TECHNISCHE UNIVERSITÄT CHEMNITZ

Sonderforschungsbereich 393

Parallele Numerische Simulation für Physik und Kontinuumsmechanik

Peter Benner   Enrique S. Quintana-Ortí   Gregorio Quintana-Ortí

Solving Linear-Quadratic Optimal Control Problems on Parallel Computers

Preprint SFB393/05-19

Abstract

We discuss a parallel library of efficient algorithms for the solution of linear-quadratic optimal control problems involving large-scale systems with state-space dimension up to $\mathcal{O}(10^4)$. We survey the numerical algorithms underlying the implementation of the chosen optimal control methods. The approaches considered here are based on invariant and deflating subspace techniques, and avoid the explicit solution of the associated algebraic Riccati equations in case of possible ill-conditioning. Still, our algorithms can also optionally compute the Riccati solution. The major computational task of finding spectral projectors onto the required invariant or deflating subspaces is implemented using iterative schemes for the sign and disk functions. Experimental results report the numerical accuracy and the parallel performance of our approach on a cluster of Intel Itanium-2 processors.

Keywords. Linear-quadratic optimal control, algebraic Riccati equation, disk function, sign function, parallel algorithms.

Preprintreihe des Chemnitzer SFB 393

ISSN 1619-7178 (Print)   ISSN 1619-7186 (Internet)

SFB393/05-19   **December 2005**

Contents

Authors' addresses:

Peter Benner
Fakultät für Mathematik
Technische Universität Chemnitz
09107 Chemnitz, Germany
e-mail: `benner@mathematik.tu-chemnitz.de`

Enrique S. and Gregorio Quintana-Ortí
Depto. de Ingeniería y Ciencia de Computadores
Universidad Jaume I
12.071–Castellón, Spain
e-mail: {`quintana,gquintan`}`@icc.uji.es`

P. Benner was supported by the DFG Sonderforschungsbereich 393 *Parallele Numerische Simulation für Physik und Kontinuumsmechanik* at TU Chemnitz.

E.S. Quintana-Ortí and G. Quintana-Ortí were supported by the CICYT project No. TIN2005-09037-C02-02 and FEDER; and project No. P1B-2004-6 of the *Fundación Caixa-Castellón/Bancaixa and UJI.*

1 Introduction

In recent years, many new and reliable numerical methods have been developed for the analysis and synthesis of moderate-size linear time-invariant (LTI) systems. In generalized state-space form, such systems are described by the following models:
Continuous-time LTI system:

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t), \qquad t > 0, \qquad x(0) = x^{(0)}, \\ y(t) &= Cx(t), \qquad\qquad\quad\; t \geq 0, \end{aligned} \tag{1}$$

Discrete-time LTI system:

$$\begin{aligned} Ex_{k+1} &= Ax_k + Bu_k, \qquad k = 0, 1, \ldots, \qquad x_0 = x^{(0)}, \\ y_k &= Cx_k, \qquad\qquad\; k = 0, 1, \ldots, \end{aligned} \tag{2}$$

where $A, E \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and $C \in \mathbb{R}^{p\times n}$. Also, n is known as the order or the state-space dimension of the system.

In particular, in this paper we consider the numerical solution of the linear-quadratic optimal control (LQOC) problems:
Continuous-time LTI system:

$$\min_{u\in L_2(0,\infty)} \frac{1}{2}\int_0^\infty \left(y(t)^T Q y(t) + 2y(t)^T S u(t) + u(t)^T R u(t)\right)\, dt, \tag{3}$$

Discrete-time LTI system:

$$\min_{u\in \ell_2(0,\infty)} \frac{1}{2}\sum_{k=0}^{\infty} \left(y_k^T Q y_k + 2y_k^T S u_k + u_k^T R u_k\right), \tag{4}$$

where $Q \in \mathbb{R}^{p\times p}$, $R \in \mathbb{R}^{m\times m}$, $S \in \mathbb{R}^{p\times m}$, $R = R^T$, $Q = Q^T$, and the systems are subject to the linear dynamic constraints in (1) and (2), respectively. Hereafter, we will assume that E is nonsingular and R is positive semidefinite. If E is singular, regularization procedures such as those described in [9, 34] allow to reduce the problem to an LTI system with nonsingular E.

The LQOC problem can be cast as an infinite-dimensional optimization problem. After a complete discretization of the problem and truncating the infinite sums, we obtain in both cases a standard quadratic problem that can be solved with any suitable algorithm for quadratic programming; see, e.g., [36]. Nevertheless, this approach destroys part of the structure of the underlying system and does not take into account the dynamics of the system nor the feedback structure of the optimal control u. Therefore, it is rarely employed in control theory unless the approaches which tackle the LQOC problems directly by aiming at their analytical solution without discretization and truncation can no longer be used due to, e.g., state or control constraints. Moreover, even in the presence of control constraints, the computational routines described in this paper can be used as an integral part of recent algorithms for computing an optimal control; see [24, 23].

The solution of (3)–(4) determines the so-called linear-quadratic regulator (LQR) and is a major computational task in stabilization, Kalman filtering, and H_2-optimization/linear-quadratic Gaussian (LQG) design for linear systems in state-space form (1)–(2); see [3, 20, 28, 26, 31, 41] to name only a few references. Large-scale LQOC problems often arise when modeling of physical process by means of partial differential equations (PDEs) is followed by a spatial discretization; the control of chemical reactions or power networks often yields a large number of ordinary differential equations (ODEs) as well. Even though these ODEs are often nonlinear, LQOC is often used for deriving feedback control laws by applying a proper linearization around a working point; see, e.g, [32, 35].

In general, numerical methods for solving LQOC problems that involve LTI systems with dense state matrices present a computational cost of $\mathcal{O}(n^3)$ floating-point arithmetic operations (flops) and require storage for $\mathcal{O}(n^2)$ numbers. While current desktop computers provide enough computational power to solve problems with state-space dimension n up to 1000, using libraries as SLICOT[1] or the MATLAB control-related toolboxes, large-scale applications clearly require the use of advanced computing techniques. Here we introduce a library of parallel algorithms for the solution of LQOC problems on parallel distributed-memory architectures. In our algorithms we not only parallelize the underlying computational steps but often replace them by methods that are better suited for parallel computation. The library, PLiCOC, can be installed and run in any (serial and) parallel architecture where the computational libraries LAPACK, BLAS, and ScaLAPACK [4, 15] are available.

The rest of the paper is structured as follows. In Section 2 we establish the relation between the LQOC problem, algebraic Riccati equations, and a class of matrix pencils. Sections 3 and 4 are devoted to the introduction of the sign and disk functions and the description of their application to the solution of LQOC problems. The integration of the LQOC algorithm in a parallel library for control theory is outlined in Section 5. Finally, the performance on a cluster of Intel Itanium-2 processors is reported in Section 6, and some concluding remarks follow in Section 7.

2 LQOC problems, algebraic Riccati equations, and matrix pencils

The LQOC problems for continuous- and discrete-time LTI systems are closely related to the (generalized) continuous-time algebraic Riccati equation (CARE)

$$\begin{aligned} 0 = \mathcal{R}_c(X) := {} & C^TQC + A^TXE + E^TXA \\ & -(E^TXB + C^TS)R^{-1}(B^TXE + S^TC), \end{aligned} \tag{5}$$

[1] Available from `http://www.slicot.net`.

and the (generalized) discrete-time algebraic Riccati equation (DARE)

$$
\begin{aligned}
0 = \mathcal{R}_d(X) := & \, C^TQC + A^TXA - E^TXE \\
& -(A^TXB + C^TS)(R + B^TXB)^{-1}(B^TXA + S^TC).
\end{aligned} \tag{6}
$$

Under mild assumptions, the minimum of the optimization problems (3)–(4) is attained by the feedback control laws

$$
u(t) = -R^{-1}(B^TX_cE + S^TC)x(t) =: -F_cx(t), \quad t \geq 0, \tag{7}
$$

in the continuous-time case, and

$$
u_k = -(R + B^TX_dB)^{-1}(B^TX_cA + S^TC)x_k =: -F_dx(t), \quad k = 0, 1, \ldots, \tag{8}
$$

in the discrete-time case, where X_c and X_d denote the stabilizing solutions of the CARE and the DARE, respectively. Here, the *stabilizing solution* is defined by the property that the closed-loop matrix pencil $(A - BF_c) - \lambda E$ has all its eigenvalues in the open left half plane, denoted as $\lambda(E, A - BF_c) \subset \mathbb{C}^-$, and the closed-loop matrix pencil $(A - BF_d) - \lambda E$ has all its eigenvalues inside the unit circle. We also say then that $(A - BF_c) - \lambda E$ is *c-stable* and $(A - BF_d) - \lambda E$ is *d-stable*. The matrices F_c and F_d are called the (optimal) *feedback gain* matrices.

Remark 2.1 *It should be emphasized here that (7) and (8) provide analytical expressions of the optimal controls. Under usual assumptions on the control problem, these solutions are the unique optimizers of the LQOC problem; see [34]. Hence there is no need for discretization and quadratic programming algorithms for computing the optimum. Only the matrices defining the gains F_c and F_d need to be computed. We will discuss their numerical computation in the following.*

Assume now that R is nonsingular and define $\tilde{A} = (A - BR^{-1}S^TC)$, $\tilde{Q} = C^T(Q - SR^{-1}S^T)C$, and $\tilde{G} = BR^{-1}B^T$. The CARE is associated with the matrix pencil

$$
H - \lambda K = \begin{bmatrix} \tilde{A} & \tilde{G} \\ \tilde{Q} & -\tilde{A}^T \end{bmatrix} - \lambda \begin{bmatrix} E & 0 \\ 0 & E^T \end{bmatrix}, \tag{9}
$$

while an analogous relation exists between the DARE and the matrix pencil

$$
M - \lambda L = \begin{bmatrix} \tilde{A} & 0 \\ \tilde{Q} & E^T \end{bmatrix} - \lambda \begin{bmatrix} E & -\tilde{G} \\ 0 & \tilde{A}^T \end{bmatrix}. \tag{10}
$$

Specifically, denoting the invariant or deflating subspace of a matrix or matrix pencil, respectively, as *c-stable* (*d-stable*) if it corresponds to the eigenvalues in the open left half plane (inside the unit circle) of the corresponding matrix or matrix pencil, we have the following relations (see [5, 34, 45]). If the columns of $\begin{bmatrix} U_c^T & V_c^T \end{bmatrix}^T$, with U_c, $V_c \in \mathbb{R}^{n \times n}$, form a basis for the c-stable deflating subspace of $H - \lambda K$, then the unique positive semidefinite solution of the CARE is given by $X_c = -V_c U_c^{-1} E^{-1}$. Analogously, if the columns of $[U_d\ V_d]^T$, with U_d, $V_d \in \mathbb{R}^{n\times n}$, form a basis for the d-stable deflating subspace of $M - \lambda L$, then the unique positive semidefinite solution of the DARE is given by $X_d = -V_d U_d^{-1} E^{-1}$. The optimal stabilizing feedbacks can be then obtained from (7) and (8). Note that for (7), X_c is not needed explicitly, but can be computed directly as

$$u(t) = u(t) = R^{-1}(B^T V_c U_c^{-1} - S^T C)x(t) = -F_c x(t). \tag{11}$$

Now, in case R is singular, neither of these two matrix pencils can be formed. Nevertheless, the approach in [5, 34] still allows to obtain the optimal stabilizing feedbacks from the following extended matrix pencils:

$$\mathcal{H} - \lambda\mathcal{K} = \begin{bmatrix} A & 0 & B \\ C^T Q C & A^T & C^T S \\ S^T C & B^T & R \end{bmatrix} - \lambda \begin{bmatrix} E & 0 & 0 \\ 0 & -E^T & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{12}$$

associated with the continuous-time case, and

$$\mathcal{M} - \lambda\mathcal{L} = \begin{bmatrix} A & 0 & B \\ C^T Q C & -E^T & C^T S \\ S^T C & 0 & R \end{bmatrix} - \lambda \begin{bmatrix} E & 0 & 0 \\ 0 & -A^T & 0 \\ 0 & -B^T & 0 \end{bmatrix}, \tag{13}$$

for the discrete-time case. Assume that the columns of $\left[U_c^T\ V_c^T\ W_c^T\right]^T$ form a basis for the c-stable deflating subspace of $\mathcal{H} - \lambda\mathcal{K}$ while the columns of $\left[U_d^T\ V_d^T\ W_d^T\right]^T$ form a basis for the d-stable deflating subspace of $\mathcal{M} - \lambda\mathcal{L}$, where U_c, V_c, U_d, $V_d \in \mathbb{R}^{n\times n}$, and W_c, $W_d \in \mathbb{R}^{m\times n}$. Then:

– If U_c is nonsingular, the optimal feedback solution of (3) is given by

$$u(t) = W_c U_c^{-1} x(t) =: F_c x(t), \quad t \geq 0. \tag{14}$$

– If U_d is nonsingular, the optimal feedback solution of (4) is given by

$$u_k = W_d U_d^{-1} x_k =: F_d x_k, \quad k = 0, 1, \ldots. \tag{15}$$

It should be noted that this approach is not only of interest when R is singular. In case R is close to a singular matrix, and hence ill-conditioned, forming the matrix pencils in (9) and (10) is numerically hazardous; roundoff error may then lead to large errors in the computed optimal controls. We therefore advise to use (12) and (13) in case an ill-conditioning of R is detected.

In the following sections we will review methods for computing bases of the required c- and d-stable deflating subspaces. The approaches discussed below are based on computing projectors on these subspaces. As they are associated with certain parts of the spectrum of the matrix pencils under consideration, we call them *spectral projectors* and both the sign and disk function methods below can therefore be considered as *spectral projection methods*.

3 Solution of LQOC problems via the matrix sign function

3.1 The matrix sign function

The sign function method was first introduced by Roberts [40] to solve the standard CARE ($E = I_n$). The *sign function* of a matrix $Z \in \mathbb{R}^{n\times n}$ with no eigenvalues on the imaginary axis can be defined as follows: Let $Z = S \left[\begin{smallmatrix} J^- & 0 \\ 0 & J^+ \end{smallmatrix}\right] S^{-1}$ denote the Jordan decomposition of Z where the Jordan blocks corresponding to the, say, k eigenvalues in the open left half plane are collected in J^- and the Jordan blocks corresponding to the remaining $n-k$ eigenvalues in the open right half plane are collected in J^+. Then

$$\operatorname{sign}(Z) \;:=\; S \begin{bmatrix} -I_k & 0 \\ 0 & I_{n-k} \end{bmatrix} S^{-1}.$$

The sign function provides projectors onto certain subspaces of the matrix. Specifically, $\mathcal{P}^- := \frac{1}{2}(I_n - \operatorname{sign}(Z))$ defines the skew projection onto the c-stable Z-invariant subspace parallel to the c-unstable Z-invariant subspace whereas $\mathcal{P}^+ := \frac{1}{2}(I_n + \operatorname{sign}(Z))$ defines the skew projection onto the c-unstable Z-invariant subspace parallel to the c-stable Z-invariant subspace. Therefore, the sign function provides a tool for a spectral decomposition along the imaginary axis, and therefore is a useful technique for solving continuous-time LQOC problems.

3.2 Newton-type iterations for the sign function

The sign function can be computed via the Newton iteration for the equation $Z^2 = I_n$ where the starting point is chosen as Z, i.e.,

$$Z_0 \leftarrow Z, \quad Z_{j+1} \leftarrow \frac{1}{2}(Z_j + Z_j^{-1}), \quad j = 0, 1, \ldots. \tag{16}$$

Under the given assumptions, the sequence $\{Z_j\}_{j=0}^{\infty}$ converges to $\operatorname{sign}(Z) = \lim_{j\to\infty} Z_j$ [40] with an ultimately quadratic convergence rate. As the initial convergence may be slow, the use of acceleration techniques is recommended; e.g., *determinantal scaling* [17] adds the following step to (16):

$$Z_j \leftarrow c_j Z_j, \quad c_j = |\det(Z_j)|^{-\frac{1}{n}}, \tag{17}$$

where $\det(Z_j)$ denotes the determinant of Z_j. For a summary of different strategies for accelerating the convergence of the Newton iteration, see [30].

The matrix sign function as computed by (16) can only be applied directly to (9) if $E = I_n$. Otherwise, one could transform the system (1) to one with $E = I_n$ by a change of coordinates $x \to Ex$, but this requires to solve linear systems of equations with E. This can not be recommended even if E is mildly ill-conditioned as errors imposed by the poor conditioning unnecessarily effect the solution of the LQOC problem. Instead, in case

$E \neq I_n$ we consider a generalization of the matrix sign function method to a matrix pencil $Z - \lambda Y$ given in [22]. Assuming that Z and Y are nonsingular, the *generalized Newton iteration* for the matrix sign function is given by

$$Z_0 \leftarrow Z, \quad Z_{j+1} \leftarrow \frac{1}{2c_j}(Z_j + c_j^2 Y Z_j^{-1} Y), \qquad j = 0, 1, \ldots, \tag{18}$$

with the scaling now defined as $c_j \leftarrow \left(\frac{|\det(Z_j)|}{|\det(Y)|}\right)^{\frac{1}{n}}$. This iteration is equivalent to computing the sign function of the matrix $Y^{-1}Z$ via the Newton iteration as given in (16). If $\lim_{j\to\infty} Z_j = Z_\infty$, then $Z_\infty - Y$ defines the skew projection onto the c-stable right deflating subspace of $Z - \lambda Y$ parallel to the c-unstable deflating subspace, and $Z_\infty + Y$ defines the skew projection onto the c-unstable right deflating subspace of $Z - \lambda Y$ parallel to the c-stable deflating subspace.

As a basis for the c-stable invariant subspace of a matrix Z or the c-stable deflating subspace of a matrix pencil $Z - \lambda Y$ is given by the range of any projector onto this subspace, it can be computed by a rank-revealing QR (RRQR) factorization [19, 25] of the corresponding projectors $\mathcal{P}^-$ or $Z_\infty - Y$, respectively. As will be explicitly stated in the next section, this is specially important as it shows that the solution of the continuous-time LQOC problems can be obtained using the sign function to compute the appropriate basis.

Much of the appeal of the matrix sign function approach comes from the high parallelism of the matrix kernels that compose the Newton-type iterations [38]. Efficient parallelization of these type of iterations for the matrix sign function has been reported, e.g., in [6, 29]. An approach that basically reduces the cost of the generalized Newton iteration to that of the Newton iteration is described in [42].

3.3 Application to LQOC problems

Let us first consider the case that R is nonsingular. The LQOC problem (1)-(3) and the associated CARE can be solved by applying the sign function method to the matrix pencil $H - \lambda K$. Specifically, let

$$Z_\infty = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}$$

denote the limit of the generalized Newton iteration (18) applied to the matrix pencil $Z - \lambda Y = H - \lambda K$. (Note that, in case $E = I_n$, it suffices to apply the Newton iteration (16) to the matrix $Z = H$ to obtain Z_∞.) An orthonormal basis for the desired deflating subspace of the matrix pencil $H - \lambda K$ can then be obtained from $Y_\infty - Z$ and the stabilizing feedback from (7). However, in practice it is more reliable to compute the solution of the CARE by observing that the basis vectors of the desired subspaces are contained in the kernel of $Z_\infty + Y$. If the special basis of the subspace, given by the columns of $\left[\begin{smallmatrix} I_n \\ X_c E \end{smallmatrix}\right]$, is sought, then this leads to the overdetermined, but consistent, homogeneous linear leastsquares (LLS) problem

$$\left[\begin{array}{c} Z_{12} \\ Z_{22}+E^T \end{array}\right] X_c E = -\left[\begin{array}{c} Z_{11}+E \\ Z_{21} \end{array}\right];$$

see [22]. Note again that for (11) it suffices to compute $X_E := X_c E$ so that E need not be factored.

Analogously, in case we are dealing with a discrete-time system, the LQOC problem (4) with constraint (2) and the DARE can be solved by applying the generalized Newton iteration to the matrix pencil $Z - \lambda Y = (L+M) - \lambda(L-M)$, and then solving the LLS problem

$$\left[\begin{array}{c} Z_{12}+G \\ Z_{22}+E^T-A^T \end{array}\right] X_d E = -\left[\begin{array}{c} Z_{11}+A-E \\ Z_{21}+Q \end{array}\right];$$

see [22]. The feedback control is obtained in this case from (8).

When R is ill-conditioned or even singular, so that the solution of both LQOC problems requires the extended pencil formulations $\mathcal{H} - \lambda\mathcal{K}$ and $\mathcal{M} - \lambda\mathcal{L}$, the sign function method is not directly applicable. We will come back to this problem in Subsection 4.3 where we describe a compression technique that reduces the extended pencils to matrix pencils which again can be treated using the generalized sign function method.

4 Solution of LQOC problems via the matrix disk function

4.1 The matrix disk function

The *right matrix pencil disk function* can be defined for a regular matrix pencil $Z - \lambda Y$, $Z,\, Y \in \mathbb{R}^{n\times n}$, as follows: Let

$$Z - \lambda Y = S \left[\begin{array}{cc} J^0 - \lambda I_k & 0 \\ 0 & J^\infty - \lambda N \end{array}\right] T^{-1}$$

denote the Kronecker (Weierstrass) canonical form of the matrix pencil [21], where $J^0 \in \mathbb{C}^{k\times k}$, $J^\infty \in \mathbb{C}^{(n-k)\times(n-k)}$ contain, respectively, the Jordan blocks corresponding to the eigenvalues of $Z - \lambda Y$ inside and outside the unit circle. Then, the matrix (pencil) disk function is defined as

$$\operatorname{disk}(Z, Y) := T\left(\left[\begin{array}{cc} I_k & 0 \\ 0 & 0 \end{array}\right] - \lambda \left[\begin{array}{cc} 0 & 0 \\ 0 & I_{n-k} \end{array}\right]\right) T^{-1} =: \mathcal{P}^0 - \lambda \mathcal{P}^\infty.$$

Alternative definitions of the disk function are given in [10].

Again, the matrix disk function can be used to compute projectors onto deflating subspaces of a matrix pencil as $\mathcal{P}^0$ is a skew projection onto the right d-stable deflating subspace of $Z - \lambda Y$ parallel to the d-unstable one, and $\mathcal{P}^\infty$ defines a skew projection onto the right d-unstable deflating subspace of $Z - \lambda Y$ parallel to the d-stable one. Thus the

Algorithm 1 Inverse Free Method.

INPUT: A matrix pencil $Z - \lambda Y$, $Z, Y \in \mathbb{R}^{n \times n}$ with no eigenvalues on the unit circle.

OUTPUT: $U \in \mathbb{R}^{n \times k}$, where the columns of U span an orthogonal basis for the d-stable deflating subspace of $Z - \lambda Y$.

1: Set $Z_0 = Z$, $Y_0 = Y$. {*Inverse free iteration*}

2: ***for*** $j = 0, 1, \ldots$ *until convergence* ***do***

3: $\begin{bmatrix} Y_j \\ -Z_j \end{bmatrix} = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix} \begin{bmatrix} R_j \\ 0 \end{bmatrix}$ *(QR factorization),*

4: $Z_{j+1} = U_{12}^T Z_j,$

5: $Y_{j+1} = U_{22}^T Y_j,$

6: $s = j + 1.$

7: ***end for***

8: *Compute an RRQR factorization of* $(Z_s + Y_s)^{-1} Y_s = URP$, $k = \mathrm{rank}\,(R)$.

9: *Set* $U = U(:, 1:k)$.

disk function provides a tool for spectral decomposition along the unit circle, which is exactly the type of decomposition required when solving discrete-time LQOC problems.

In the next subsection we expose how the matrix disk function can be computed iteratively without having to invert any of the iterates.

4.2 The inverse free iteration

Malyshev [33] proposed an iterative scheme based on earlier methods [16] that can be used to compute the disk function. The algorithm was improved in [7] to become inverse-free, and in [43] to reduce its cost. This method is designed to compute the deflating subspace of a matrix pencil corresponding to the eigenvalues inside or outside the unit circle. Other splittings may be obtained by applying an appropriate *Möbius transformation* to the matrix pencil.

Given a regular matrix pencil $Z - \lambda Y$ having no eigenvalues on the unit circle, Algorithm 1 reports an implementation of the inverse-free method which computes an approximation to the right deflating subspace corresponding to the eigenvalues inside the unit circle. It is based on a generalized power iteration (see [8, 43] for more details) and the fact that (see [8])

$$\lim_{j \to \infty} (Z_j + Y_j)^{-1} Y_j = \mathcal{P}^0, \quad \lim_{j \to \infty} (Z_j + Y_j)^{-1} Z_j = \mathcal{P}^\infty.$$

The RRQR factorization in Step 3 can be computed without inverting $Z_s + Y_s$ explicitly as described in [7]: first, compute an RRQR factorization

$$Y_s = \hat{U} \hat{R} \hat{P},$$

where $\hat{U}$ is orthogonal, $\hat{R}$ is upper triangular, and $\hat{P}$ is a permutation matrix. Here,

$k = \text{rank}\,(Y_s)$ and can be thus estimated from the RRQR factorization. Next, factor

$$\hat{U}^T(Z_s + Y_s) \;=\; \tilde{R}\tilde{U},$$

where $\tilde{R}$ is upper triangular and $\tilde{U}$ is orthogonal. Then in Step 4, $U = \tilde{U}^T$. This shows that the algorithm is really *inverse free*.

The convergence of the inverse free iteration can be shown to be globally quadratic [7].

The price paid for avoiding matrix inversions during the iteration is that every iteration step is about twice as expensive as an step of the Newton iteration in the matrix pencil case and three times as expensive as the Newton iteration in the matrix case. On the other hand, the inverse free iteration can be implemented as efficiently on a parallel distributed-memory architecture as the sign function method since it is based on matrix multiplications and QR factorizations which are well studied problems in parallel computation; see, e.g., [25, 39] and the references given therein.

4.3 Application to LQOC problems

The discrete-time LQOC problem (4) with constrain (2) and the DARE can be solved by a straightforward application of Algorithm 1 to the corresponding matrix pencil $M - \lambda L$ or the extended matrix pencil $\mathcal{M} - \lambda\mathcal{L}$, as long as these pencils are regular and have an n–dimensional d-stable deflating subspace such that for a basis of this subspace spanned by the columns of $U \in \mathbb{R}^{2n\times n}$ or $U \in \mathbb{R}^{(2n+m)\times n}$, the upper $n \times n$–part of U is invertible. Note that infinite eigenvalues of any of the matrix pencils are d-unstable and thus do not cause problems here *per se*. Therefore, in principle, we can apply the disk function method to the matrix pencils even if R is singular. However, in this situation a singular matrix pencil may result, and a stable deflating subspace cannot be extracted. In this case, the extended pencil should be first processed by the compression technique described in Subsection 4.4 in order to reduce the problem to that of a regular pencil. The same is true if R is numerically singular as the ill-conditioning of R will also effect the computations even if it is not inverted.

In the continuous-time case, the LQOC problem and the CARE can be handled by applying Algorithm 1 to $Z - \lambda Y = (H + K) - \lambda(H - K)$. However, this is no longer true for the extended matrix pencil in (12) as $\mathcal{H} - \lambda\mathcal{K}$ has m infinite eigenvalues, even if R is invertible. The Cayley transformation employed when forming $(\mathcal{H}+\mathcal{K})-\lambda(\mathcal{H}-\mathcal{K})$ will map the infinite eigenvalues to the unit circle so that the assumptions needed for convergence of Algorithm 1 are violated. This problem can be overcome by deflating out the infinite eigenvalues using the technique described in Subsection 4.4. At the same time, a possibly singular matrix pencil is avoided.

Although we are not aware of a strategy to accelerate the inverse free iteration in general, a strategy for the continuous-time case is developed in [8]. It is based on the scaling proposed for the generalized Newton iteration and uses the equivalence of the iterates in Algorithm 1 and (16) proven in [8]. Here, we initially set

$$\tilde{H} - \lambda\tilde{K} := \frac{1}{|\det H|^{\frac{1}{2n}}} H - \lambda \frac{1}{|\det K|^{\frac{1}{2n}}} K = \frac{1}{|\det H|^{\frac{1}{2n}}} H - \lambda \frac{1}{|\det E|^{\frac{1}{n}}} K. \tag{19}$$

Note that this is exactly the determinantal scaling (17) if we would apply the sign function method to $H - \lambda K$. The scaling (19) is only applied in the first step as it would require additional factorizations if employed in each iteration step. Examples in [8] demonstrate that this scaling strategy is often quite successful. This may be explained by the fact that scaling aims at improving the convergence in early stages of the Newton iteration while it has no effect when the region of quadratic convergence is reached.

When applying the sign function method to the solution of algebraic Riccati equations, it was observed in Subsection 3.3 that the solution could be obtained by a LLS problem thus avoiding the explicit computation of the deflating subspace. We describe next how this can similarly be achieved for the disk function method. Let

$$Z_\infty = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{12} & Z_{22} \end{bmatrix}$$

denote the limit of the inverse free iteration (Step 2 in Algorithm 1) with $Z - \lambda Y = (H + K) - \lambda(H - K)$ in the continuous-time case and $Z - \lambda Y = M - \lambda L$ for the discrete-time case. Then, $\ker(Z_\infty) = \ker(\mathcal{P}^\infty)$, where $\ker(W)$ denotes the *kernel* (or *nullspace*) of W. As the stable deflating subspace is contained in $\ker(\mathcal{P}^\infty)$ it can be obtained by solving a homogeneous linear system of equations. Employing again the special basis exhibiting the solution of the CARE or DARE in the lower part, this yields the LLS

$$\begin{bmatrix} Z_{12} \\ Z_{22} \end{bmatrix} X_E = \begin{bmatrix} Z_{11} \\ Z_{21} \end{bmatrix}. \tag{20}$$

The continuous-time LQOC problem can thus be solved without explicitly computing the solution of the CARE using $F_c = R^{-1}(B^T X_E + S^T C)$. If the stabilizing solution of the CARE is sought rather than the optimal stabilizing feedback, one still has to solve the linear system $X_c E = X_E$.

In the discrete-time case the situation is somewhat different. The optimal feedback control as given by (8) is

$$u_k = -(R + B^T X_d B)^{-1}(B^T X_d A + S^T C)x_k$$

such that we can not use X_E to compute that directly. Therefore, we first have to solve the linear system

$$X_d E = X_E \tag{21}$$

for X_d and then proceed with $F_d = (R+B^T X_d B)^{-1}(B^T X_d A+S^T C)$. If E is ill-conditioned with respect to inversion, computing X_d from (21) might not be such a good idea. In that case, and also when R is singular or ill-conditioned, we suggest to use the extended pencil formulation as described in the sequel.

Assume the extended pencil $\mathcal{M} - \lambda\mathcal{L}$ is regular, has no eigenvalue on the imaginary axis, and that it has an n–dimensional d-stable deflating subspace. If we then apply Algorithm 1 to this matrix pencil, so that at convergence

$$Z_\infty = \begin{bmatrix} Z_{11} & Z_{12} & Z_{13} \\ Z_{21} & Z_{22} & Z_{23} \\ Z_{31} & Z_{32} & Z_{33} \end{bmatrix},$$

the considerations leading to (20) also show that we can obtain the solution of the LQOC problem from

$$\begin{bmatrix} Z_{12} & Z_{13} \\ Z_{22} & Z_{23} \\ Z_{32} & Z_{33} \end{bmatrix} \begin{bmatrix} X_E \\ F_d \end{bmatrix} = - \begin{bmatrix} Z_{11} \\ Z_{21} \\ Z_{31} \end{bmatrix}. \tag{22}$$

If the explicit solution of the DARE is required we can compute this by solving $X_E = X_d E$.

4.4 Van Dooren's compression technique

In order to deflate infinite eigenvalues and to deal with a (numerically) singular matrix R, a compression technique suggested in [45] can be used. This technique requires the computation of the QR factorization of the last m columns of $\mathcal{H}$ from (12) or $\mathcal{M}$ from (13),

$$\begin{bmatrix} B \\ S \\ R \end{bmatrix} = U \begin{bmatrix} 0 \\ 0 \\ \bar{R} \end{bmatrix},$$

where U is orthogonal. Applying U from the left to $\mathcal{H} - \lambda\mathcal{K}$ or $\mathcal{M} - \lambda\mathcal{L}$, the last m rows and columns, i.e., those corresponding to the infinite eigenvalues, can be deflated and the upper left $2n \times 2n$ part of the resulting matrix pencil has only finite eigenvalues. The sign or disk function methods can then be applied to the extracted $2n \times 2n$ matrix pencils and the stable deflating subspaces can be computed as the stable deflating subspace of these pencils as U does not affect the right deflating subspaces.

We stress that this technique should also be employed if the condition number of R exceeds a certain threshold as an ill-conditioning of R often leads to problems with the sign and disk function iterations (see also Subsection 6.1).

5 A Parallel Library for LQOC Problems

5.1 General Approach and Structure

The numerical algorithms that we have described in the previous sections are all composed of basic matrix computations such as solving linear systems, matrix products, and QR factorizations (with and without column pivoting). Fortunately, there exist modern linear algebra libraries for parallel distributed-memory computers with highly efficient implementations of these operations [15, 44]. The use of these libraries enhances the reliability and improves portability of the optimal control routines. The performance will mostly depend on the efficiencies of the underlying serial and parallel computational linear algebra libraries and the communication routines.

Here we will employ the ScaLAPACK parallel library [15]. This is a freely available library that implements parallel versions of many of the kernels in LAPACK [4], using the message-passing paradigm. ScaLAPACK is based on BLACS for communication and therefore can be ported to any (serial and) parallel architecture with an implementation of the MPI or PVM library [27].

In ScaLAPACK the computations are performed by a logical grid of $n_p = p_r \times p_c$ processes. The processes are mapped onto the physical processors, depending on the available number of these. All data (matrices) have to be distributed among the process grid prior to the invocation of a ScaLAPACK routine. (See [15] for details).

Using the kernels in ScaLAPACK, we have implemented our library PLiCOC for the solution of LQOC problems. The library contains eight driver routines for the LQOC problem and several computational routines for the solution of related equations in control and spectral projection methods. In addition to the methods described in the previous section, we also provide implementations of Newton's method for AREs as described in [11, 12]. These can be used for iterative refinement of solutions computed using the sign or disk function methods. As each iteration step of Newton's method requires the solution of a linear matrix equation, solvers for Lyapunov and Stein equations are also included. All the routines in the library are named following a ScaLAPACK-like notation. After the usual `pd-` prefix (for parallel, double precision), we employ two letters, either `ge` or `gg`, to indicate if the routine deals with a standard problem (related to an eigenproblem for a matrix) or a generalized one (related to an eigenproblem for a matrix pencil). The following two letters specify the purpose of the routine, where the first one usually specifies whether the problem is associated with a continuous-time (`c`) or a discrete-time system (`d`), and the second indicates the problem that is to be solved: `o` for optimal control, `r` for an ARE, and `l` for a Lyapunov or a Stein (discrete Lyapunov) equation. There is an exception to this rule: The spectral projectors based on the sign and disk functions are not strictly associated with a continuous-time or discrete-time problem, so we use both letters to specify its purpose: `si` for the sign function and `di` for the disk function. Finally, the last two letters serve as an indicator of the method that is employed for solving the specific problem as follows:

– LQOC problem. Use of the standard pencil formulation (recommended in case R is well-conditioned) or the extended one: suffixes `-ca` (for Variant A) and `-cb` (for Variant B), respectively.

– Iterative refinement procedures for the ARE based on Newton's method (see [12]): suffix `-ne`.

– Linear matrix equation (LME) solvers using the Newton iteration for the matrix sign function: suffix `-ne`, or Smith's iteration: suffix `-sm`.

– Spectral projection methods: Newton iteration for the sign function: suffix `-ne`, or inverse free iteration for the disk function: suffix `-if`.

Tables 1 and 2 list, respectively, the driver and the computational routines in the PLiCOC library.

	Standard problem		Generalized problem	
	Continuous	Discrete	Continuous	Discrete
Standard form.	`pdgecoca`	`pdgedoca`	`pdggcoca`	`pdggdoca`
Extended form.	`pdgecocb`	`pdgedocb`	`pdggcocb`	`pdggdocb`

Table 1: List of the parallel driver routines in PLiCOC.

	Standard problem		Generalized problem	
Spectral division	`pdgesine`		`pdggsine, pdggdiif`	
	Continuous	Discrete	Continuous	Discrete
ARE refinement	`pdgecrnx`	`pdgecrnx`	`pdggcrnx`	`pdggdrnx`
LME solver	`pdgeclne`	`pdgedlsm`	`pdggclne`	`pdggdlsm`

Table 2: List of the parallel computational routines in PLiCOC.

PLiCOC can be downloaded from

`http://www.pscom.uji.es/software.html#PLiCOC`

In addition to the computational routines, the PLiCOC archive file also contains `HTML` documentation of the driver routines as well as testing routines.

5.2 Implementation Details

Many of the driver routines in the library require the computation of the numerical rank of a matrix. Using the SVD for this purpose is highly expensive in terms of arithmetic computations so we use instead an estimation of the numerical rank based on the QR factorization with column pivoting and an incremental estimator [14]. Setting a tolerance threshold for the numerical rank is a delicate problem, specially if the matrix has no large gap in its singular value distribution. As a general solution, in order to determine the numerical rank of a square matrix of order s, we set the rank tolerance threshold, τ_{rank}, to

$$\tau_{\text{rank}} = 10 \cdot \sqrt{s} \cdot \varepsilon,$$

where ε is the machine precision. We found this threshold to serve our purposes in most of the experiments presented in the next section (only for Example 2, the threshold had to be readjusted).

Most of the computational routines in the library are based on iterative methods; e.g., a Newton refinement procedure is used to improve the accuracy of the solution of the AREs, a Newton iteration is also employed in the computation of the sign function and the solution of Lyapunov equations via this matrix function, Malyshev's (inverse free) iteration is the key to obtain the disk function, etc. In all these cases we set the maximum number of iterations to 30 and use an iteration tolerance threshold, τ_{iter}, defined as

$$\tau_{\text{iter}} = 10 \cdot t \cdot \sqrt{\varepsilon},$$

where t stands for the problem dimension. Table 3 lists the specific convergence criteria employed by the computational routines. As all the iterative algorithms in the library present an ultimately quadratic convergence, once the corresponding threshold is satisfied two more iterations are carried out to guarantee the maximum attainable accuracy.

Standard problem			
`pdgesine`	$\|H_{k+1} - H_k\| < \tau_{\text{iter}} \cdot \|H_k\|$		
Continuous		Discrete	
`pdgecrnx`	$\|R_c(X_k)\| < \tau_{\text{iter}} \cdot \|X_k\|$	`pdgedrnx`	$\|R_d(X_k)\| < \tau_{\text{iter}} \cdot \|X_k\|$
`pdgeclne`	$\|A_k + I_n\| < \tau_{\text{iter}} \cdot \sqrt{n}$	`pdgedlsm`	$\|A_k\| < \tau_{\text{iter}} \cdot \|A\|$
Generalized problem			
`pdggsine`	$\|H_{k+1} - H_k\| < \tau_{\text{iter}} \cdot \|H_k\|$		
`pdggdiif`	$\|R_{k+1} - R_k\| < \tau_{\text{iter}} \cdot \|R_k\|$		
Continuous		Discrete	
`pdggcrnx`	$\|R_c(X_k)\| < \tau_{\text{iter}} \cdot \|X_k\|$	`pdggdrnx`	$\|R_d(X_k)\| < \tau_{\text{iter}} \cdot \|X_k\|$
`pdggclne`	$\|A_k + E\| < \tau_{\text{iter}} \cdot \|E\|$	`pdggdlsm`	$\|E^{-1}A_k\| < \tau_{\text{iter}} \cdot \|A\|$

Table 3: Convergence criteria for the iterations in the computational routines. The Frobenius norm is employed in all cases.

6 Experimental Results

All the experiments presented in this section were performed on a cluster of 9 nodes using IEEE double-precision floating-point arithmetic ($\varepsilon \approx 2.2204 \times 10^{-16}$). Each node consists of four Intel Itanium-2 processors running at 1.5 GHz with 4 GBytes of RAM per node. We employ a BLAS library, specially tuned for the Itanium-2 processor, that achieves around 5.6 Gflops (billions of flops per second) for the matrix product (routine `DGEMM`) on a single processor. The nodes are connected via an *Infiniband* network; the communication library BLACS is based on an implementation of the MPI communication library specially developed and tuned for this network. The performance of the interconnection network was measured by a simple loop-back message transfer resulting in a latency of 8 μs and a bandwidth of 2.70 Gbit/s. We made use of the LAPACK and ScaLAPACK libraries whenever possible.

We evaluate the numerical and parallel performances of the parallel driver routines in PLiCOC using several LQOC problems from [1, 2, 18, 37, 13]. Numerical properties of the examples, in particular several condition numbers and norms that affect the numerical behavior of the algorithms, can be found in those references. Our test bed is composed of seven examples that we describe in brief next. Two of these problems correspond to standard continuous-time LTI systems, three to generalized continuous-time systems, and there is a standard discrete-time system and a generalized one.

Example 1. Standard continuous-time system (Example 3.1 from [1]): The matrices in this example describe a mathematical model for the control of the position and the velocity of a string of L high-speed vehicles. The system has $n = 2L - 1$ states, $m = L$ inputs, and $p = L - 1$ outputs. No particular bad properties arise for growing n so this example is perfectly suited to test the behavior of the parallel algorithms for growing problem size.

Example 2. Standard continuous-time system (Example 3.2 from [1]): All system matrices in this example are circulant, with dimensions $n = m = p$. Growth of the system dimensions does not influence norms or condition numbers of the problem. Thus, the problem is specially well suited to test how an algorithm behaves when the dimension of the problem increases.

Example 3. Standard discrete-time system (Example 4.1 from [2]): This is a system with a single input and $p = n$ outputs, parameterized by a value r that we set in our experiments to 1. Although there is no real application behind this system, we include the case for evaluation purposes of the parallel routines for standard discrete-time systems.

Example 4. Generalized continuous-time system (Example 4.2 from [1]): This model comes from an LQOC problem of one-dimensional (1-D) heat flow. The problem is described in infinite dimensional operators on a Hilbert space. Using a standard finite element approach based on linear B-splines, a finite dimensional approximation to the problem can be obtained by solving an ARE. If L denotes the approximation index, using this approach we obtain a system with $n = L$ states, a single input, and a single output ($m = p = 1$). Increasing the value of L results in a finer grid for the underlying approximation scheme.

The condition number and norms for this example depend on a series of parameters. In our tests with this example we employ the default values provided in [1]: $a = 0.01$, $b = c = 1.0$, $[\beta_1, \beta_2] = [0.2, 0.3]$, and $[\gamma_1, \gamma_2] = [0.2, 0.3]$.

Example 5. Generalized continuous-time system: (Example 4.3 from [1].) The matrices in this example model a string of L coupled springs, dashpots, and masses. The inputs are two forces acting on the left and right extremes of the strings. The system dimensions are $n = 2L$, $m = 2$, and $p = 2L$; the entries of the matrices depend on three parameters. Again we use the defaults from [1]: to $\mu = 4.0$, $\delta = 4.0$, and $\kappa = 1.0$.

Example 6. Generalized discrete-time system (Example 2.8 from [18]): This model comes from a full discretization of a point control problem for a 1-D heat equation obtained using the implicit Crank-Nicholson scheme. The given equation models the heat diffusion in a (1-dimensional) thin rod with a single heat source. The resulting LTI system is a single-input/single-output system. The data set provided in [18] is of relatively small size ($n = 200$). The sampling time (the time steps in the Crank-Nicholson scheme) are $\Delta t = 1$sec., so that the Nyquist frequency is π. We use larger problems obtained by using a finer grid in the discretization scheme.

Example 7. Generalized continuous-time system ([13, 37]): This model arises in an industrial manufacturing process for steel profiles. The goal is to design a control that achieves moderate temperature gradients when the rail is cooled down. The mathematical model corresponds to the boundary control for a 2-D heat equation. A finite element discretization, followed by adaptive adaptive refinement via bisection and different mesh sizes, result in two generalized systems of state-space dimension $n = 1357$ and 5177, with 7 inputs and 6 outputs both.

Unless otherwise stated, the weight matrices are defined as $R = I_m$, $Q = I_p$, and S is initialized with all entries set to zeros. Three different problem sizes are used in our experiments: small, medium, and large, with the specific dimension listed in Table 4. Note that m and p are fixed once n is defined.

Ex.	`Small`	`Medium`	`Large`
1	251	2501	6001
2	250	2500	6000
3	250	2500	6000
4	250	2500	6000
5	250	2500	6000
6	250	2500	6000
7	–	–	5177

Table 4: State-space dimension n of the LQOC problems employed in the experimental evaluation.

Each LQOC example is solved using four different approaches, depending on whether the associated ARE is solved using the standard/extended pencil formulation (Variants A/B, respectively) and whether spectral projection is performed via the sign or disk function.

6.1 Numerical Performance

All experiments in this subsection were performed using Octave 2.1.57 and Matlab M-scripts that exactly mimic the computations as performed in our parallel routines for LQOC. In order to evaluate the numerical performance of the routines in PLiCOC, we consider the small-scale LQOC examples in our benchmark (see Table 4). Let X_{spec} denote the solution computed via the PLiCOC spectral projection methods (sign or disk functions) for the ARE associated with each problem. Table 5 reports the normalized residuals for the continuous and discrete-time systems,

$$\|R_c(X_{\rm spec})\|_F/\|X_{\rm spec}\|_F \quad \text{and} \quad \|R_d(X_{\rm spec})\|_F/\|X_{\rm spec}\|_F,$$

respectively. For each example, the residuals attained with all four methods are close (within two orders of magnitude) except for Example 4, where a slightly larger difference of magnitude is found between Variant A/disk and Variant B/disk. The convergence rates of the methods are also similar, resulting in a similar number of iterations for each example.

Ex.	`Variant A/sign`	`Variant A/disk`	`Variant B/sign`	`Variant B/disk`
1	1.16e-13 (10)	5.67e-14 (9)	2.36e-13 (10)	5.60e-14 (10)
2	9.73e-12 (6)	2.42e-14 (8)	2.93e-14 (6)	4.12e-14 (7)
3	5.12e-12 (11)	4.46e-12 (12)	5.07e-12 (11)	5.11e-12 (12)
4	1.88e-14 (16)	1.64e-12 (18)	1.88e-14 (16)	3.06e-15 (18)
5	1.00e-11 (15)	1.24e-12 (17)	1.74e-11 (15)	8.30e-13 (16)
6	8.09e-10 (15)	8.79e-11 (16)	8.07e-10 (15)	1.46e-10 (16)

Table 5: Normalized residuals of the solutions computed using the PLiCOC spectral projection methods for the small-scale examples. The number inside parentheses indicates the number of iterations required for convergence of the corresponding spectral projection method.

The solutions resulting from the application of the spectral projection methods are next employed as the starting point for the Newton refinement procedure for the ARE. The normalized residuals of the refined solutions, $X_{\rm newton}$, after three iterations of this procedure are reported in Table 6. These results show that the refinement achieves a general reduction of the normalized residuals, which become of the order of the machine precision or smaller in all cases but for Example 6. In that example, all approaches attain basically the same level of accuracy.

In order to illustrate the benefits of using the extended pencil formulation we analyze next the numerical performance of the LQOC codes using Example 1 with a weighting matrix R of increasing ill-conditioning. In particular, we define R to be the diagonal matrix $R = \text{diag}(10^0, 10^{-a}, 10^{-2a}, \ldots, 10^{-(m-1)a})$, with $a = \delta/m$ and $\delta \in \{0, 4, 8, 10, 12, 14\}$. Thus, increasing values of δ produce weighting matrices with larger condition numbers.

Table 7 reports the normalized residuals obtained from using the spectral projection methods and Table 8 lists the normalized residuals after these solutions have been refined with Newton's method. The first table shows the superior numerical performance of solvers based on Variant B when δ is large; the advantage also shows up in a smaller number of iterations. The refined solutions all present similar levels of accuracy. However, in order to attain these residuals using Variant A/sign 10 and 15 iterations of the refinement procedure

Ex.	Variant A/sign	Variant A/disk	Variant B/sign	Variant B/disk
1	2.45e-16	2.41e-16	1.23e-16	1.32e-16
2	9.83e-16	1.01e-15	3.14e-16	3.09e-16
3	7.59e-72	3.20e-71	2.85e-71	4.52e-71
4	2.64e-18	2.71e-18	2.39e-18	2.41e-18
5	1.98e-15	2.05e-15	1.36e-15	1.37e-15
6	1.75e-11	1.67e-11	1.81e-11	1.80e-11

Table 6: Normalized residuals of the solutions refined with the Newton procedure for the small-scale examples.

were needed, respectively, for $\delta = 12$ and 14. Only three refinement iterations were required in the remaining cases.

δ	Variant A/sign	Variant A/disk	Variant B/sign	Variant B/disk
0	1.16e-13 (10)	5.67e-14 (9)	2.36e-13 (10)	5.60e-14 (10)
4	3.18e-14 (9)	1.53e-12 (11)	1.56e-13 (9)	7.52e-13 (10)
8	3.02e-12 (9)	4.16e-09 (15)	1.82e-12 (9)	6.15e-10 (11)
10	7.62e-09 (9)	1.69e-07 (16)	1.75e-12 (9)	3.01e-08 (12)
12	2.38e+00 (17)	6.34e-06 (18)	1.51e-12 (9)	1.77e-06 (13)
14	2.93e+01 (19)	3.35e-03 (20)	2.14e-11 (10)	5.11e-05 (13)

Table 7: Normalized residuals of the solutions computed using the PLiCOC spectral projection methods for Example 1 with ill-conditioned R. The number inside parentheses indicates the number of iterations required for convergence of the corresponding spectral projection method.

6.2 Parallel Performance

In all the results presented in this subsection, the solutions obtained by the spectral projection methods are refined using three iterations of the Newton procedure. The execution times that are reported in the following include the time required to perform the refinement.

Our next experiment is designed to measure the benefits offered by the parallel PLiCOC driver routines in terms of lower execution times. For that purpose, we consider now the medium-scale LQOC examples in the benchmark. Figure 1 shows how by using several processors the execution time required to solve the six examples by means of all approaches (Variants/spectral projectors) is considerably reduced. The acceleration (speed-up) roughly varies between 1.5, when using 2 processors, and 4.7 when 8 processors are employed. Thus, for example, the use of the parallel implementation of Variant B/disk

δ	Variant A/sign	Variant A/disk	Variant B/sign	Variant B/disk
0	2.45e-16	2.41e-16	1.23e-16	1.32e-16
4	6.46e-16	6.51e-16	2.57e-16	2.75e-16
8	1.07e-15	9.93e-16	3.76e-16	3.13e-16
10	1.24e-15	1.32e-15	3.55e-16	3.70e-16
12	1.45e-15 (10)	1.47e-15	3.44e-16	4.14e-16
14	1.43e-15 (15)	1.20e-15	4.20e-16	4.90e-16

Table 8: Normalized residuals of the solutions refined with Newton's method for Example 1 with ill-conditioned R.

using 8 processors reduces the solution time of Example 4 from about 3'5 hours to slightly more than 40 minutes.

Finally, in order to demonstrate the ability of the parallel PLiCOC driver routines to solve large problems, in Table 9 we show the execution times required to solve the large-scale LQOC examples in the benchmark using all the resources of the cluster (36 processors and 36 GB of RAM overall). As could be expected from the theoretical computational costs, the disk function projectors render higher execution times than the sign function projectors. Other differences can be explained by the higher costs of the approaches based on Variant B over Variant A in case m is large and by the specific number of iterations required by each combination of Variant/spectral projection method.

Ex.	Variant A/sign	Variant A/disk	Variant B/sign	Variant B/disk
1	27m 7s (15)	1h 16m 26s (14)	43m 27s (16)	1h 24m 43s (15)
2	14m 38s (8)	42m 34s (8)	22m 6s (7)	41m 50s (7)
3	25m 5s (16)	1h 25m 21s (17)	25m 17s (16)	1h 26m 37s (17)
4	1h 2m 34s (25)	2h 31m 19s (28)	1h 6m 30s (25)	2h 27m 29s (27)
5	1h 3m 30s (25)	2h 18m 10s (26)	1h 5m 45s (25)	2h 15m 20s (25)
6	38m 7s (25)	1h 55m 59s (26)	38m 34s (25)	1h 57m 45s (24)
7	33m 19s (21)	1h 11m 17s (19)	35m 1s (21)	1h 20m 18s (22)

Table 9: Execution time (in hours, minutes, and seconds) of the PLiCOC driver routines required for solving the large-scale LQOC problems. The number inside parentheses indicates the number of iterations required for convergence of the corresponding spectral projection method.

This last experiment demonstrates that the use of the parallel algorithms and platform allowed us to obtain the solutions of these problems in a reasonable time, which varied from 14 minutes in the fastest case to about 2'5 hours in the slowest one. None of these examples could be solved using a single node of the parallel platform due to memory restrictions;

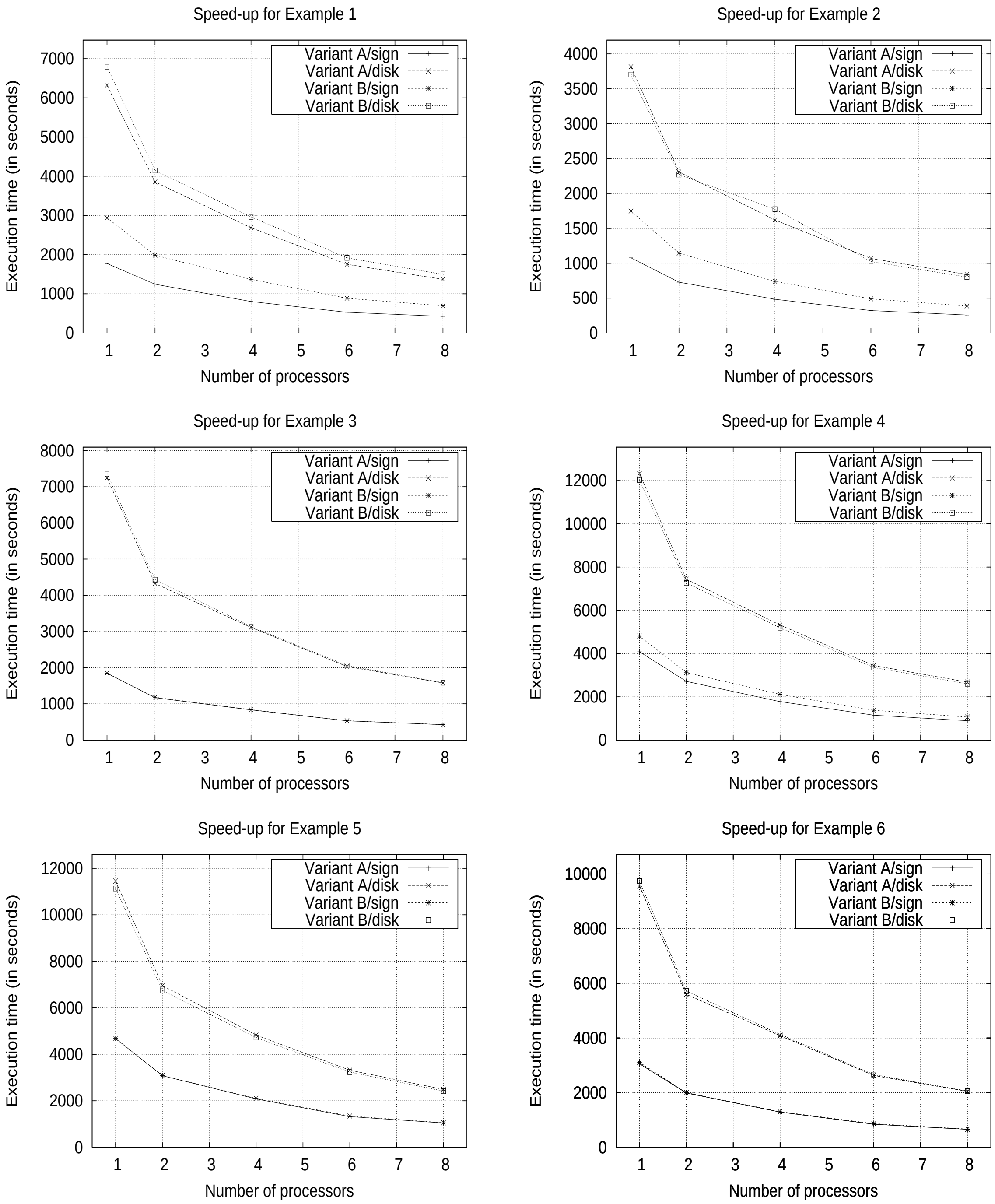

Figure 1: Reduction of the execution time achieved by the PLiCOC parallel driver routines on the medium-scale LQOC problems.

nonetheless, if we have had enough memory to store all data matrices in a single node, the execution times would have been much higher.

7 Concluding Remarks

We have described the library PLiCOC for the solution of LQOC problems on parallel architectures. This library is composed of iterative algorithms which can be easily and efficiently parallelized on distributed-memory platforms given the appropriate parallel numerical linear algebra library. In particular, the current implementation of the library is based on ScaLAPACK.

A collection of benchmark examples shows that, using the kernels in the PLiCOC library, problems involving LTI systems with up to a few thousands of states can be rapidly and accurately solved on a cluster of moderate dimensions. A larger computational parallel platform would enable the solution of systems with state-space dimension of order up to 10^5.

References

[1] J. Abels and P. Benner. CAREX – a collection of benchmark examples for continuous-time algebraic Riccati equations (version 2.0). SLICOT Working Note 1999-14, November 1999. Available from `www.slicot.de`.

[2] J. Abels and P. Benner. DAREX – a collection of benchmark examples for discrete-time algebraic Riccati equations (version 2.0). SLICOT Working Note 1999-15, November 1999. Available from `www.slicot.de`.

[3] H. Abou-Kandil, G. Freiling, V. Ionescu, and G. Jank. *Matrix Riccati Equations in Control and Systems Theory*. Birkhäuser, Basel, Switzerland, 2003.

[4] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, third edition, 1999.

[5] W.F. Arnold, III and A.J. Laub. Generalized eigenproblem algorithms and software for algebraic Riccati equations. *Proc. IEEE*, 72:1746–1754, 1984.

[6] Z. Bai, J. Demmel, J. Dongarra, A. Petitet, H. Robinson, and K. Stanley. The spectral decomposition of nonsymmetric matrices on distributed memory parallel computers. *SIAM J. Sci. Comput.*, 18:1446–1461, 1997.

[7] Z. Bai, J. Demmel, and M. Gu. An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblems. *Numer. Math.*, 76(3):279–308, 1997.

[8] P. Benner. *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*. Logos–Verlag, Berlin, Germany, 1997. *Also:* Dissertation, Fakultät für Mathematik, TU Chemnitz–Zwickau, 1997.

[9] P. Benner. On a numerical method for regularization of descriptor systems. *Z. Angew. Math. Mech.*, 81, Suppl. 2:S707–S708, 2001.

[10] P. Benner and R. Byers. Disk functions and their relationship to the matrix sign function. In *Proc. European Control Conf. ECC 97,* Paper 936. BELWARE Information Technology, Waterloo, Belgium, 1997. CD-ROM.

[11] P. Benner and R. Byers. An exact line search method for solving generalized continuous-time algebraic Riccati equations. *IEEE Trans. Automat. Control*, 43(1):101–107, 1998.

[12] P. Benner, R. Byers, E.S. Quintana-Ortí, and G. Quintana-Ortí. Solving algebraic Riccati equations on parallel computers using Newton's method with exact line search. *Parallel Comput.*, 26(10):1345–1368, 2000.

[13] P. Benner and J. Saak. A semi-discretized heat transfer model for optimal cooling of steel profiles. In P. Benner, V. Mehrmann, and D. Sorensen, editors, *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lecture Notes in Computational Science and Engineering*, pages 353–356. Springer-Verlag, Berlin/Heidelberg, Germany, 2005.

[14] C.H. Bischof. Incremental condition estimation. *SIAM J. Matrix Anal. Appl.*, 11(2):312–322, 1990.

[15] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaley. *ScaLAPACK Users' Guide*. SIAM, Philadelphia, PA, 1997.

[16] A.Y. Bulgakov and S.K. Godunov. Circular dichotomy of the spectrum of a matrix. *Siberian Math. J.*, 29(5):734–744, 1988.

[17] R. Byers. Solving the algebraic Riccati equation with the matrix sign function. *Linear Algebra Appl.*, 85:267–279, 1987.

[18] Y. Chahlaoui and P. Van Dooren. A collection of benchmark examples for model reduction of linear time invariant dynamical systems. SLICOT Working Note 2002–2, February 2002. Available from `www.slicot.de`.

[19] T. Chan. Rank revealing QR factorizations. *Linear Algebra Appl.*, 88/89:67–82, 1987.

[20] B.N. Datta. *Numerical Methods for Linear Control Systems*. Elsevier Academic Press, 2004.

[21] F.R. Gantmacher. *Theory of Matrices*, volume 1. Chelsea, New York, 1959.

[22] J.D. Gardiner and A.J. Laub. A generalization of the matrix-sign-function solution for algebraic Riccati equations. *Internat. J. Control*, 44:823–832, 1986.

[23] R. Goebel and M. Subbotin. Continuous time constrained linear quadratic regulator — convex duality approach. In *Proc. American Cont. Conf. 2005*, volume 2, pages 1401–1406, 2005.

[24] C. Gökcek, P.T. Kabamba, and S.M. Meerkov. An LQR/LQG theory for systems with saturating actors. *IEEE Trans. Automat. Control*, 46(10):1529–1542, 2001.

[25] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition, 1996.

[26] M. Green and D.J.N Limebeer. *Linear Robust Control*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[27] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press, Cambridge, MA, 1994.

[28] B. Hassibi, A.H. Sayed, and T. Kailath. *Indefinite-Quadratic Estimation and Control*. SIAM Publications, Philadelphia, PA, 1999.

[29] S. Huss, E.S. Quintana-Ortí, X. Sun, and J. Wu. Parallel spectral division using the matrix sign function for the generalized eigenproblem. *Int. J. of High Speed Computing*, 11(1):1–14, 2000.

[30] C. Kenney and A.J. Laub. The matrix sign function. *IEEE Trans. Automat. Control*, 40(8):1330–1348, 1995.

[31] V. Kučera. *Analysis and Design of Discrete Linear Control Systems*. Academia, Prague, Czech Republic, 1991.

[32] W.S. Levine, editor. *The Control Handbook*. CRC Press, 1996.

[33] A.N. Malyshev. Parallel algorithm for solving some spectral problems of linear algebra. *Linear Algebra Appl.*, 188/189:489–520, 1993.

[34] V. Mehrmann. *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution*. Number 163 in Lecture Notes in Control and Information Sciences. Springer-Verlag, Heidelberg, July 1991.

[35] A.G.O. Mutambara. *Design and Analysis of Control Systems*. CRC Press, Boca Raton, FL, 1999.

[36] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer-Verlag, 1999.

[37] T. Penzl. Algorithms for model reduction of large dynamical systems. Technical Report SFB393/99-40, Sonderforschungsbereich 393 *Numerische Simulation auf massiv parallelen Rechnern*, TU Chemnitz, 09107 Chemnitz, FRG, 1999. Available from `http://www.tu-chemnitz.de/sfb393/sfb99pr.html`.

[38] E.S. Quintana-Ortí, G. Quintana-Ortí, X. Sun, and R.A. van de Geijn. A note on parallel matrix inversion. *SIAM J. Sci. Comput.*, 22:1762–1771, 2001.

[39] G. Quintana-Orti, X. Sun, and C.H. Bischof. A BLAS-3 version of the QR factorization with column pivoting. *SIAM J. Sci. Comput.*, 19:1486–1494, 1998.

[40] J.D. Roberts. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *Internat. J. Control*, 32:677–687, 1980. (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971).

[41] A. Saberi, P. Sannuti, and B.M. Chen. *H_2 Optimal Control*. Prentice-Hall, Hertfordshire, UK, 1995.

[42] X. Sun and E.S. Quintana-Ortí. The generalized iteration for the matrix sign function. *SIAM J. Sci. Comput.*, pages 669–683, 2002.

[43] X. Sun and E.S. Quintana-Ortí. Spectral division methods for block generalized Schur decompositions. *Math. Comp.*, 73:1827–1847, 2004.

[44] R.A. van de Geijn. *Using PLAPACK: Parallel Linear Algebra Package*. MIT Press, Cambridge, MA, 1997.

[45] P. Van Dooren. A generalized eigenvalue approach for solving Riccati equations. *SIAM J. Sci. Statist. Comput.*, 2:121–135, 1981.

Other titles in this preprint series:

04-01 A. Meyer, F. Rabold, M. Scherzer. Efficient Finite Element Simulation of Crack Propagation. February 2004.

04-02 S. Grosman. The robustness of the hierarchical a posteriori error estimator for reaction-diffusion equation on anisotropic meshes. March 2004.

04-03 A. Bucher, A. Meyer, U.-J. Görke, R. Kreißig. Entwicklung von adaptiven Algorithmen für nichtlineare FEM. April 2004.

04-04 A. Meyer, R. Unger. Projection methods for contact problems in elasticity. April 2004.

04-05 T. Eibner, J. M. Melenk. A local error analysis of the boundary concentrated FEM. May 2004.

04-06 H. Harbrecht, U. Kähler, R. Schneider. Wavelet Galerkin BEM on unstructured meshes. May 2004.

04-07 M. Randrianarivony, G. Brunnett. Necessary and sufficient conditions for the regularity of a planar Coons map. May 2004.

04-08 P. Benner, E. S. Quintana-Ortí, G. Quintana-Ortí. Solving Linear Matrix Equations via Rational Iterative Schemes. October 2004.

04-09 C. Pester. Hamiltonian eigenvalue symmetry for quadratic operator eigenvalue problems. October 2004.

04-10 T. Eibner, J. M. Melenk. An adaptive strategy for hp-FEM based on testing for analyticity. November 2004.

04-11 B. Heinrich, B. Jung. The Fourier-finite-element method with Nitsche-mortaring. November 2004.

04-12 A. Meyer, C. Pester. The Laplace and the linear elasticity problems near polyhedral corners and associated eigenvalue problems. December 2004.

04-13 M. Jung, T. D. Todorov. On the Convergence Factor in Multilevel Methods for Solving 3D Elasticity Problems. December 2004.

05-01 C. Pester. A residual a posteriori error estimator for the eigenvalue problem for the Laplace-Beltrami operator. January 2005.

05-02 J. Badía, P. Benner, R. Mayo, E. Quintana-Ortí, G. Quintana-Ortí, J. Saak. Parallel Order Reduction via Balanced Truncation for Optimal Cooling of Steel Profiles. February 2005.

05-03 C. Pester. CoCoS – Computation of Corner Singularities. April 2005.

05-04 A. Meyer, P. Nestler. Mindlin-Reissner-Platte: Einige Elemente, Fehlerschätzer und Ergebnisse. April 2005.

05-05 P. Benner, J. Saak. Linear-Quadratic Regulator Design for Optimal Cooling of Steel Profiles. April 2005.

05-06 A. Meyer. A New Efficient Preconditioner for Crack Growth Problems. April 2005.

05-07 A. Meyer, P. Steinhorst. Überlegungen zur Parameterwahl im Bramble-Pasciak-CG für gemischte FEM. April 2005.

05-08 T. Eibner, J. M. Melenk. Fast algorithms for setting up the stiffness matrix in hp-FEM: a comparison. June 2005.

05-09 A. Meyer, P. Nestler. Mindlin-Reissner-Platte: Vergleich der Fehlerindikatoren in Bezug auf die Netzsteuerung Teil I. June 2005.

05-10 A. Meyer, P. Nestler. Mindlin-Reissner-Platte: Vergleich der Fehlerindikatoren in Bezug auf die Netzsteuerung Teil II. July 2005.

05-11 A. Meyer, R. Unger. Subspace-cg-techniques for clinch-problems. September 2005.

05-12 P. Ciarlet, Jr, B. Jung, S. Kaddouri, S. Labrunie, J. Zou. The Fourier Singular Complement Method for the Poisson Problem. Part III: Implementation Issues. October 2005.

05-13 T. Eibner, J. M. Melenk. Multilevel preconditioning for the boundary concentrated *hp*-FEM. December 2005.

05-14 M. Jung, A. M. Matsokin, S. V. Nepomnyaschikh, Yu. A. Tkachov. Multilevel preconditioning operators on locally modified grids. December 2005.

05-15 S. Barrachina, P. Benner, E. S. Quintana-Ortí. Solving Large-Scale Generalized Algebraic Bernoulli Equations via the Matrix Sign Function. December 2005.

05-16 B. Heinrich, B. Jung. Nitsche- and Fourier-finite-element method for the Poisson equation in axisymmetric domains with re-entrant edges. December 2005.

05-17 M. Randrianarivony, G. Brunnett. C-Null paving of closed meshes with quadrilateral patches. December 2005.

05-18 M. Randrianarivony, G. Brunnett. Quadrilateral removal and 2-ear theorems. December 2005.

05-19 P. Benner, E. S. Quintana-Ortí;, G. Quintana-Ortí. Solving linear-quadratic optimal control problems on parallel computers. December 2005.

The complete list of current and former preprints is available via
`http://www.tu-chemnitz.de/sfb393/preprints.html`.