in the sense of (3) with $\| \cdot \|$) than if $L$ is computed using (41)-(43). Computing $L$ from (44)-(46) is therefore likely to improve the convergence of the algorithm compared to using (41)-(43), i.e. the method in Beck et al. (2013).

**Remark 42.** Note that $L_{M,j}$ is restricted to be block-diagonal to facilitate a distributed implementation. However, we could superimpose additional internal structural constraints on each sub-block $L_{M,j,i}$ that may differ from one block to the next. For instance, each sub-block $L_{M,j,i}$ could be restricted to be diagonal, sparse, or full.

**Remark 43.** Applying the dual fast forward backward splitting on $\mathbb{E}_L$ with $L$ computed as in (44)-(46), enables for a distributed implementation (since the resulting $L$ is block-diagonal). If $g = I_{\mathbb{E}_L}(y)$, the problem has similar structure as QP-splitting 2 in Section 7.1. Then, as in QP-splitting 2, we can choose $L = AH^{-1}A^T$ since no prox-operation is needed for the dual variable update. This choice of $L$ gives a parallel algorithm where the primal variables are updated in parallel, and the dual variable updates require centralized computations. The centralized computations reduce to one forward and one backward substitution after an initial Cholesky factorization of $AH^{-1}A^T$.

### 8. Numerical Examples

In this section, we present numerical evaluations of the proposed methods. We apply the methods on a (small-scale) aircraft control problem and on large-scale separable randomly generated problems that are solved in distributed fashion.

**8.1 Aircraft control**

Here, we apply QP-splitting 1 and QP-splitting 2 from Section 7.1 to the AFTI-16 aircraft model in Kapasouris et al. (1990); Bemporad et al. (1997). As in Bemporad et al. (1997), the continuous time model from Kapasouris et al. (1990) is sampled using zero-order hold every 0.05 s. The system has four states $x = (x_1, x_2, x_3, x_4)$, two inputs $u = (u_1, u_2)$, and obeys the following dynamics

$$x^+ = \begin{bmatrix} 0.999 & -1.008 & -0.113 & -1.608 \\ -0.029 & -0.029 & -0.029 & -0.029 \\ 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 \end{bmatrix} x + \begin{bmatrix} 0.089 & -0.635 \\ -0.868 & -0.092 \\ -0.022 & -0.002 \end{bmatrix} u,$$

$$y = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} x$$

where $x^+$ denotes the state in the next time step. The system is unstable, the magnitude of the largest eigenvalue of the dynamics matrix is 1.313. The outputs are the attack and pitch angles, while the inputs are the elevator and flap/veron angles. The inputs are physically constrained to satisfy $|u_1| \leq 25^\circ$, $i = 1, 2$. The outputs are soft constrained to satisfy $-s_1 - 0.5 \leq y_1 \leq 0.5 + s_2$ and $-s_3 - 100 \leq y_2 \leq 100 + s_4$ respectively, where $s = (s_1, s_2, s_3, s_4) \geq 0$ are slack variables. The cost in each time step is

$$\ell(x, u, s) = \frac{1}{2} (x - x_r)^T Q (x - x_r) + u^T R u + s^T S s$$

where $x_r$ is a reference, $Q = \text{diag}(10^{-4}, 10^2, 10^{-3}, 10^2)$, $R = 10^{-2} I$, and $S = 10^3 I$. This gives a condition number of $10^{10}$ of the full cost matrix. Further, the terminal cost is $Q$, and the control and prediction horizon is $N = 10$. The numerical data in Tables 1 and 2 are obtained by following a reference trajectory on the output. The objective is to
change the pitch angle from 0° to 10° and then back to 0° while the angle of attack satisfies the output constraints $-0.5° \leq y_1 \leq 0.5°$. The constraints on the angle of attack limits the rate on how fast the pitch angle can be changed. The full optimization problem can be written on the form

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|z\|^2 + \langle \tau_1, z \rangle_2 \\
\text{subject to} & \quad B z = b \tau_t \\
& \quad d \leq C z \leq d
\end{align*}$$

(47)

where $x_t$ and $\tau_t$ changes from one sampling instant to the next.

In Table 1 we compare the performance of QP-splitting 1 and QP-splitting 2 from Section 7.1, when solving (47). We compare the performance of these when applied in the Euclidean setting $E_T$ and when applied on a space $E_L$, where $L$ is computed as suggested in Section 6. Since QP-splitting 1 in the Euclidean space is exactly the algorithm proposed in Patrinos and Bemporad (2014), and QP-splitting 2 in the Euclidean space is exactly the algorithm proposed in Richter et al. (2013), the comparison in Table 1 is a comparison to other methods proposed for embedded model predictive control in the literature. We also compare the performance of these methods with the performance of the ADMM-based method for embedded model predictive control proposed in O’Donoghue et al. (2013). Since the method in O’Donoghue et al. (2013) is based on ADMM, the $\rho$-parameter much be chosen. In Table 1 we provide simulation results for three different values of $\rho$, the best performing $\rho$ that we found ($\rho = 3$) and one larger and one smaller value of $\rho$. All algorithms are implemented in MATLAB and the numerical results in Table 1 are obtained by running the simulations on a Linux machine using a single core running at 2.9 GHz. To create an easily transferable and fair termination criterion, the optimal solution to each optimization problem $z^*$ is computed to high accuracy using an interior point solver. The optimality condition is then set to $\|z^k - z^*\|_2 / \|z^*\|_2 \leq 0.005$, where $z^k$ is the primal iterate in the algorithm at iteration $k$. This stopping criterion implies that a relative accuracy of 0.5% of the primal solution is required.

Table 1 indicates that applying the fast forward backward splitting method for this example using QP-splitting 1 on $E_L$, where $L$ is diagonal and computed by minimizing the (pseudo) condition number of $CH^{-1}C^T$ or $CM_{11}C^T$, outperforms with one to two orders of magnitude QP-splitting 1 applied in the Euclidean space, i.e. the algorithm in Patrinos and Bemporad (2014). From Table 1 we also read that QP-splitting 2 applied on $E_L$ with $L = BH^{-1}B^T$ for this example outperforms the algorithm in Richter et al. (2013), i.e. QP-splitting 2 on the Euclidean space, with more than three orders of magnitude. We also see that the fast dual forward-backward splittings methods applied on appropriate spaces converge considerably faster than the method in O’Donoghue et al. (2013).

In Table 2, we compare different solvers implemented in C. For QP-splitting 1 and QP-splitting 2, we generate C code that take the reference trajectory and the initial state as inputs. Compared to the corresponding MATLAB implementations in Table 1, the generated C code is more than 20 times faster. Also, the two different QP-splitting have similar performance. These implementations are compared to FORCES and MOSEK. FORCES, see Domahidi et al. (2012), is an optimized interior point C code generator for MPC problems. The structure of the MPC problem is exploited to significantly reduce the computational time when solving the KKT-system in each iteration. The comparison also includes MOSEK, which is a general commercial QP-solver that does not have the advantage of generating code for this specific problem beforehand. Table 2 reveals that the proposed fast dual forward-backward splitting methods are at least two to three times faster than FORCES when applied to this aircraft problem. Table 2 also shows that the fast dual forward-backward splitting methods and FORCES outperform the general purpose commercial C solver MOSEK with more than one order of magnitude.

### Table 1: Comparison to other first-order methods, all implemented in MATLAB. FDFBS refers to fast dual forward backward splitting, ADMM refers to alternating direction method of multipliers, QPi refers to QP-splitting i (for i = 1, 2).

<table>
<thead>
<tr>
<th>Algorithm/splitting/(reference)</th>
<th>Parameters</th>
<th>exec time (ms)</th>
<th>nbr iters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>avg.</td>
<td>max</td>
<td>avg.</td>
</tr>
<tr>
<td>FDFBS/QP1</td>
<td>$L$ diag fr. (30) with $Q = CM_{11}C^T$</td>
<td>1.4</td>
<td>7.1</td>
</tr>
<tr>
<td>FDFBS/QP1</td>
<td>$L$ diag fr. (29) with $Q = CH^{-1}C^T$</td>
<td>1.2</td>
<td>5.8</td>
</tr>
<tr>
<td>FDFBS/QP1/Patrinos and Bemporad (2014)</td>
<td>$L = |CM_{11}C^T|_2$</td>
<td>98.5</td>
<td>673.0</td>
</tr>
<tr>
<td>FDFBS/QP1/Patrinos and Bemporad (2014)</td>
<td>$L = |CH^{-1}C^T|_2$</td>
<td>98.9</td>
<td>679.4</td>
</tr>
<tr>
<td>FDFBS/QP2</td>
<td>$L = BH^{-1}B^T$</td>
<td>2.3</td>
<td>12.1</td>
</tr>
<tr>
<td>FDFBS/QP2/Richter et al. (2013)</td>
<td>$L = |BH^{-1}B^T|_2$</td>
<td>4713.9</td>
<td>28411</td>
</tr>
<tr>
<td>ADMM/QP1/O’Donoghue et al. (2013)</td>
<td>$\rho = 0.3$</td>
<td>193.9</td>
<td>920.6</td>
</tr>
<tr>
<td>ADMM/QP1/O’Donoghue et al. (2013)</td>
<td>$\rho = 3$</td>
<td>29.7</td>
<td>142.2</td>
</tr>
<tr>
<td>ADMM/QP1/O’Donoghue et al. (2013)</td>
<td>$\rho = 30$</td>
<td>35.1</td>
<td>264.4</td>
</tr>
</tbody>
</table>

### Table 2: Comparison to state-of-the-art solvers, all implemented in C.

<table>
<thead>
<tr>
<th>Alg.(/split.)</th>
<th>Parameters</th>
<th>exec time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>avg.</td>
<td>max</td>
</tr>
<tr>
<td>FDFBS/QP1</td>
<td>$L$ diag fr. (30) with $Q = CH^{-1}C^T$</td>
<td>0.061</td>
</tr>
<tr>
<td>FDFBS/QP2</td>
<td>$L = BH^{-1}B^T$</td>
<td>0.079</td>
</tr>
<tr>
<td>FORCES</td>
<td>-</td>
<td>0.347</td>
</tr>
<tr>
<td>MOSEK</td>
<td>-</td>
<td>4.9</td>
</tr>
</tbody>
</table>

8.2 Distributed examples

Here, we apply the fast dual forward-backward splitting method to randomly generated dynamical systems with a sparse dynamic interaction. The dynamic interaction structure is decided using the method in (Kraning et al., 2013 §6.1) and the number of subsystems are 500, 2000, and 8000 respectively. The resulting average degree of the generated interconnection structures are 2.27, 2.23, and 2.23 respectively. The number of states in each subsystem
Table 3: Numerical evaluation for fast dual forward-backward splitting (FDFBS) applied on different spaces $E_L$, including the space selection from Beck et al. (2013) and the Euclidean space. The comparison also includes the dual Newton CG method in Kozma et al. (2014).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
<th># so/vars./constr.</th>
<th># communication rounds</th>
<th>avg. exec. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDFBS</td>
<td>$L = AH^{-1}A^T$</td>
<td>500/87k/246k</td>
<td>local</td>
<td>global</td>
</tr>
<tr>
<td>FDFBS</td>
<td>$L_{fr.}$ (44)-(46), min trace</td>
<td>500/87k/246k</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FDFBS</td>
<td>$L_{fr.}$ (44)-(46), $|\cdot|_2$-equil.</td>
<td>500/87k/246k</td>
<td>523.7</td>
<td>774</td>
</tr>
<tr>
<td>Beck et al. (2013)</td>
<td>$L_{fr.}$ (41)-(43)</td>
<td>500/87k/246k</td>
<td>4789.8</td>
<td>7558</td>
</tr>
<tr>
<td>FDFBS</td>
<td>$L = |AH^{-1}A^T|_2I$</td>
<td>500/87k/246k</td>
<td>6114.7</td>
<td>6556</td>
</tr>
<tr>
<td>Kozma et al. (2014)</td>
<td>$\epsilon = 10^{-4}, \mu = 0.8, \sigma = 0.3$</td>
<td>500/87k/246k</td>
<td>6661.1</td>
<td>28868</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
<th># so/vars./constr.</th>
<th># communication rounds</th>
<th>avg. exec. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDFBS</td>
<td>$L = AH^{-1}A^T$</td>
<td>2000/351k/993k</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>FDFBS</td>
<td>$L_{fr.}$ (44)-(46), min trace</td>
<td>2000/351k/993k</td>
<td>356.8</td>
<td>652</td>
</tr>
<tr>
<td>FDFBS</td>
<td>$L_{fr.}$ (44)-(46), $|\cdot|_2$-equil.</td>
<td>2000/351k/993k</td>
<td>1138.0</td>
<td>1666</td>
</tr>
<tr>
<td>Beck et al. (2013)</td>
<td>$L_{fr.}$ (41)-(43)</td>
<td>2000/351k/993k</td>
<td>2530.5</td>
<td>3218</td>
</tr>
<tr>
<td>FDFBS</td>
<td>$L = |AH^{-1}A^T|_2I$</td>
<td>2000/351k/993k</td>
<td>4474.9</td>
<td>4608</td>
</tr>
<tr>
<td>Kozma et al. (2014)</td>
<td>$\epsilon = 10^{-4}, \mu = 0.8, \sigma = 0.3$</td>
<td>2000/351k/993k</td>
<td>6461.4</td>
<td>20624</td>
</tr>
</tbody>
</table>

The evaluation in Table 3 is obtained by generating 200 feasible random initial conditions from the state constraint set for each of the systems. The corresponding control problems are solved using the different algorithms, each utilizing 12 parallel cores. For each problem size, we compare the performance when the $L$-matrix (that defines $E_L$) is computed using (44)-(46), where in (44) we use different methods from Section 6. We have chosen to include the cases where the local $L_M$, in (44) are computed by trace-minimization and 2-norm equilibration only. Computing $L_M$, by minimizing the (pseudo) condition number is not included due to computational complexity, and equilibration in 1-norm and $\infty$-norm is also excluded since the performance is very similar to the 2-norm equilibration case. The performance of these methods is compared to the performance of the method from Beck et al. (2013), i.e., when the $L$-matrix is computed using (41)-(43), and to fast dual decomposition with the optimal parameter selection given by $L = \|AH^{-1}A^T\|_2I$. We see that the parallel algorithm has fewest communications rounds (which is two times the iteration count for these algorithms), then the algorithm with $L$ computed using trace minimization, then $L$ computed using equilibration, thereafter the method from Beck et al. (2013), and finally fast dual decomposition with a global step-size. This is indeed expected, since as we traverse up the list, the approximation used in the algorithm that is defined by the $L$-matrix matches better and better the smooth part of the dual function, $d$. Note that the algorithm with fewest communication rounds is a parallel algorithm, i.e. $L$ has no block-diagonal structure. For the algorithm where the $L$-matrix is computed using local trace-minimization problems, the sub-blocks in $L$ have a sparsity pattern that is not diagonal, while the remaining algorithms have diagonal $L$-matrices. The added flexibility in having more non-zeros elements in $L$ gives a reduced number of iterations. We also note that the dual Newton CG method in Kozma et al. (2014) performs considerably worse than the other methods on this example.
The execution times in Table 3 are pure execution times for the DMPC scheme, i.e., without offline pre-processing steps such as metric selection and factorization of matrices. To solve a semi-definite program, or to factorize a very large matrix to find the metric $L$ is usually only computationally beneficial if the resulting $L$ can be used to solve many optimization problems, as is the case in DMPC. Thus the first two rows in Table 3 of every problem size are merely for DMPC applications, or other applications with the same characteristic. The computational complexity for equilibration is only slight higher than for the method in Beck et al. (2013). Both these methods are considerably less computational demanding than computing $\|AH^{-1}A^T\|^2_2$ which is required to compute a global step-size. This implies that these methods are very useful when selecting metric also in general distributed optimization.

9. CONCLUSIONS

We have proposed several methods, with different computational complexity, to compute spaces on which to perform fast dual forward-backward splitting, when the primal optimization problem is strongly convex. We have evaluated these methods by applying them to an aircraft control problem and to distributed optimization problems. For the most ill-conditioned problem, the numerical evaluations show that it is possible to reduce the computational effort by up to three orders of magnitude compared to applying the algorithms on the Euclidean space.

REFERENCES


