Hamiltonian Decomposition for Model Predictive Control

Eduardo Poupard∗ William P. Heath∗

∗School of Electrical and Electronic Engineering, The University of Manchester, Sackville Street Building, Manchester, UK, M13 9PL
(e-mail: eduardo.navarropoupard@manchester.ac.uk, william.heath@manchester.ac.uk).

Abstract: We present the application of an eigenvalue decomposition for the solution of the optimal control problem in model predictive control (MPC). This approach can be used as an alternative to the Riccati recursion that occurs within interior-point solvers, which are used to compute the solution of the optimization problem in constrained MPC. We first demonstrate that a known method applied to the finite-horizon linear quadratic regulator (LQR) for free-final state can be extended to the LQ tracking problem. Likewise, the receding-horizon idea is implemented on these two optimal control problems. In addition, the implications for online implementation are discussed.

Keywords: Optimal Control, Hamiltonian Matrix, Predictive Control, Optimization Algorithms, On-line Control.

1. INTRODUCTION

In 1960 Kalman showed that the linear quadratic regulator (LQR) can be solved by applying the Riccati equation backwards in time. One technique used to solve this equation was proposed by MacFarlane [1963], where an eigenvalue decomposition is applied. This method is limited since it can be only used for infinite-horizon problems. Vaughan ([1968] and [1970]) generalized this approach (for continuous and discrete-time systems respectively), where the solution can also be obtained for finite-horizon problems.

In Vaughan’s work the prediction horizon $N$ is included explicitly within the algorithm. In this paper we begin by extending this idea to unconstrained Model Predictive Control (MPC) as preliminary to our main contribution. An unconstrained MPC is basically the implementation of the receding-horizon concept on the finite-horizon LQR problem. In MPC only $u_0$ is required, instead of the sequence of optimal control inputs. Therefore, the eigenvalue decomposition is computed only for $u_0$ and with a computational load independent of the horizon.

MPC is the only control technique that can deal effectively with state and input constraints and as a consequence has been widely applied in the industry (Maciejowski [2002]). Real time implementation of MPC requires the solution of a quadratic program at each sample. There has been considerable recent interest in finding efficient online solvers. Mainly three methods exist for solving this problem: (i) the fast gradient method (Nesterov [2003]), which makes use of the search direction’s first-order information. Recent contributions using fast gradient on embedded systems have been implemented on programmable logic controllers (PLCs) (Kufaol et al. [2014]), and on field-programmable gate arrays (FPGAs) (Jerez et al. [2013]); (ii) the active-set method (Fletcher [1987]), which considers only the set of inequality constraints that are active. An optimization solver using this method is proposed in Ferreau et al. [2008] and it is also implemented on a fast online application in Wills et al. [2008]; (iii) the interior-point method, IPM (Wright [1997]), which is based on Newton’s step method. It solves a fixed-size linear system of equations at every sample, distinguishing it from the active set method whose structure size can change. The replacement of inequality constraints with a barrier function for MPC algorithms has been studied by Wills and Heath [2004], and can be used for efficient online solvers (Wang and Boyd [2010]).

Rao et al. [1998] use the Riccati equation to factorize the search direction’s linear system of equations. In their work, the Hessian matrix is considered sparse (states and inputs considered in the optimization vector) and the computational cost is linear in the horizon $N$. Rao et al.’s work achieves this through one backward substitution, solving the Riccati equation and the modified residual recursively. This is followed by a forward substitution in order to recover the search direction vector. To this day, this method is a matter of interest for on-line MPC research, being analyzed for different contexts. For instance, it serves as a basis for efficient optimization solvers by Frison and Jorgensen [2013], and as a benchmark for comparison by Cantoni et al. [2016].

In this paper, our main contribution is to propose an equivalent algorithm that predominantly utilizes an eigenvalue decomposition to factorize the search direction’s system within the IPM algorithm. The paper is organized as follows. Section 2 reviews the definition of the free-final state LQR problem and its solution via an eigenvalue decomposition (Vaughan’s work), followed by the application of the receding-horizon idea. We then demonstrate that this can be extrapolated to the LQ tracking problem (section 3) and to the MPC approach (section 4).
2. EIGENVALUE DECOMPOSITION IN THE LQR PROBLEM

2.1 Problem formulation

The following description is standard [Lewis et al., 2012, p.19-35]. The mathematical formulation of the free-final state problem for the finite-horizon LQR is described below

\[
\min_{x,u} \frac{1}{2} x^T \Sigma x + \frac{1}{2} \sum_{k=0}^{N-1} \left( x^T_0 Q x_k + u^T_k R u_k \right) \\
\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k, \quad x_0 = x(0) \tag{1}
\]

where \( x_k \in \mathbb{R}^{n_x} \) is the state variable with sequence \( X = (x_0, x_1, \ldots, x_N) \), \( u_k \in \mathbb{R}^{n_u} \) is the input variable with sequence \( U = (u_0, u_1, \ldots, u_{N-1}) \) and \( N \) is the prediction horizon. For simplicity, we will consider a time-invariant system. Weighting matrices \( Q, P \geq 0 \) and \( R > 0 \) are symmetric. The Hamiltonian function for (1) is

\[
H^k = \frac{1}{2} (x^T Q x_k + u^T_k R u_k) + \lambda_k^T (Ax_k + Bu_k), \tag{2}
\]

where \( \lambda_k \in \mathbb{R}^{n_x} \) is the Lagrange multiplier. By applying the necessary conditions for optimality, the coupled state and costate equations can be described as follows

\[
\begin{bmatrix}
  x_{k+1} \\
  \lambda_k
\end{bmatrix} =
\begin{bmatrix}
  A - BR^{-1}B^T \\
  QA - QA^{-1}B^T + QA^{-1}BR^{-1}B^T
\end{bmatrix}
\begin{bmatrix}
  x_k \\
  \lambda_{k+1}
\end{bmatrix}, \tag{3}
\]

where the coefficient matrix of (4) is called \( H_m \) only depends on the number of states \( n_x \), as shown in the coefficient matrix of (4).

Remark 2.1. The case of \( A \) being singular is beyond the scope of this paper. However, this problem can be circumvented by using the generalized eigenvalue concept as presented by Gaalman [1980] and Pappas et al. [1980]. We carry out the following change of variable

\[
\begin{bmatrix}
  x_k \\
  \lambda_k
\end{bmatrix} = \mathcal{V} \begin{bmatrix}
  w_k \\
  y_k
\end{bmatrix}, \tag{4}
\]

in system (4) by using transformation (9)

\[
\begin{bmatrix}
  w_k \\
  y_k
\end{bmatrix} = \mathcal{D} \begin{bmatrix}
  w_{k+1} \\
  y_{k+1}
\end{bmatrix}. \tag{5}
\]

The well-conditioned solution of (11) is easy to obtain since \( \mathcal{D} \) is diagonal

\[
\begin{bmatrix}
  w_N \\
  y_N
\end{bmatrix} = \begin{bmatrix}
  M^{-(N-k)} \\
  M^{-(N-k)}
\end{bmatrix} \begin{bmatrix}
  w_k \\
  y_k
\end{bmatrix}. \tag{6}
\]

Using final condition (5) in (10) gives

\[
V_{21} w_N + V_{22} y_N = P (V_{11} w_N + V_{12} y_N). \tag{7}
\]

Solving for \( y_N \) in terms of \( w_k \) gives

\[
y_k = - (V_{22} - PV_{12})^{-1} (V_{21} - PV_{11}) w_k. \tag{8}
\]

Using (12) we obtain

\[
y_k = T_k w_k, \tag{9}
\]

where

\[
T_k = M^{-(N-k)} TM^{-(N-k)} \tag{10}
\]

The next step is to solve \( w_k \) in terms of \( x_k \) from (11) and

\[
w_k = (V_{11} + V_{12} T_k)^{-1} x_k. \tag{11}
\]

Remark 2.2. Algorithm 1 is non-recursive for the unconstrained free-final state MPC (setting \( k = 0 \)). The algorithm’s computational cost is less than the one using the Riccati equation since the only part that depends on the horizon \( N \) is the optimal control input \( u_k \). For this, the Riccati equation (7) is computed backwards. With our proposal, one only has to set \( k = 0 \) and compute once (15), (18) and then \( u_0 \) (8) at the expense of the eigenvalue decomposition (9).

Remark 2.3. Algorithm 1 is non-recursive for the unconstrained free-final state MPC (setting \( k = 0 \)). The algorithm’s computational cost is less than the one using the Riccati equation since the only part that depends on the horizon \( N \) is the optimal control input \( u_k \). For this, the Riccati equation (7) is computed backwards. With our proposal, one only has to set \( k = 0 \) and compute once (15), (18) and then \( u_0 \) (8) at the expense of the eigenvalue decomposition (9).

2.2 Application of the eigenvalue decomposition to LQR

The following is based on Vaughan [1970]. Matrix \( H_m \) from (4) is called symplectic since it satisfies \( H_m^T \mathcal{J} H_m = \mathcal{J} \), where

\[
\mathcal{J} = \begin{bmatrix}
  0 & I_n \\
  -I_n & 0
\end{bmatrix},
\]

and \( I_n \) is the \( n \times n \) identity matrix. The eigenvalues of symplectic matrices come in reciprocal pairs, namely if \( \mu \) is an eigenvalue of \( H_m \), so is \( 1/\mu \). Assuming that \( H_m \) is diagonalizable (semisimple eigenvalues), therefore it can be expressed as

\[
H_m = \mathcal{V} \mathcal{D} \mathcal{V}^{-1}, \tag{9}
\]

where

\[
\mathcal{D} = \begin{bmatrix}
  M & M^{-1} \\
  M^{-1} & M
\end{bmatrix}, \mathcal{V} = \begin{bmatrix}
  V_1 \\
  V_2
\end{bmatrix}, \mathcal{V}^{-1} = \begin{bmatrix}
  V_1 & V_2 \\
  V_2 & V_1
\end{bmatrix}, \tag{10}
\]

where \( M \in \mathbb{R}^{n_x \times n_x} \) is diagonal and has the eigenvalues outside the unit circle and columns of \( \mathcal{V} \in \mathbb{R}^{2n_x \times 2n_x} \) are the eigenvectors.

2.3 Direct solution for Receding Horizon Control (RHC)

Loosely speaking, receding-horizon control (RHC) is the theoretical framework for MPC (herefore still unconstrained) which only requires \( u_0 \) of the computed sequence \( U \). For this, the Riccati equation (7) is computed backwards. With our proposal, one only has to set \( k = 0 \) and compute once (15), (18) and then \( u_0 \) at the expense of the eigenvalue decomposition (9).

Remark 2.4. Algorithm 1 is non-recursive for the unconstrained free-final state MPC (setting \( k = 0 \)). The algorithm’s computational cost is less than the one using the Riccati equation since the only part that depends on the horizon \( N \) is the optimal control input \( u_k \). For this, the Riccati equation (7) is computed backwards. With our proposal, one only has to set \( k = 0 \) and compute once (15), (18) and then \( u_0 \) at the expense of the eigenvalue decomposition (9).
Algorithm 1 RHC Solution for LQR free-final state

1: \( H_m = VDV^{-1} \) \( \triangleright \) EV Decomposition
2: \( T = -(V_{22} - PV_{12})^{-1} (V_{21} - PV_{11}) \)
3: \( T_1 = M^{-(N-1)}T M^{-(N-1)} \)
4: \( P_1 = (V_{21} + V_{22} T_1) (V_{11} + V_{12} T_1)^{-1} \)
*Optimal control input for RHC:
5: \( u_0 = -(B^T P_1 B + R)^{-1} B^T P_1 A x_0 \) \( \triangleright \) \( x_0 \) given

3. EXTENSION TO THE LQ TRACKING PROBLEM

An important control strategy is to make a given system follow a desired trajectory. We present here an extension from Vaughan’s method to the LQ tracking problem, which will be used as a basis for our core approach in the next section.

3.1 Problem formulation

The following results are standard [Lewis et al., 2012, p.190-194]. The mathematical description of the LQ tracking problem is as follows

\[
\min_{x, u} \frac{1}{2} (x_N - x_N^r)^T P (x_N - x_N^r)
+ \frac{1}{2} \sum_{k=0}^{N-1} ((x_k - x_k^r)^T Q (x_k - x_k^r) + u_k^T R u_k) \\
\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k, \quad x_0 = x(0). \quad (19)
\]

where \( x^r \) is the reference signal from \( k = 0, \ldots, N \). The Hamiltonian function for (19) is

\[
H^k = \frac{1}{2} ((x_k - x_k^r)^T Q (x_k - x_k^r) + u_k^T R u_k) \\
+ \lambda_{k+1}^T (Ax_k + Bu_k). \quad (20)
\]

Then, using necessary conditions we can obtain the Hamiltonian system corresponding to the LQ tracking problem

\[
\begin{bmatrix}
\lambda_k \\
x_k
\end{bmatrix} = H_m \begin{bmatrix}
\lambda_{k+1} \\
x_{k+1}
\end{bmatrix} + \begin{bmatrix}
0 \\
-Q
\end{bmatrix} x_k^r, \quad (21)
\]

where \( H_m \) was already defined in (4). The boundary condition (5) for this problem is

\[
\lambda_N = P (x_N - x_N^r). \quad (22)
\]

3.2 Eigenvalue decomposition for the LQ tracking problem

First, we will solve the finite-horizon LQ tracking problem (19), and then the receding horizon idea will follow. In this regard, the eigenvalue decomposition of \( H_m \) (9) still applies for this problem; and with the change of variable (10), system (21) becomes

\[
\begin{bmatrix}
w_k \\
y_k
\end{bmatrix} = D \begin{bmatrix}
w_{k+1} \\
y_{k+1}
\end{bmatrix} + G x_k^r, \quad (23)
\]

\[D = \begin{bmatrix}
M^{-(N-k)} \\
M^{-(N-k)}
\end{bmatrix} \begin{bmatrix}
w_N \\
y_N
\end{bmatrix} + \begin{bmatrix}
-S_{k-N-1}^w \\
S_{k-N-1}^y
\end{bmatrix} \quad (25)
\]

where \( S_{k-N-1}^w, S_{k-N-1}^y \in \mathbb{R}^{n_x} \) are forcing functions defined as

\[
S_{k-N-1}^w = \sum_{j=k}^{N-1} M^{k-j-1} G^w x_{N-j+(k-1)}, \quad (26)
\]

\[
S_{k-N-1}^y = \sum_{j=k}^{N-1} M^{-(N-j-1)} G^y x_{N-j+(k-1)}^r. \quad (27)
\]

The final condition (22) and its relation with (23) at the final time \( N \) is

\[
y_N = -(T_w^{-1} T_y) w_N - (T_w^{-1}) P x_N^r. \quad (28)
\]

Solving for \( y_N \) in terms of \( w_N \) gives

\[
y_N = -(T_w^{-1} T_y) w_N - (T_w^{-1}) P x_N^r, \quad (29)
\]

where \( T_w = (V_{22} - PV_{12}) \) and \( T_y = (V_{21} - PV_{11}) \). Using (25) and after a few steps we obtain

\[
y_k = T_k w_k + v_k, \quad (30)
\]

where

\[
T_k = T_k^N M^{-(N-k)}, \quad (31)
\]

\[
v_k = -(T_w^{-1} T_y) w_k - (T_w^{-1}) P x_k^r, \quad (32)
\]

The next step is to solve \( w_k \) in terms \( x_k \) from (23) and (28)

\[
T_k = (T_{k+1}^N T_{k+1}^r)^{-1} (x_k - x_k^r) + v_k^r, \quad (33)
\]

\[
T_k = (T_{k+1}^N T_{k+1}^r)^{-1} (x_k - x_k^r) + v_k^r, \quad (34)
\]

\[
S_{k-N}^y = G^y x_k^r, \quad (35)
\]

The optimal control input for the LQ tracking problem

\[
(36)
\]

Algorithm 2 Eigenvalue decomposition for LQ tracking

1: \( H_m = VDV^{-1} \) \( \triangleright \) EV Decomposition
2. *Set:** \( T_{k+1} = P \) and \( \pi_{k+1} = P x_N^r \)
3. for \( k = N-1 \) to 0 do \( \triangleright \) Offline computation
4. \( T_{wk} = (V_{22} - T_{k+1} V_{12}), T_{y_k} = (V_{21} - T_{k+1} V_{11}) \)
5. \( T_k = -M^{-1} (T_{wk}^{-1} T_y) M^{-1} \)
6. \( T_k = (V_{11} + V_{12} T_k), T_k = (V_{21} + V_{22} T_k) \)
7. \( T_k = T_k^N (T_k^r)^{-1} \) \( \triangleright \) Gain Sequence
8. \( S_k^w = G^w x_k^r, S_k^y = G^y x_k^r \)
9. \( v_k = -T_k S_k^w - M^{-1} (T_{wk}^{-1} \pi_{k+1}) + S_k^y \)
10. \( v_k = V_{12} v_k, v_k^r = V_{22} v_k \)
11. \( \pi_k = -T_k v_k^r + v_k^r \) \( \triangleright \) Residual
12. end for

*Solution of the system given \( x_0 \):
13. for \( k = 0 \) to \( N-1 \) do \( \triangleright \) Online Computation
14. \( u_k = -(B^T T_{k+1} B + R)^{-1} B^T [T_{k+1} A x_k + \pi_{k+1}] \)
15. \( x_{k+1} = A x_k + B u_k \)
16. end for

Remark 3.1. Algorithm 2 presents a novel approach to solve LQ tracking problems, as the one encountered in
Lewis et al. [2012]. The main difference lies in how $T_k$ and $v_k$ are obtained. In algorithm 2, the $T_k$ and $v_k$ are computed backwards in time using the eigenvalue decomposition. This decomposition is derived outside the off-line for-loop and independent of the horizon $N$. With the known method, $T_k$ is computed with the Riccati equation backwards in time along with the residual $v_k$.

3.3 Direct solution for Receding Horizon Control (RHC)

We implement the receding-horizon idea to previous results in order to implement it on the unconstrained MPC tracking problem, where solely $u_0$ is needed.

**Algorithm 3 RHC Solution for LQ tracking problem**

1. $H_m = VDV^{-1}$ \[\Rightarrow\] EV Decomposition
2. $T_w = (V_{22} - PV_{12}), T_y = (V_{21} - PV_{11});$
3. $T_1 = -M^{-1}(N-1)(T_{w1}T_{y1});$
4. $T_1 = T_1M^{-(N-1)}1;$
5. $T_1 = (V_{11} + V_{12}T_{y1}), T_{11} = (V_{21} + V_{22}T_{y1});$
6. $T_1 = T_{11}^{-1}(T_{11}^{-1});$ \[\Rightarrow\] Gain Sequence
7. $S_{N-1}^{y} = \sum_{j=1}^{N-1}M^{-j}G^{j}x_{N-j}^{y};$
8. $S_{N-1}^{x} = \sum_{j=1}^{N-1}M^{-j-N}G^{j}x_{N-j}^{x};$
9. $v_1 = -T_{11}S_{N-1}^{x} + M^{-N-1}(T_{w1}P_{x_{N}^{r}}) + S_{N-1}^{y};$
10. $v_1^{T} = V_{12}v_1, v_1^{y} = V_{22}v_1;$
11. $\pi_1 = -T_{11}v_1^{T} + v_1^{T};$ \[\Rightarrow\] Residual

*Optimal control input for RHC:

12. $u_0 = -(B^{T}T_{11}B + R)^{-1}B^{T}[T_{11}A_{x0} + \pi_1];$

Remark 3.2. Algorithm 3 is for the unconstrained MPC tracking problem (setting $k = 0$). Similar to algorithm 1, the computational cost of the eigenvalue decomposition $H_m$ depends only on the states $n_x$. The only terms that depend on the horizon $N$ are $T_1$ (on the diagonal matrix $M$, steps 3 and 4) and the residual $v_1$ (as a recursive procedure).

4. IMPLEMENTATION OF EIGENVALUE DECOMPOSITION ON THE INTERIOR-POINT METHOD

Rao et al. [1998] showed that, within the Interior-Point Method (IPM), the discrete time-varying Riccati equation (7) arises naturally in the factorization of the search parameter system. In this section we will present an alternative method that builds upon these results to implement the idea of subsection 3.2 on MPC. We then discuss its implications and potential for online implementation.

4.1 Problem formulation

We first define the constrained free-final state linear MPC problem as follows

$$\min_{x, u} \frac{1}{2}x_{N}^{T}P_{x}x_{N} + \frac{1}{2}\sum_{i=0}^{N-1}(x_{i}^{T}Qx_{i} + u_{i}^{T}Ru_{i})$$

s.t. $x_{i+1} = A_{x}x_{i} + Bu_{i}$, $x_{0} = x(0)$

$$x_{low} \leq x_{i} \leq x_{high}$$

$$u_{low} \leq u_{i} \leq u_{high},$$

where affine inequality constraints are defined by minimum and maximum value in the states ($x_{low}, x_{high}$) and inputs ($u_{low}, u_{high}$), all remaining parts were already defined in (1). Therefore we can now generalize (34) into a convex quadratic problem that is defined as follows [Boyd and Vandenberghe, 2004, Sect. 4.4]

$$\min_{\theta} f(\theta) = \frac{1}{2}\theta^{T}H\theta + g^{T}\theta$$

s.t. $F\theta = b$

$$C\theta \leq d,$$

where $\theta \in \mathbb{R}^{n}$ is defined as a decision vector, the Hessian matrix $H \in \mathbb{R}^{n \times n}, F \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{p \times n}$. The overall aim is to find a decision vector $\theta$ which minimizes the quadratic performance index, $f(\theta)$, subject to both constraints. The necessary conditions for optimality of (35), known as Karush-Kuhn-Tucker (KKT) conditions are

$$H\theta + F^{T}\lambda + C^{T}\mu + g = 0,$$

$$F\theta - b = 0,$$

$$C\theta - d + t = 0,$$

$$MTe = \sigma e,$$

$$\mu, t \geq 0,$$

where $\lambda$ and $\mu$ are the so-called Lagrange multipliers of the equality and inequality constraints respectively and $t$ is a slack variable. Then $M = \text{diag}(\mu_1, \mu_2, ..., \mu_p)$, $\mathcal{T} = \text{diag}(t_1, t_2, ..., t_p)$ and $e = (1, 1, ..., 1)^{T}$. Additionally, $\sigma \in [0, 1]$ is the centering parameter and the duality gap is $\eta = \mu^{T}t/p$. The term $\sigma e$ plays a stabilizing role to allow the algorithm to converge steadily towards the solution of (35) [Wright, 1997, p.36-40].

We will use the IPM algorithm to solve system (36), in particular Mehrotra’s Predictor-Corrector algorithm (Mehrotra [1992]), where Newton’s step equation is $J(\theta, \lambda, \mu, t)$ or

$$[H \ F^{T} \ C^{T} \ 0] \Delta \theta$$

$$[F \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \theta \ \\
C \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \Omega \ \\
0 \ 0 \ \mathcal{T} \ \\
\Delta_{\lambda} \ \\
\Delta_{\mu} \ \\
\Delta_{t} \ \\
\Omega] = -Y \Psi,$$

where $Y = H\theta + F^{T}\lambda + C^{T}\mu + g$, $\Phi = F\theta - b, \Psi = C\theta - d + t$ and $\Omega = MTe - \sigma e$.

4.2 Hamiltonian matrix on IPM

Using the sparse approach (states and inputs as optimal variables) in system (37) we can deduce the Hamiltonian system via block factorization. Hence, taking only equations with iterations $k$ and $k+1$, i.e.

$$[Q \ 0 \ 0 \ -I \ A^{T} \ \omega^{T} \ 0 \ \\
0 \ 0 \ R \ 0 \ B^{T} \ \phi^{T} \ 0 \ \\
A - I \ B \ 0 \ 0 \ 0 \ 0 \ \\
\omega \ 0 \ \phi \ 0 \ 0 \ 0 \ I \ \\
0 \ 0 \ 0 \ 0 \ 0 \ \mathcal{T}_{k} \ M_{k} \ \\
\Delta_{x_{k}} \ \\
\Delta_{\lambda_{k+1}}] = \Delta_{x_{k}}' \ \\
\Delta_{\lambda_{k}}' \ \\
\Delta_{\mu_{k+1}} \ \\
\Delta_{t}'] = -[egin{bmatrix} \Delta_{x_{k}} \\
\Delta_{\lambda_{k+1}} \\
\Delta_{\mu_{k+1}} \\
\Delta_{t} \end{bmatrix}]$$

where $\Delta_{x_{k}} = 0$ and the other residuals are
\[ r_k^1 = Qx_k - \lambda_k + A^T\lambda_{k+1} + \omega^T\mu_k, \]
\[ r_k^2 = R u_k + B^T\lambda_{k+1} + \phi^T\mu_k, \]
\[ r_k^3 = \omega x_k + \phi u_k - \nu + t_k, \]
\[ r_k^4 = M_k T_k e, \]

of appropriate dimensions and \( \omega = [-I, I, 0, 0]^T \in \mathbb{R}^{2(n_u + n_x) \times n_x} \), \( \phi = [0, 0, -I, I]^T \in \mathbb{R}^{2(n_u + n_x) \times n_x} \), and \( v = [-x_{low}, x_{high}, -x_{low}, x_{high}]^T \in \mathbb{R}^{2(n_u + n_x)} \). Next step is to eliminate \( \Delta \mu_k \) and \( \Delta t_k \)

\[
\begin{bmatrix}
Q_k & 0 & 0 & -I & A^T \\
0 & R_k & 0 & 0 & B^T \\
A & -I & B \end{bmatrix} \begin{bmatrix}
\Delta x_k \\
\Delta u_k \\
\Delta \lambda_k \\
\Delta \lambda_{k+1} \\
\Delta \lambda_{k+1} \\
\end{bmatrix} = - \begin{bmatrix}
r_k^1 \\
r_k^2 \\
r_k^3 \\
r_k^4 \\
\end{bmatrix},
\]

where
\[
\Delta \mu_k = L_k (\omega \Delta x_k + \phi \Delta u_k + r_k^4 - M_k^{-1} r_k^5),
\]
\[
\Delta t_k = -M_k^{-1} (T_k \Delta \mu_k + r_k^5),
\]

\[
\Delta \mu_k = L_k (\omega \Delta x_k + \phi \Delta u_k + r_k^4 - M_k^{-1} r_k^5),
\]
\[
\Delta t_k = -M_k^{-1} (T_k \Delta \mu_k + r_k^5),
\]

where
\[
\Delta \mu_k = L_k (\omega \Delta x_k + \phi \Delta u_k + r_k^4 - M_k^{-1} r_k^5),
\]
\[
\Delta t_k = -M_k^{-1} (T_k \Delta \mu_k + r_k^5),
\]

Now, by eliminating \( \Delta u_k = -R_k^{-1} (B^T \Delta \lambda_{k+1} + r_k^5) \), system (39) becomes

\[
\begin{bmatrix}
\Delta x_k \\
\Delta u_k \\
\Delta \lambda_k \\
\Delta \lambda_{k+1} \\
\Delta \lambda_{k+1} \\
\end{bmatrix} = \begin{bmatrix}
A & -B R_k^{-1} B^T \\
Q_k & A^T \\
\end{bmatrix} \begin{bmatrix}
\Delta \mu_k \\
\Delta \mu_{k+1} \\
\end{bmatrix} + \begin{bmatrix}
r_k^1 \\
r_k^2 \\
r_k^3 \\
r_k^4 \\
\end{bmatrix},
\]

where residual \( r_k^4 = r_k^1 - \omega^T L_k (r_k^4 - M_k^{-1} r_k^5) \).

Remark 4.1. Coefficient matrix in system (42) is the \textit{discrete Hamiltonian matrix} in the interior-point algorithm for MPC.

So as to implement transformation (9) we have to write system (42) as a backward recursion, i.e.,

\[
\begin{bmatrix}
\Delta \mu_k \\
\Delta \lambda_k \\
\end{bmatrix} = H_m^{-1} \begin{bmatrix}
\Delta x_k \\
\Delta u_k \\
\Delta \lambda_k \\
\Delta \lambda_{k+1} \\
\Delta \lambda_{k+1} \\
\end{bmatrix} + \begin{bmatrix}
r_k^1 \\
r_k^2 \\
r_k^3 \\
r_k^4 \\
\end{bmatrix},
\]

where \( H_m^{-1} \) is

\[
H_m^{-1} = \begin{bmatrix}
A^{-1} & A^{-1} B R_k^{-1} B^T \\
Q_k A^{-1} & A^{-1} B R_k^{-1} B^T \\
\end{bmatrix}.
\]

The residuals are \( r_k^1 = -A^{-1} r_k^3 \) and \( r_k^3 = r_k^1 - Q_k A^{-1} r_k^3 \). It is interesting to observe the similarities between system (21), applied to the LQ tracking problem (subsection 3.1), and system (43). The only difference lies in the second term on the right hand side. For the LQ tracking problem this term corresponds to the reference signal \( x' \), whereas with IPM this corresponds to the residuals. Therefore, we can then apply the eigenvalue decomposition (9) to factorize system (37). In order to do so we first define the boundary conditions at the final time \( N \)

\[
\begin{bmatrix}
P - I & 0 & 0 & 0 & \omega \end{bmatrix} \begin{bmatrix}
\Delta \mu_N \\
\Delta \lambda_N \\
\Delta \mu_N \\
\Delta \lambda_N \\
\Delta \lambda_N \\
\end{bmatrix} = - \begin{bmatrix}
r_N^1 \\
r_N^2 \\
r_N^3 \\
r_N^4 \\
\end{bmatrix},
\]

Then we eliminate \( \Delta \mu_N \) and \( \Delta t_N \)

\[
\Delta \mu_N = L_N (\omega \Delta x_N + r_N^4 - M_N^{-1} r_N^5),
\]
\[
\Delta t_N = -M_N^{-1} (T_N \Delta \mu_N + r_N^5),
\]

so that we can define the terminal condition

\[
\Delta \lambda_N = P \Delta x_N + \hat{r}_N^4,
\]

where

\[
\hat{P} = P + \omega^T L_N \omega,
\]

\[
\hat{r}_N = r_N - \omega^T L_N (r_N - M_N^{-1} r_N^5).
\]

Now, we follow the same procedure as subsection 3.2 with the change of variable (10) applied to system (43)

\[
\begin{bmatrix}
\Delta \mu_k \\
\Delta \lambda_k \\
\end{bmatrix} = D_k \begin{bmatrix}
\Delta \mu_{k+1} \\
\Delta \lambda_{k+1} \\
\end{bmatrix} + \begin{bmatrix}
V_{w_k} \\
V_{y_k} \\
\end{bmatrix} r_k,
\]

where \( r_k = [r_k^1, r_k^2] \), \( V_{w_k} = [\tilde{V}_{11_k}, \tilde{V}_{12_k}] \) and \( V_{y_k} = [\tilde{V}_{21_k}, \tilde{V}_{22_k}] \). The well-conditioned solution of (47) is

\[
\begin{bmatrix}
\Delta w_N \\
\Delta y_k \\
\end{bmatrix} = \begin{bmatrix}
M_k^{-(N-i)} \\
M_k^{-(N-i)} \\
\end{bmatrix} \begin{bmatrix}
\Delta \mu_k \\
\Delta \lambda_k \\
\end{bmatrix} + \begin{bmatrix}
-S_{w_k,N-1}^w \\
-S_{w_k,N-1}^y \\
\end{bmatrix},
\]

where \( i = k \) and \( S_{w_k,N-1}^w, S_{w_k,N-1}^y \in \mathbb{R}^{N \times N} \) are

\[
S_{w_k,N-1}^w = \sum_{j=k}^{N-1} M_j^{-k-j} V_{w_j} V_{n_j-j+k-1},
\]

\[
S_{w_k,N-1}^y = \sum_{j=k}^{N-1} M_j^{-N-j-1} V_{y_j} V_{n_j-j+k-1}.
\]

We now can follow the same steps from equation (26) to (28) in subsection 3.2, so as to find \( T_k \) and \( v_k \)

\[
T_k = T_k M_k^{-(N-k)},
\]

\[
v_k = -T_k S_{w_k,N-1}^w + M_k^{-(N-k)} T_{w_k}^{-1} S_{w_k,N-1}^w.
\]

where \( T_k = -M_k^{-(N-k)} T_{w_k}^{-1} T_k \). From (31) to (32), \( \Delta \lambda x \) can be obtained

\[
\Delta \lambda x = \bar{T}_k (\Delta x_k - v_k^x) + v_k^x.
\]

Remark 4.2. Instead of computing directly the control input \( v_k \), like in algorithms 1 and 3, our aim here is to compute the search direction \( \nabla p \). Then, since \( \Delta x_0 = 0 \), we just need to compute \( \Delta \lambda x \) from (52) and carry out a forward substitution to obtain \( \Delta x, \Delta u, \Delta \lambda, \Delta \mu \) and \( \Delta t \). The algorithm 4 shows how to compute \( \nabla p \).

4.3 Implications on online implementation

The fundamental difference between the method proposed here and the one used by Rao et al. [1998] is how the backward recursion is computed. The former uses the eigenvalue decomposition and the latter uses the \textit{discrete time-varying Riccati equation} (7). Loosely speaking, both algorithms are linearly dependent on the horizon \( N \) and cubic on the states \( n_u \), i.e. their complexity is \( \mathcal{O}(N^2n_u^2) \).

5. CONCLUSION

In this paper, we first demonstrated that the eigenvalue decomposition used for the solution of the finite-horizon LQR can be extended to the LQ tracking problem. We also applied the receding horizon idea to these two optimal control problems in order to implement them on the unconstrained
Algorithm 4 For $\nabla p$ in Interior-Point Method

1: function direct($\bar{Q}, \bar{P}, \bar{R}, r_1^N, r_2^N, r_3^N, r_4^N, r_5^N$)
2: *Set: $T_{k+1} = \bar{P}$ and $\pi_{k+1} = \bar{r}_N^N$
3: *Backward Substitution:
4: for $k = N-1$ to 0 do
5: $H_k = \bar{V}_k D_k V_k^T$ 
6: $T_{w_k} = (V_{22_k} - T_{k+1} V_{12_k})$
7: $T_{u_k} = (V_{21_k} - T_{k+1} V_{11_k})$
8: $T_k = -M_k^{-1} (T_{w_k} T_{u_k}) M_k^{-1}$
9: $T_k = T_k + T_{k} (T_{k}^{-1})^T$; 
10: $S_k = V_{11_k} r_2^k + V_{21_k} r_3^k + V_{22_k} r_4^k$
11: $v_k = -T_k S_k^T + M_k^{-1} (T_{w_k} \pi_{k+1}) + S_k^T$
12: $v_k = V_{12_k} v_k, v_k^T = V_{22_k} v_k$; 
13: $\pi_k = -T_k v_k + v_k$; 
14: end for
15: $\Delta \lambda_0 = T_0 \Delta x_0 + \pi_0$
16: *Forward Substitution: 
17: for $k = 0$ to $N-1$ do
18: $\Delta \lambda_{k+1} = (A_k T_k)^{-1} (Q_k \Delta x_k + \Delta \lambda_k - \bar{r}_k^k)$
19: $\Delta u_k = -\bar{R}_k^{-1} (B_k^T \Delta \lambda_{k+1} + \bar{r}_k^T)$
20: $\Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k + \bar{r}_k^{k+1}$
21: end for
22: $\Delta \mu = \mathcal{L} (\Delta \theta + r^4 - M_k^{-1} r_5^k)$
23: $\Delta t = -M_k^{-1} (T \Delta \mu + r_5^k)$
24: return $\nabla p = [\Delta \theta, \Delta \lambda, \Delta \mu, \Delta t]^T$
24: end function

MPC. These techniques serve as a preamble for a method (section 4) that introduces a novel approach to solve the optimization problem for MPC using an eigenvalue decomposition on the Hamiltonian matrix (42) within the IPM algorithm. Although in terms of complexity the algorithm has a similar cost to the Riccati recursion, it paves the way to address the optimization problem in a novel manner within MPC. Current research indicates there is considerable potential to reduce computational cost. Therefore, in future work attention will be paid to the computational efficiency of the algorithm in order to evaluate its value and feasibility.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from two entities, Festo AG & Co.KG and the Mexican Council of Science and Technology (CONACYT). In addition, we would like to thank Prof. David J. Bell for the helpful discussions during the course of this work.

REFERENCES

Boyd, S. and Vandenberghe, L. (2004). Convex Optimization. Cambridge University Press.
Cantoni, M., Farokhi, F., Kerrigan, E., and Shames, I. (2016). A scalable QP solver for optimal control of cascades with constraints. Australian Control Conference, 132–134.
Ferreau, H.J., Bock, H.G., and Diehl, M. (2008). An online active set strategy to overcome the limitations of explicit MPC. International Journal of Robust and Nonlinear Control, 18, 816–830.
Fletcher, R. (1987). Practical Methods of Optimization. John Wiley and Sons, Second edition.
Frison, G. and Jorgensen, J.B. (2013). A fast condensing method for solution of linear-quadratic control problems. IEEE Conference on Decision and Control, 7715–7720.
Gaalman, G. (1980). Comments on a nonrecursive algebraic solution for the discrete Riccati equation. IEEE Transactions on Automatic Control, 25, 610–612.
Jerez, J.L., Goulart, P.J., Richter, S., Constantines, G.A., Kerrigan, E.C., and Morari, M. (2013). Embedded predictive control on an FPGA using the fast gradient method. European Control Conference (ECC), 3614–3620.
Kalman, R. (1960). Contributions to the theory of optimal control. Boletin de la Sociedad Mexicana de Matemáticas, 5, 102–119.
Kufolal, D.K.M., Richter, S., Imland, L., Johansen, T.A., Morari, M., and Eikrem, G.O. (2014). Embedded model predictive control on a PLC using a primal-dual first-order method for a subsea separation process. Mediterranean Conference on Control and Automation, 368–373.
Lewis, F.L., Vrabie, D.L., and Syrmos, V.L. (2012). Optimal Control. John Wiley and Sons INC, Third edition.
MacFarlane, A.G.J. (1963). An eigenvector solution of the optimal linear regulator problem. Electronics and Control, 14:6, 643–654.
Maciejowski, J. (2002). Predictive Control with Constraints. Prentice Hall.
Mehrotra, S. (1992). On the implementation of a primal-dual interior-point method. Society for Industrial and Applied Mathematics, 2:4, 575–601.
Nesterov, Y. (2003). Introductory Lectures on Convex Optimization: A Basic Course. Springer Science and Business Media.
Pappas, T., Laub, A.J., and Sandell, N.R. (1980). On the numerical solution of the discrete-time algebraic Riccati equation. IEEE Transactions on Automatic Control, 25, 631–641.
Rao, C., Wright, S., and Rawlings, J. (1998). Application of interior-point method to model predictive control. Journal of Optimization Theory and Applications, 99, 723–757.
Vaughan, D.R. (1968). A negative exponential solution for the linear optimal regulator problem. Joint Automatic Control Conference preprints, 717–725.
Vaughan, D.R. (1970). A nonrecursive algebraic solution for the discrete Riccati equation. IEEE Transactions on Automatic Control, 597–599.
Wang, Y. and Boyd, S. (2010). Fast model predictive control using online optimization. IEEE Transactions on Control Systems Technology, 18, 267–278.
Wills, A.G., Bates, D., Fleming, A.J., NIMNESS, B., and MOHEIMANI, S.O.R. (2008). Model predictive control applied to constraint handling in active noise and vibration control. IEEE Transactions on Control Systems Technology, 16, 3–12.
Wills, A.G. and Heath, W.P. (2004). Barrier function based model predictive control. Automatica, 40, 1415–1422.
Wright, S.J. (1997). Primal-Dual Interior Point Methods. Society for Industrial and Applied Mathematics.