NICONET Report 1999-1

A Portable Subroutine Library for Solving Linear Control Problems on Distributed Memory Computers [1]

Peter Benner[2] Enrique S. Quintana-Ortí[3] Gregorio Quintana-Ortí[4]

January 1999

[1]This paper presents research results of the European Community BRITE-EURAM III Thematic Networks Programme NICONET (contract number BRRT-CT97-5040) and is distributed by the Working Group on Software WGS. *WGS secretariat*: Mrs. Ida Tassens, ESAT - Katholieke Universiteit Leuven, K. Mercierlaan 94, 3001-Leuven-Heverlee, BELGIUM. This report is also available by anonymous ftp from *wgs.esat.kuleuven.ac.be* in the directory *pub/WGS/REPORTS/nic1999-1.ps.Z*

[2]Zentrum für Technomathematik, Fachbereich 3 – Mathematik und Informatik, Universität Bremen, D–28334 Bremen, Germany; `benner@math.uni-bremen.de`.

[3]Departamento de Informática, Universidad Jaime I, 12080 Castellón, Spain; `quintana@inf.uji.es`

[4]Departamento de Informática, Universidad Jaime I, 12080 Castellón, Spain; `gquintan@inf.uji.es`

Abstract

This paper describes the design of a software library for solving the basic computational problems that arise in analysis and synthesis of linear control systems. The library is intended for use in high performance computing environments based on parallel distributed memory architectures. The portability of the library is ensured by using the BLACS, PBLAS, and ScaLAPACK as the basic layer of communication and computational routines. Preliminary numerical results demonstrate the performance of the developed codes on parallel computers. The suggested library can serve as a basic layer for PSLICOT, a parallel extension of the *Subroutine Library in Control Theory* (SLICOT).

1 Introduction

In recent years, many new and reliable numerical methods have been developed for 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:

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

Discrete-time LTI system:

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

In both cases, $A, E \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and $C \in \mathbb{R}^{p\times n}$. Here we assume that E is nonsingular. Descriptor systems with singular E also lead —after appropriate transformations— to the above problem formulation with a nonsingular (and usually, diagonal or triangular) matrix E; see, e.g., [40, 53].

The traditional approach to design a regulator for the above LTI systems involves the minimization of a cost functional of the form

$$
\mathcal{J}_c(x_0, u) \;=\; \frac{1}{2}\int_0^\infty \Big(y(t)^TQy(t) + 2y(t)^TSu(t) + u(t)^TRu(t)\Big)\,dt \tag{3}
$$

in the continuous-time case, and

$$
\mathcal{J}_d(x_0, u) \;=\; \frac{1}{2}\sum_{k=0}^{\infty}\Big(y_k^TQy_k + 2y_k^TSu_k + u_k^TRu_k\Big) \tag{4}
$$

in the discrete-time case. The matrices $Q \in \mathbb{R}^{p\times p}$, $S \in \mathbb{R}^{p\times m}$, and $R \in \mathbb{R}^{m\times m}$ are chosen in order to weight inputs and outputs of the system. The *linear-quadratic regulator problem* consists in minimizing (3) or (4) subject to the dynamics (1) or (2), respectively. It is well-known that the solution of this optimal control problem is given by the closed-loop control

$$
u^*(t) = -R^{-1}(B^TX_cE + S^TC)x(t) =: F_cx(t), \qquad t \geq 0, \tag{5}
$$

in the continuous-time case, and

$$
u_k^* = -(R + B^TX_dB)^{-1}(B^TX_dA + S^TC)x_k =: F_dx_k, \qquad k = 0,1,2,\ldots, \tag{6}
$$

in the discrete-time case. See, e.g., [1, 40, 36] for details and further references. The matrices X_c and X_d in (5) and (6) denote particular solutions of the *(generalized) continuous-time algebraic Riccati equation* (CARE)

$$
\begin{array}{rcl}
0 = \mathcal{R}_c(X) & := & C^TQC + A^TXE + E^TXA - \\
& - & (E^TXB + C^TS)R^{-1}(B^TXE + S^TC),
\end{array} \tag{7}
$$

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

$$\begin{aligned} 0 \; = \; \mathcal{R}_d(X) \;\; &:= \;\; C^T QC + A^T XA - E^T XE - \\ &- \;\; (A^T XB + C^T S)(R + B^T XB)^{-1}(B^T XA + S^T C). \end{aligned} \tag{8}$$

The optimal control in (5) and (6) is obtained from the *stabilizing* solutions of (7) and (8). That is, we need to compute X_c and X_d such that the resulting *closed-loop* matrices

$$A_c := A + BF_c, \qquad F_c := -R^{-1}(B^T X_c E + S^T C),$$

and

$$A_d := A + BF_d, \qquad F_d := -(R + B^T X_d B)^{-1}(B^T X_d A + S^T C)$$

have stable spectra. In the continuous-time case this means that A_c has all its eigenvalues in the open left half plane while in the discrete-time case all the eigenvalues of A_d are of modulus less than one. We will call matrices (matrix pencils) with all eigenvalues in the open left half plane *c–stable* and those with spectra inside the open unit disk will be called *d–stable*. The matrices F_c and F_d are called the *optimal feedback gain matrices*. Under standard assumptions on LTI systems and the weighting matrices in the cost functionals $\mathcal{J}_c$ and $\mathcal{J}_d$, the stabilizing solutions of the CARE and DARE exist, are unique, and symmetric. See [36] for a detailed account on conditions for existence and uniqueness of solutions to (7) and (8).

The algebraic Riccati equations in (7) and (8) can be formulated in the more general forms

$$0 \; = \; \mathcal{R}_c(X) \; = \; \tilde{Q} + \tilde{A}^T X\tilde{E} + \tilde{E}^T X\tilde{A} - X\tilde{G}X \tag{9}$$

for the CARE, and

$$\begin{aligned} 0 \; = \; \mathcal{R}_d(X) \;\; &= \;\; \tilde{Q} + \tilde{A}^T X\tilde{A} - \tilde{E}^T X\tilde{E} - \\ &- \;\; (\tilde{A}^T X\tilde{B} + \tilde{S})(\tilde{R} + \tilde{B}^T X\tilde{B})^{-1}(\tilde{B}^T X\tilde{A} + \tilde{S}^T) \end{aligned} \tag{10}$$

for the DARE. Written in this form, they also include the algebraic Riccati equations arising in many areas of modern control theory like robust control, H_2- and H_∞-control, model reduction, etc.; see, e.g., [27, 45, 46, 54]. The algorithms used here for the numerical solution of equations of the forms (9) and (10) do not depend on the particular form given in (7) and (8) and hence can be used to solve any algebraic Riccati equations given as in (9) and (10).

In the course of solving the above nonlinear systems of equations via Newton's method and in many other analysis and synthesis problems for LTI control problems, linear matrix equations of the form

$$\hat{A}X\hat{B} + \hat{C}X\hat{D} + \hat{E} = 0 \tag{11}$$

have to be solved. Here $\hat{A}, \hat{C} \in \mathbb{R}^{n\times n}$, $\hat{B}, \hat{D} \in \mathbb{R}^{m\times m}$, and $\hat{E}, X \in \mathbb{R}^{n\times m}$. Linear systems of equations as in (11) are called *generalized Sylvester equations.* Some particular instances of (11) are given below:

$$\hat{A}X + X\hat{D} + \hat{E} \; = \; 0, \qquad (\textit{Sylvester equation}) \tag{12}$$

$$\hat{A}X\hat{B} - X + \hat{E} \; = \; 0, \qquad (\textit{“discrete” Sylvester equation}) \tag{13}$$

and for $\hat{E} = \hat{E}^T$,

$$
\begin{aligned}
\hat{A}X + X\hat{A}^T + \hat{E} &= 0, && (\textit{Lyapunov equation}) && (14)\\
\hat{A}X\hat{C}^T + \hat{C}X\hat{A}^T + \hat{E} &= 0, && (\textit{generalized Lyapunov equation}) && (15)\\
\hat{A}X\hat{A}^T - X + \hat{E} &= 0, && (\textit{Stein equation}) && (16)\\
\hat{A}X\hat{A}^T - \hat{C}X\hat{C}^T + \hat{E} &= 0, && (\textit{generalized Stein equation}) && (17)
\end{aligned}
$$

Stein equations are often also referred to as *discrete Lyapunov equations.*

In addition to the above, we will consider special cases of (generalized) Lyapunov and Stein equations where $\hat{E}$ is semidefinite and factored as $\hat{E} = \pm\hat{E}_1\hat{E}_1^T$. In this case, if $\hat{A} - \lambda\hat{C}$ is a stable matrix pencil, then the solution of the corresponding Lyapunov or Stein equation is also semidefinite and can be factored as $X = \pm X_1 X_1^T$. This is the case, e.g., when computing the *controllability Gramian* W_c and *observability Gramian* W_o of a continuous-time LTI system via the Lyapunov equations

$$
\begin{aligned}
AW_cE^T + EW_cA^T + BB^T &= 0, && (18)\\
A^TW_oE + E^TW_oA + C^TC &= 0. && (19)
\end{aligned}
$$

In the discrete-time case these Gramians are given by the corresponding Stein equations

$$
\begin{aligned}
AW_cA^T + EW_cE^T + BB^T &= 0, && (20)\\
A^TW_oA - E^TW_oE + C^TC &= 0. && (21)
\end{aligned}
$$

The Gramians of LTI systems play a fundamental role in many analysis and design problems of LTI systems as computing balanced, minimal, or partial realizations, the Hankel singular values and Hankel norm of LTI systems, and model reduction. Often, the *Cholesky factors* X_1 of the solutions to the above equations are needed. Hence, special algorithms are designed to compute these factors without ever forming the solution matrix explicitly.

We consider special algorithms for all the above equations. The subroutines resulting from implementing these algorithms will be used in order to tackle some computational problems for LTI systems:

- **C1** stabilize an LTI system, i.e., find $F \in \mathbb{R}^{m\times n}$ such that $E - \lambda(A + BF)$ is a stable matrix pencil;
- **C2** model reduction, i.e., find low-order matrices (E_r, A_r, B_r, C_r) such that the LTI system defined by these matrices approximates the input-output behavior of the original system;
- **C3** solve the linear-quadratic optimization problems discussed above using (5) and (6);
- **C4** compute the optimal H_2 controller;
- **C5** compute a suboptimal H_∞ controller.

In addition to the computational subroutines provided by the PBLAS and ScaLAPACK [16] and the solvers for the above linear and nonlinear matrix equations we will also need tools for the spectral decomposition of matrices and matrix pencils in order to accomplish Task **C1**.

The need for parallel computing in this area can be seen from the fact that already for a system with state-space dimension $n = 1000$, the corresponding Sylvester, Lyapunov, Stein, or Riccati equations represent a set of linear or nonlinear equations with one million unknowns. Systems of such a dimension driven by ordinary differential(-algebraic) equations are not uncommon in chemical engineering applications, are standard for second order systems arising from modeling mechanical multibody systems or large flexible space structures, and represent rather coarse grids when derived from the discretization of partial differential equations. We assume here that the coefficient matrices are dense. If sparsity of matrices is to be exploited, other computational techniques have to be employed. The algorithms considered here are implemented in Fortran 77 using the kernels in libraries BLACS, PBLAS, and ScaLAPACK. The resulting subroutines will form a subroutine library with tentative name **PLILCO**, **P**arallel Software **Li**brary for **l**inear **Co**ntrol theory.

This prospectus of the future PLILCO is organized as follows. In Section 2 we will review the basic numerical algorithms that can be employed in order to solve the computational problems needed to accomplish the required tasks. In order to obtain a high portability of the subroutines to be implemented, we will follow the guidelines and computation model used in ScaLAPACK [16] as well as the implementation and documentation standards given in [17]. A short review of the parallel computing paradigms used and a survey of the design and contents of the prospective library will be given in Section 3. Preliminary results in Section 4 will demonstrate the performance of the developed subroutines in several parallel computing environments with shared/distributed memory. An outlook on future activities is given in Section 5.

2 Numerical Algorithms

2.1 The QR and QZ algorithms

The traditional approaches to solving the computational problems introduced in the preceding section involve the computation of invariant/deflating subspaces by means of the QR/QZ algorithms; see, e.g., [26, 47].

The QR algorithm consists of an initial reduction step which transforms a given matrix $A \in \mathbb{R}^{n \times n}$ to upper Hessenberg form, i.e.,

$$A_0 \;:=\; U_0^T A U_0 \;=\; \left[\ \diagdown\!\!\!\diagdown\ \right], \tag{22}$$

where U_0 is orthogonal. Afterwards, a sequence of similarity transformations $A_{j+1} := U_{j+1}^T A_j U_{j+1}$ for $j = 0, 1, 2, \ldots$ is performed. The transformation matrices U_j are chosen such that all iterates A_j are upper Hessenberg matrices and converge to upper quasi-triangular form. That is, if $\tilde{A}_* = \lim_{j\to\infty} A_j$, then $\tilde{A}_*$ is upper triangular with 1×1 and 2×2 blocks on the diagonal. The 1×1 blocks correspond to real eigenvalues of A while 2×2 blocks represent pairs of complex conjugate eigenvalues of A. Usually, convergence takes place in $\mathcal{O}(n)$ iterations. The similarity transformations with U_j can be implemented at a computational cost of $\mathcal{O}(n^2)$ such that the overall computational cost of this algorithm is $\mathcal{O}(n^3)$. If we set $\tilde{U} := \lim_{j\to\infty} \prod_{k=0}^{j} U_j$, then $\tilde{A}_* = \tilde{U}^T A \tilde{U}$. The upper quasi-triangular matrix $\tilde{A}_*$ is called the *(real) Schur form* of A. Applying a finite sequence of orthogonal similarity transformations to $\tilde{A}_*$, the diagonal blocks can be swapped such that the upper $k \times k$ block of the transformed matrix contains those eigenvalues of A that are inside some subset of the complex plain which is closed under complex conjugation. If we denote the accumulated transformation matrices that achieve this re-ordering by $\hat{U}$ and set $U := \hat{U}\tilde{U}$ then the first k columns of U span the A-invariant subspace corresponding to these eigenvalues.

The QZ algorithm applied to matrix pencil $A - \lambda E$ computes orthogonal matrices $\tilde{U}, \tilde{Z} \in \mathbb{R}^{n \times n}$ such that $\tilde{U}^T (A - \lambda E)\tilde{Z} \;=\; A_* - \lambda E_*$, where A_* is upper quasi-triangular and E_* is upper triangular. Again the matrices $\tilde{U}, \tilde{Z}$ can be chosen such that the first k columns of $Z := \hat{Z}\tilde{Z}$ span a particular right deflating subspace of $A - \lambda E$ corresponding to some desired subset of eigenvalues of $A - \lambda E$. The QZ algorithm is equivalent to applying the QR algorithm to AE^{-1} without ever forming the product or the inverse explicitly. The matrix pencil $A_* - \lambda E_*$ is called the *generalized (real) Schur form* of $A - \lambda E$.

In order to compute a spectral decomposition of a matrix or matrix pencil as required, e.g., in Task **C1**, the QR (QZ) algorithm can be applied to the matrix (pencil). The re-ordering must then be performed such that the spectrum of the leading $k \times k$ diagonal block of A_* ($A_* - \lambda B_*$) corresponds to the eigenvalues on the one side of the line dividing the spectrum while the trailing diagonal block corresponds to the eigenvalues of the other side of this line. For continuous-time systems, usually a spectral division along the imaginary axis is needed while for discrete-time systems, the usual spectral division line is the unit circle.

When solving the symmetric linear matrix equations (14)–(17) with the most widely used method, the *Bartels-Stewart method*, the QR and QZ algorithms are used for initial reductions of the involved matrix $\hat{A}$ or matrix pencil $\hat{A} - \lambda\hat{C}$ to upper quasi-triangular form. This initial stage is followed by a backsubstitution process in order to solve the resulting triangular systems. Note that the main computational work is done during the initial reduction. This approach is used, e.g., in [5, 23, 24, 43] for the equations (14)–(17) and also when solving semidefinite Lyapunov and Stein equations of the form (18)–(21) in [28, 52, 43].

For the nonsymmetric equations (12) and (13), it is usually sufficient to transform one of the coefficient matrices to upper quasi-triangular form and the other one to Hessenberg form. This approach is called the *Hessenberg-Schur method* following [25] and is extended to (11) in [23, 24].

The algebraic Riccati equations (9) and (10) can be solved via the QR/QZ algorithms using the relation to certain invariant or deflating subspaces of the corresponding matrices/matrix pencils. If the stable right deflating subspace of

$$H - \lambda K \;:=\; \begin{bmatrix} \tilde{A} & \tilde{G} \\ \tilde{Q} & -\tilde{A}^T \end{bmatrix} - \lambda \begin{bmatrix} \tilde{E} & 0 \\ 0 & \tilde{E}^T \end{bmatrix} \tag{23}$$

is spanned by $\left[\begin{smallmatrix} Z_{11} \\ Z_{21} \end{smallmatrix}\right]$, $Z_{11}, Z_{21} \in \mathbb{R}^{n \times n}$, and Z_{11} is invertible, the stabilizing solution of (9) is given by $X_c = -Z_{12}Z_{11}^{-1}\tilde{E}^{-1}$. Hence the CARE (9) can be solved applying the QZ algorithm to $H - \lambda K$ and re-ordering the eigenvalues such that the stable eigenvalues (i.e., those with negative real parts) appear in the upper $n \times n$ diagonal block of the generalized Schur form of $H - \lambda K$. Then the first n columns of the matrix Z computed by the QZ algorithm span the required stable right deflating subspace of $H - \lambda K$. Note that the optimal control $u_*(t)$ can be computed using $F_c = R^{-1}(B^T Z_{12} Z_{11} - S^T C)$ without solving the CARE explicitly. In case

$E = I_n$, it is sufficient to apply the QR algorithm to the *Hamiltonian matrix* H from (23) and to order the Schur form of H accordingly. This approach was first suggested in [37] and outlined in [3] for $E \neq I_n$. The resulting methods are called the *(generalized) Schur vector methods.*

Similar observations as in the continuous-time case lead to Schur vector methods for DAREs as given in (10). Here the QZ algorithm and an appropriate re-ordering are to be applied to

$$M - \lambda L = \begin{bmatrix} \tilde{A} & 0 & \tilde{B} \\ \tilde{Q} & -\tilde{E}^T & \tilde{S} \\ \tilde{S}^T & 0 & \tilde{R} \end{bmatrix} - \lambda \begin{bmatrix} \tilde{E} & 0 & 0 \\ 0 & -\tilde{A}^T & 0 \\ 0 & -\tilde{B}^T & 0 \end{bmatrix}. \tag{24}$$

If the generalized Schur form of $M - \lambda L$ is ordered such that the leading $n \times n$ diagonal block contains the eigenvalues inside the unit disk, then the first n columns of the Z-matrix computed by the QZ algorithm span the stable (with respect to the unit circle) right deflating subspace of $M - \lambda L$. Partitioning these n columns of Z as $[Z_{11}^T, Z_{21}^T, Z_{31}^T]^T$, where $Z_{11}, Z_{21} \in \mathbb{R}^{n\times n}$, $Z_{31} \in \mathbb{R}^{m\times n}$, and assuming Z_{11} nonsingular, $X_d = Z_{21}Z_{11}^{-1}\tilde{E}^{-1}$ and $F_d = Z_{31}Z_{11}^{-1}$ [40]. Note that using this approach it is possible to compute the optimal control u_k^* directly without solving the DARE explicitly. The computational cost of this approach can be lowered if R is invertible and well-conditioned by applying the QZ algorithm to

$$\tilde{M} - \lambda \tilde{L} = \begin{bmatrix} \tilde{A} - \tilde{R}^{-1}\tilde{S} & 0 \\ \tilde{Q} - \tilde{S}^T\tilde{R}^{-1}\tilde{S} & \tilde{E}^T \end{bmatrix} - \lambda \begin{bmatrix} \tilde{E} & -\tilde{B}\tilde{R}^{-1}\tilde{B}^T \\ 0 & (\tilde{A} - \tilde{R}^{-1}\tilde{S})^T \end{bmatrix}. \tag{25}$$

If $E = I_n$, $\tilde{M} - \lambda \tilde{L}$ is a symplectic matrix pencil. These Schur vector methods for the discrete-time case have been proposed in [3, 42, 51].

If the standard approaches to the spectral division problem and to the solution of the linear and nonlinear matrix equations described above are to be used for computations on parallel distributed memory computers, we will need efficient implementations of the QR and QZ algorithms for these computing environments. In ScaLAPACK, only the QR algorithm is available so far. However, in order to solve the linear matrix equations considered here, the QR algorithm can only be used for (12)–(14) and (16). In all other cases, the QZ algorithm has to be employed in the initial stage when solving these equations via the most frequently used Hessenberg-Schur and Bartels-Stewart methods as described above. Solving (9) and (10) by the (generalized) Schur vector methods, again the QR algorithm can only be used in the CARE case with $E \neq I_n$; for all other cases, the QZ algorithm is needed.

A different approach to solving the algebraic Riccati equations (9) and (10) is to consider these equations as nonlinear sets of equations. From this perspective, the most obvious choice to solve algebraic Riccati equations is Newton's method. In each iteration step of Newton's method applied to CAREs or DAREs [3, 32, 35, 36, 40], a (generalized) Lyapunov or Stein equation of the form (14)–(17) has to be solved; see Section 2.4 below. Thus a parallel implementation of Newton's method also depends heavily on the parallel performance of the Lyapunov or Stein solver employed, i.e., if the Bartels-Stewart method is to be used, once more on the efficiency of the parallelized QR/QZ algorithms.

From the above considerations we can conclude that in order to use the traditional algorithms for solving linear and algebraic Riccati matrix equations, it is necessary to have efficient parallelizations of the QR and QZ algorithms. However, several experimental studies report the difficulties in parallelizing the double implicit shifted QR algorithm on parallel distributed multiprocessors (see, e.g., [18, 29, 30, 49]). The algorithm presents a fine granularity which introduces performance looses due to communication start-up overhead (latency). Besides, traditional data layouts (column/row block scattered) lead to an unbalanced distribution of the computational load. A different approach relies on a block Hankel distribution, which improves the balancing of the computational load [30]. Attempts to increase the granularity by employing multishift techniques have been recently proposed in [31]. Nevertheless, the parallelism and scalability of these algorithms are still far from those of matrix multiplications, matrix factorizations, triangular linear systems solvers, etc.; see, e.g., [16] and the references given therein.

Although the parallelization of the QR algorithm has been thoroughly studied, in contrast, the parallelization of the QZ algorithm remains unexplored to the best of our knowledge. Moreover, since both the QR and the QZ algorithms are composed of the same type of fine-grain computations, similar or even worse parallelism and scalability results are to be expected from the QZ algorithm.

In order to avoid the problems arising from the difficult parallelization of the QR and QZ algorithms, we will use a different computational approach here. It is well-known that under suitable assumptions, the above matrix equations can be solved via the sign function method. It has long been acknowledged that algorithms based on the sign function are relatively easy to parallelize. The methods that will be employed in the PLILCO will be considered in the next sections.

2.2 The Sign Function Method and the Smith Iteration

The sign function method was first introduced in 1971 by Roberts [44] for solving algebraic Riccati equations of the form (9) with $E = I_n$. Roberts also shows how to solve stable Sylvester and Lyapunov equations via the matrix sign function. The application to CAREs and DAREs with $E \neq I_n$ is investigated in [21, 22] while the application to (14) with $E \neq I_n$ is examined in [14].

The computation of the sign function requires basic numerical linear algebra tools like matrix multiplication, inversion and/or solving linear systems. These computations are implemented efficiently on most parallel architectures and, in particular, ScaLAPACK [16] provides easy to use and portable computational kernels for these operations. Hence, the sign function method is an appropriate tool to design and implement efficient and portable numerical software for distributed memory parallel computers.

Let $Z \in \mathbb{R}^{n\times n}$ have no eigenvalues on the imaginary axis and denote by $Z = S \left[\begin{smallmatrix} J^- & 0 \\ 0 & J^+ \end{smallmatrix}\right] S^{-1}$ its Jordan decomposition with $J^- \in \mathbb{C}^{k\times k}$, $J^+ \in \mathbb{C}^{(n-k)\times(n-k)}$ containing the Jordan blocks corresponding to the eigenvalues in the open left and right half planes, respectively. Then the *matrix sign function* of Z is defined as

$$\mathrm{sign}\,(\mathrm{Z}) \;:=\; \mathrm{S} \begin{bmatrix} -I_k & 0 \\ 0 & I_{n-k} \end{bmatrix} \mathrm{S}^{-1}. \tag{26}$$

Note that $\mathrm{sign}\,(\mathrm{Z})$ is unique and independent of the order of the eigenvalues in the Jordan decomposition of Z (see, e.g., [36, Section 22.1]). Many other equivalent definitions for sign (Z) can be given; see, e.g., the recent survey paper [33].

The application of the matrix sign function method to a matrix pencil $Z - \lambda Y$ as given in [21] in case Z and Y are nonsingular can be presented as

$$Z_0 \;:=\; Z, \qquad Z_{k+1} \;:=\; \frac{1}{2c_k}\left(Z_k + c_k^2 Y Z_k^{-1} Y\right), \qquad k = 0, 1, 2, \ldots, \tag{27}$$

where c_k is a scaling parameter. E.g., for determinantal scaling, c_k is given as $c_k = \left(\frac{|\det(Z_k)|}{|\det(Y)|}\right)^{\frac{1}{n}}$ [21]. This iteration is equivalent to computing the sign function of the matrix $Y^{-1}Z$ via the standard Newton iteration as proposed in [44]. The property needed here is that if $Z_\infty := \lim_{k\to\infty} Z_k$, then $(Z_\infty - Y)/2$ (or $(Z_\infty + Y)/2$) defines the skew projection onto the stable (or anti-stable) right deflating subspace of $Z - \lambda Y$ parallel to the anti-stable (or stable) deflating subspace.

In [21] the iteration (27) is used to compute the stabilizing solution of the CARE (9) and the DARE (10) using the matrix pencils (23) and (25). The algebraic Riccati equations (9) can be solved by applying (27) to $Z - \lambda Y = H - \lambda K$ and then forming the resulting projector $Z_\infty - Y$ onto the stable deflating subspace of $H - \lambda K$. A basis of this subspace is then given by the range of that projector. This subspace is usually not computed explicitly as $X_E := X_c\tilde{E}$ can be obtained by solving the overdetermined but consistent set of linear equations

$$\left[\begin{array}{c} Z_{12} \\ Z_{22} + \tilde{E}^T \end{array}\right] X_E = \left[\begin{array}{c} Z_{11} + \tilde{E} \\ Z_{21} \end{array}\right], \tag{28}$$

where $Z_\infty = \left[\begin{smallmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{smallmatrix}\right]$; see [19, 21, 36, 44]. The matrix X_c can be obtained by solving $X_E = X_c\tilde{E}$ while the optimal gain matrix and therefore the optimal control is obtained directly using $F_c = -R^{-1}(B^T X_E + S^T C)$.

The DARE (10) can not be solved directly using the sign function method as we need the d–stable deflating subspace of $\tilde{M} - \tilde{L}$ from (25). One possibility to switch back-and-forth between c– and d–stable matrix pencils $A - \lambda B$ (or c– and d–stable deflating subspaces) is the *Cayley transformation*

$$C_\mu(A - \lambda B) = (\mu A + B) - \lambda(A - \mu B), \qquad |\mu| = 1, \;\; \det(A - \mu B) \neq 0.$$

In order to keep computations real, one has to choose $\mu = \pm 1$; here we restrict ourselves to $\mu = 1$. It is well-known (see, e.g., [38, 41]) that if $A - \lambda B$ is c–stable (d–stable), then $C_\mu(A - \lambda B)$ is d–stable (c–stable) and the c–stable (d–stable) right deflating subspace of $A - \lambda B$ is the d–stable (c–stable) right deflating subspace of $C_\mu(A - \lambda B)$. Hence, the DARE (10) can be solved with the sign function method applied to $C_\mu(\tilde{M} - \lambda\tilde{L})$. The solution X_d is then obtained from (28) replacing X_c by X_d.

Note that none of the methods considered so far can be used to solve (10) via (24): as we need the d–stable deflating subspace of $M - \lambda L$, the sign function method can not be applied directly. Though this subspace is given by the c–stable right deflating subspace of the Cayley transformed matrix pencil $C_\mu(M - \lambda L)$, the sign function method can in general not be used here as $M + L$ and $M - L$ may be singular.

A different approach to solve the spectral division problem and the considered matrix equations is reviewed in Section 2.3. This approach will also overcome the problems for the DARE (10) mentioned above.

The (generalized) Lyapunov and Stein equations (14) are special instances of the CARE (9) and DARE (10), respectively. This implies that one can solve (14) and (15) by means of the sign function method applied to the matrix pencil in (23) which then takes the form

$$H - \lambda K \;=\; \begin{bmatrix} \hat{A} & 0 \\ \hat{E} & -\hat{A}^T \end{bmatrix} - \lambda \begin{bmatrix} \hat{C} & 0 \\ 0 & \hat{C}^T \end{bmatrix}. \tag{29}$$

For stable matrix pencils $\hat{A} - \lambda \hat{C}$, $H - \lambda K$ is regular and has an n-dimensional stable deflating subspace such that the solution of (14) can be obtained analogously to that of (9).

In [14] it is observed that applying the generalized Newton iteration (27) to the matrix pencil $H - \lambda K$ in (29) and exploiting the block-triangular structure of all matrices involved, (27) boils down to

$$\begin{array}{llll} A_0 := \hat{A}, & A_{k+1} := \dfrac{1}{2}\left(A_k + \hat{C} A_k^{-1} \hat{C}\right), & & \\ E_0 := \hat{E}, & E_{k+1} := \dfrac{1}{2}\left(E_k + \hat{C}^T A_k^{-T} E_k A_k^{-1} \hat{C}\right), & & k = 0, 1, 2, \ldots \end{array} \tag{30}$$

and that $X = \frac{1}{2}\hat{C}^{-T}\left(\lim_{k\to\infty} E_k\right)\hat{C}^{-1}$. In case $\hat{C} = I_n$, the iteration in (30) has already been derived by Roberts [44]. The semidefinite Lyapunov equations as in (18)–(21) can be solved using a factored version of the iteration for the $\hat{E}_k$'s in (30), i.e., the iteration is performed starting with the factor $\hat{F}$ of $\hat{E} = \hat{F}^T\hat{F}$. This iteration then converges to $2X_1\hat{C}$ if the solution is factored as $X = X_1^T X_1$. Details of this algorithm can be found in [14] and its application to computing the system Gramians for continuous-time LTI systems as given in (18), (19) is described in [11].

In case the spectra of $\hat{A} - \lambda \hat{C}$ and $\hat{B} - \lambda \hat{D}$ satisfy $\sigma(\hat{A}, \hat{C}) \subset \mathbb{C}^-$ and $\sigma(\hat{B}, \hat{D}) \subset \mathbb{C}^-$, the Sylvester equation (12) can also be solved using the sign function method applied to

$$H - \lambda K \;:=\; \begin{bmatrix} \hat{D} & 0 \\ \hat{E} & \hat{A} \end{bmatrix} - \lambda \begin{bmatrix} \hat{B} & 0 \\ 0 & -\hat{C} \end{bmatrix}. \tag{31}$$

Using again the block-triangular structure of the matrix pencil $H - \lambda K$, the iteration can be performed on the blocks as follows:

$$\begin{array}{lll} A_0 := \hat{A}, \quad D_0 := \hat{D}, \quad E_0 := \hat{E}, & & \\ A_{k+1} := \frac{1}{2}\left(A_k + C A_k^{-1} C\right), & & \\ D_{k+1} := \frac{1}{2}\left(D_k + B D_k^{-1} D\right), & & k = 0, 1, 2, \ldots. \\ E_{k+1} := \frac{1}{2}\left(E_k + C A_k^{-1} E_k D_k^{-1} B\right), & & \end{array} \tag{32}$$

The solution of (11) is then given by the solution of the linear system of equations $2\hat{C}X\hat{B} = \lim_{k\to\infty} E_k$.

In case $\hat{C} = I_n$ and $\hat{D} = I_m$, other iterative schemes for computing the sign function like the Newton-Schulz iteration or Halley's method can also be implemented efficiently to solve the corresponding Lyapunov and Sylvester equations (14) and (12); details of the resulting algorithms will be reported in [15].

So far we have only considered the linear matrix equations for continuous-time control problems. That is, we have assumed stability with respect to the imaginary axis. In discrete-time control problems, stability properties are given with respect to the unit circle. The linear matrix equations encountered in discrete-time control problems are (13), (16), and (17). Let us first consider (13) of which (16) is a special instance. If we rewrite the equation in fixed point form, $X = \hat{A}X\hat{B} + \hat{E}$ and form the fixed point iteration

$$X_0 := \hat{E}, \qquad X_{k+1} = \hat{E} + \hat{A}X_k\hat{B}, \quad k = 0, 1, 2, \ldots.$$

then this iteration converges to X if A and B are d-stable. The convergence rate of this iteration is linear. A quadratically convergent version of the fixed point iteration is suggested in [20, 48],

$$\begin{array}{lcl} A_0 & := & \hat{A}, \quad B_0 := \hat{B}, \quad X_0 := \hat{E}, \\ X_{k+1} & := & A_k X_k B_k + X_k, \\ A_{k+1} & := & A_k^2, \quad B_{k+1} := B_k^2, \end{array} \qquad k = 0, 1, 2, \ldots. \tag{33}$$

The above iteration is referred to as the *Smith iteration*. We employ it to solve (16) and (13). In case (17) is to be solved with the Smith iteration, one has to apply (33) to $(\hat{A}\hat{C}^{-1})^T X(\hat{A}\hat{C}^{-1}) - X + \hat{C}^{-T}\hat{E}\hat{C}^{-1} = 0$. This has the disadvantage that the iteration is started with data that is already corrupted by roundoff errors basically determined by the condition of $\hat{C}$.

One possibility to avoid the initial inversion of $\hat{C}$ when solving (17) by the Smith iteration is to transform (17) to a generalized Lyapunov equation without inverting any matrices using the Cayley transformation and then applying (30) to the transformed equation

$$(\hat{A} + \hat{C})^T X(\hat{A} - \hat{C}) + (\hat{A} - \hat{C})^T X(\hat{A} + \hat{C}) + 2Q \;=\; 0 \tag{34}$$

which has the same solution as (17). Of course, the same approach can be used for (16) setting $\hat{C} = I_n$. But this yields a generalized Lyapunov equation. In order to obtain a standard Lyapunov equation of the form (14) one has to multiply (34) from the left by $(\hat{A} - \hat{C})^{-T}$ and $(\hat{A} - \hat{C})^{-1}$ from the right. This introduces again unnecessary rounding errors and we will therefore not follow this approach here.

2.3 The Disk Function Method

Let $Z - \lambda Y$, $Z, Y \in \mathbb{R}^{n\times n}$, be a regular matrix pencil having no eigenvalues on the unit circle. Suppose the *Weierstraß (Kronecker) canonical form* of $Z - \lambda Y$ is given by

$$Z - \lambda Y = T \begin{bmatrix} J^0 - \lambda I & 0 \\ 0 & J^\infty - \lambda N \end{bmatrix} S^{-1}$$

where Jordan blocks corresponding to eigenvalues inside the unit disk are collected in J^0 while J^∞ corresponds to eigenvalues outside the unit disk and N contains nilpotent blocks corresponding to infinite eigenvalues. The *matrix pencil disk function* is defined in [6] as

$$\operatorname{disk}(Z, Y) \;:=\; S\left(\begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 0 & 0 \\ 0 & I_{n-k} \end{bmatrix}\right) S^{-1} \;=:\; D_Z - \lambda D_Y. \tag{35}$$

A *matrix disc function* was also introduced in [44] using a different approach. In [6] it is shown that this is a special case of the above definition using $Y = I_n$. From the disk function, we can obtain the d–stable deflating subspace of $Z - \lambda Y$ as D_Z is a skew projector onto this subspace. Hence, a basis for this subspace is given by a basis of the column space of D_Z.

The disk function has received some interest in recent years as it provides the mathematical framework for an algorithm proposed in [39] and made feasible for practical computations in [4] for solving the spectral division problem. This *inverse free spectral division algorithm* can be given as follows:

$$\begin{array}{rcl} Z_0 & := & Z, \qquad Y_0 \;:=\; Y, \\ \begin{bmatrix} Y_k \\ -Z_k \end{bmatrix} & := & \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix} \begin{bmatrix} R_k \\ 0 \end{bmatrix} \quad \text{(QR decomposition)}, \qquad k = 0, 1, 2, \ldots. \\ Z_{k+1} & := & U_{12}^T Z_k, \qquad Y_{k+1} \;:=\; U_{22}^T Y_k, \end{array} \tag{36}$$

It follows that $\operatorname{disk}(Z, Y) = (Z_\infty + Y_\infty)^{-1}(Y_\infty - \lambda Z_\infty)$. Hence a basis for the d–stable right deflating subspace of $Z - \lambda Y$ can be computed via a rank-revealing QR decomposition of $(Z_\infty + Y_\infty)^{-1} Y_\infty$, where $\lim_{k\to\infty}(Z_k, Y_k) =: (Z_\infty, Y_\infty)$. Note that this QR decomposition can be computed without explicitly inverting $(Z_\infty + Y_\infty)$; see [4] for details. Moreover, a complete spectral decomposition of $Z - \lambda Y$ along the unit circle can be computed using only one iteration of the form (36); see [50].

From the above considerations we can conclude that the DARE (10) can be solved applying iteration (36) to $M - \lambda L$ from (24). It is shown in [6] that it is not necessary to compute a basis for the d–stable deflating subspace explicitly. Using the relation between the nullspaces $\operatorname{Ker}(D_Z) = \operatorname{Ker}(Z_\infty)$, and the fact that if the stabilizing solution X_d of (10) exists, then $[\, I_n \ (X_d \tilde{E})^T \ F_d^T \,] \in \operatorname{Ker}(D_Z)$, one can show (see [6]) that the solution of the DARE and the optimal gain matrix of the discrete-time optimal control problem can be obtained from the solution of the overdetermined but consistent set of linear equations

$$\begin{bmatrix} Z_{12} & Z_{13} \\ Z_{22} & Z_{23} \\ Z_{32} & Z_{33} \end{bmatrix} \begin{bmatrix} X_E \\ F_d \end{bmatrix} = - \begin{bmatrix} Z_{11} \\ Z_{21} \\ Z_{31} \end{bmatrix}. \tag{37}$$

Here, the Z_{kj}, $k, j = 1, 2, 3$, define a block partitioning of Z_∞ conformal to the partitioning in (24). Solving (10) with this approach will be referred to as the *disk function method*.

Note that the CARE (9) can also be solved with the disk function method by applying the iteration (36) to $C_\mu(H - \lambda K)$ with $H - \lambda K$ as in (23). The solution X_c of (9) as well as the optimal gain matrix F_c can be obtained from the overdetermined but consistent set of linear equations

$$\begin{bmatrix} Z_{12} \\ Z_{22} \end{bmatrix} X_E = \begin{bmatrix} Z_{11} \\ Z_{21} \end{bmatrix}, \qquad Z_\infty := \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix},$$

and $X_c = X_E \tilde{E}^{-1}$, $F_c = -(B^T X_E + S^T C)$.

The linear matrix equations (16) and (17) can also be solved via the disk function method applied to (25) noting that (16), (17) are special instances of (10). The corresponding matrix pencil then takes the form

$$\tilde{M} - \lambda \tilde{L} = \left[\begin{array}{cc} \hat{A} & 0 \\ \hat{E} & \hat{C}^T \end{array} \right] - \lambda \left[\begin{array}{cc} \hat{C} & 0 \\ 0 & \hat{A}^T \end{array} \right]. \tag{38}$$

Equations (14) and (15) are special instances of (9) and hence can be solved via the disk function method applied to $C_\mu(H - \lambda K)$ for $H - \lambda K$ as in (29).

Unfortunately, the iteration in (36) can not be decomposed into iterations on the matrix blocks such that no computational savings is obtained compared to the solution of the DARE. Hence, the computational cost for solving linear matrix equations with the disk function method is in general prohibitive.

More details and computational aspects of the disk function method can be found in [4, 6, 7, 50]. Though a general scaling strategy in order to accelerate convergence in (36) is yet not known, an initial scaling of $H - \lambda K$ in (23) can significantly improve the disk function methods for CAREs; see [6].

2.4 Newton's Method

The methods presented so far have addressed the algebraic Riccati equations by their relation to eigenproblems. By nature, they are systems of nonlinear equations. It is therefore straightforward to apply methods for solving nonlinear equations. In [35], Kleinman shows that Newton's method, applied to the CARE (9) with $\hat{E} = I_n$ and properly initialized, converges to the desired stabilizing solution of the CARE. The application to the generalized equation (9) is considered in [3, 40]. Given some initial guess X_0, the resulting algorithm can be stated in different ways. We have chosen here the variant that is most robust with respect to accumulation of rounding errors.

`FOR` $k = 0, 1, 2, \ldots$ "until convergence"

1. $A_k \;:=\; \tilde{A} - \tilde{G} X_k$.
2. Solve for N_k in the generalized Lyapunov equation

$$0 \;=\; \mathcal{R}_c(X_k) + A_k^T N_k \tilde{E} + \tilde{E} N_k A_k.$$

3. $X_{k+1} \;:=\; X_k + N_k$.

The main computational cost in this algorithm comes from the solution of the (generalized) Lyapunov equation in each iteration step. Convergence to X_c is globally quadratic if X_0 is chosen such that A_0 is c-stable. Finding a stabilizing X_0 usually is a difficult task and requires the stabilization of an LTI system, i.e., the solution of Task **C1**. The computational cost is equivalent to one iteration step of Newton's method; see, e.g., [47] and the references therein. Moreover, X_0 determined by a stabilization procedure may lie far from X_c. Though ultimately quadratic convergent, Newton's method may initially converge slowly. This can be due to a large error $\|X_0 - X_c\|$ or to a disastrously bad first step, leading to a large error $\|X_1 - X_c\|$; see, e.g., [34, 6, 8]. Due to the initial slow convergence, Newton's method often requires too many iterations to be competitive with other Riccati solvers. Therefore it is most frequently only used to refine an approximate CARE solution computed by any other method.

Recently an exact line search procedure was suggested that accelerates the initial convergence and avoids "bad" first steps [6, 8]. Specifically, Step 3. of Newton's method given above is modified to $X_{k+1} = X_k + t_k N_k$, where t_k is chosen in order to minimize the Frobenius norm of the residual $\mathcal{R}_c(X_k + tN_k)$. As computing the exact minimizer is very cheap compared to a Newton step and usually accelerates the initial convergence significantly while benefiting from the quadratic convergence of Newton's method close to the solution, this method becomes attractive, even as a solver for CAREs (at least in some cases), see [6, 8, 9] for details. Moreover, for some ill-conditioned CAREs, exact line search improves Newton's method also when used only for iterative refinement.

Similarly, Newton's method can be applied to the DARE (10). The resulting algorithm is described in [32] for $\tilde{E} = I_n$ and in [3, 40] for $\tilde{E} \neq I_n$. The main computation there is again the solution of a linear matrix equation which is in this case a (generalized) Stein equation. Again, line searches can be employed to (partially) overcome the difficulties mentioned above as they apply analogously to DAREs [6].

In both the continuous- and discrete-time case, in each iteration step of Newton's method, a linear matrix equation has to be solved. Hence the key to an efficient parallelization of Newton's method is an efficient solver for the linear matrix equation in question. Hence we employ the iterative schemes discussed above (sign function method or Smith iteration). Note that all other computations required by Newton's method apart from solving Lyapunov equations basically consist of matrix multiplications and can therefore be implemented efficiently on parallel computers. The parallelization of Newton's method with exact line search based upon solving the generalized Lyapunov equations via (30) is discussed in [9] where also several numerical experiments are reported.

3 Prospectus of the PLILCO

In this section we first describe the ScaLAPACK library [16] which is used as the parallel infrastructure for our PLILCO routines. We then describe the specific routines in PLILCO, including both the available routines and those that will be included in a near future to extend the functionality of the library.

3.1 The ScaLAPACK library

The ScaLAPACK (Scalable LAPACK) library [16] is designed as an extension of the successful LAPACK library [2] for parallel distributed memory multiprocessors. ScaLAPACK mimics the LAPACK, both in structure and notation. The parallel kernels in this library rely on the use of those in the PBLAS (Parallel BLAS) library and the BLACS (Basic Linear Algebra Communication Subroutines). The serial computations are performed by calls to routines from the BLAS and LAPACK libraries; the communication routines in BLACS are usually implemented on top of a standard communication library as MPI or PVM.

This structured hierarchy of dependences (see Figure 2) enhances the portability of the codes. Basically, a parallel algorithm that uses ScaLAPACK routines can be migrated to any vector processor, superscalar processor, shared memory multiprocessor, or distributed memory

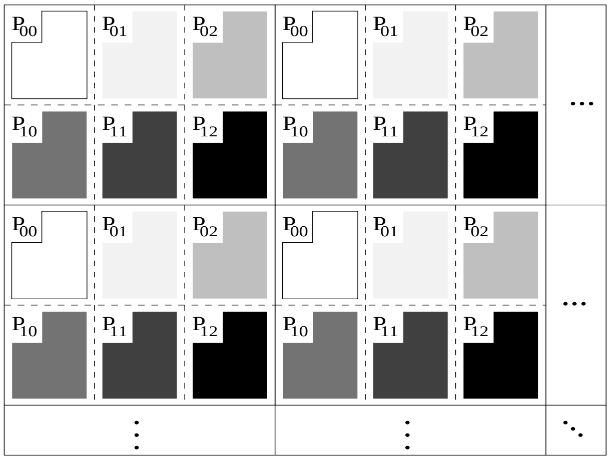

Figure 1: Data layout in a logical grid of 2×3 processes.

multicomputer where the BLAS and the MPI (or PVM) are available.

ScaLAPACK implements parallel routines for solving linear systems, linear least squares problems, eigenvalue problems, and singular value problems. The performance of these routines depends on those of the serial BLAS and the communication library (MPI or PVM).

ScaLAPACK employs the so-called message-passing paradigm. That is, the processes collaborate in solving the problem and explicit communication requests are performed whenever a process requires a datum that is not stored in its local memory.

In ScaLAPACK the computations are performed by a logical grid of $P_r \times P_c$ 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. It is the user's responsibility to perform this data distribution. Specifically, in ScaLAPACK the matrices are partitioned into $nb \times nb$ square blocks and these blocks are distributed (and stored) among the processes in column-major order. A graphical representation of the data layout is given in Figure 1 for a logical grid of 2×3 processes.

Although not strictly part of ScaLAPACK, the library also provides routines for distributing a matrix among the process grid. The communication overhead of this initial distribution is well balanced in most medium and large-scale applications by the improvements in performance achieved with parallel computation.

3.2 Structure of PLILCO

PLILCO heavily relies on the use of the available parallel infrastructure in ScaLAPACK (see Figure 2). Although ScaLAPACK is incomplete, the kernels available in the current version (1.6) allows us to implement most of our PLILCO routines. PLILCO will benefit from future extensions and developments in the ScaLAPACK project. Improvements in performance of the PBLAS kernels will also be specially welcome.

In PLILCO the routines are named, following the convention in LAPACK and ScaLAPACK,

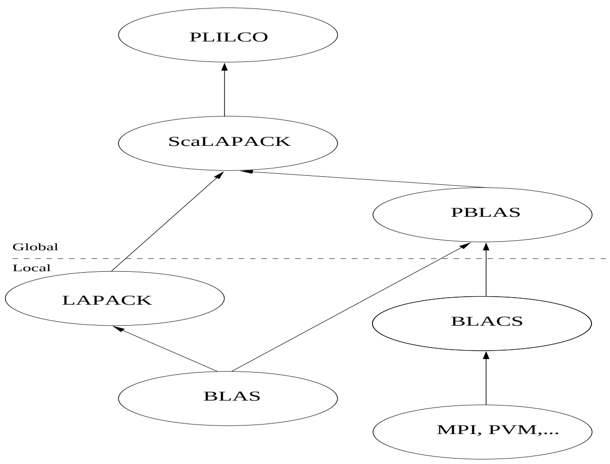

Figure 2: Structure of PLILCO.

as `PDxxyyzz`. The `PD-` prefix in each name indicates that this is a Parallel routine with Double-precision arithmetic. The following two letters, `xx` indicate the type of LTI system addressed by the routine. Thus, `GE` or `GG` indicate, respectively, a standard LTI system ($E = I_n$) or a generalized LTI system ($E \neq I_n$). The last four letters in the name indicate the specific problem (`yy`), and the method employed for that problem (`zz`). In most cases, a sequence `Cyzz` indicates that the routine deals with a continuous-time problem while a sequence `Dyzz` indicates a discrete-time problem.

The PLILCO routines can be classified in 4 groups according to their functionality (computation of basic matrix functions, linear matrix equation solvers, optimal control, and feedback stabilization). Two more groups will be included in the near future. We next review the routines in each of these 4 groups.

Group MTF: Basic matrix functions.
The routines in this group implement iterative schemes to compute functions of matrices or matrix pencils. For instance, three routines are available for computing the sign function of a matrix. These routines employ different variants for the matrix sign function iteration:

- `PDGESGNW`. The Newton iteration.
- `PDGESGNS`. The Newton-Schulz iteration.
- `PDGESGHA`. The Halley iteration.

Two more routines are designed for computing the sign function or the disk function of matrix pencils:

- `PDGGSGNW`. The generalized Newton iteration for the matrix sign function.
- `PDGGDKMA`. The iteration (36) for computing the disk function.

Type of Function	Problem	
	Standard	Generalized

Matrix sign function	`PDGESGNW`
	`PDGESGNS`
	`PDGESGHA`
		`PDGGSGNW`

Matrix pencil disk function		`PDGGDKMA`

Table 1: PLILCO routines in MTF group.

Note that the iteration (36) for the disk function only deals with matrix pencils and therefore PLILCO does not provide any routine for the standard problem. The disk function of a matrix Z is obtained by applying routine `PDGGDKMA` to $Z - \lambda I$.

Table 1 lists the PLILCO routines in the MTF group.

Group LME: Linear matrix equation solvers.
The routines in this group are solvers for several particular instances of generalized Sylvester matrix equations, see (11)–(17).

All the solvers for those equations arising in continuous-time LTI systems are based on the matrix sign function and require that the coefficient matrices of the equation are stable.

PLILCO includes three solvers for stable Sylvester equations that differ in the iteration used for the computation of the matrix sign function:

- `PDGECSNW`. The Newton iteration.
- `PDGECSNS`. The Newton-Schulz iteration.
- `PDGECSHA`. The Halley iteration.

In the generalized problem a solver for generalized Sylvester equations is also included:

- `PDGGCSNW`. The generalized Newton iteration as in (32).

All these solvers have their analogous routines for stable Lyapunov equations (three routines) and stable generalized Lyapunov equations (one routine):

- `PDGECLNW`. The Newton iteration.
- `PDGECLNS`. The Newton-Schulz iteration.
- `PDGECLHA`. The Halley iteration.
- `PDGGCLNW`. The generalized Newton iteration as in (30).

Type of equation	Problem	
	Standard	Generalized
Sylvester	PDGECSNW PDGECSNS PDGECSHA	PDGGCSNW
Lyapunov	PDGECLNW PDGECLNS PDGECLHA PDGECLNC	PDGGCLNW PDGGCLNC
Discrete-time Sylvester	PDGEDSNW	
Stein	PDGEDLSM	

Table 2: PLILCO routines in the LME group.

Furthermore, in case the constant term $\hat{E}$ is semidefinite it is also possible to obtain the Cholesky factor of the solution directly by means of routines

- `PDGECLNC`. The Newton iteration for the Cholesky factor.
- `PDGGCLNC`. The generalized Newton iteration for the Cholesky factor.

In the discrete-time case, the iterative solvers in PLILCO are based on the Smith iteration and require the coefficient matrices to be stable (in the discrete-time sense). So far PLILCO only includes two solvers, for the discrete-time Sylvester equation (13) and the Stein (or discrete-time Lyapunov) equation (16), respectively:

- `PDGEDSSM`. The Smith iteration for (13).
- `PDGEDLSM`. The Smith iteration for (16).

Special versions of the Smith iteration for computing the Cholesky factors of semidefinite Stein equations as in (20) and (21) are to be developed in the future.

The solution of generalized discrete-time linear equations can be obtained by transforming this equation into an standard one, as there is no generalized version of the Smith iteration. Note that this transformation involves explicit inversion of the coefficient matrices in the generalized equation.

Table 2 summarizes the PLILCO routines in the LME group.

Group RIC: Riccati matrix equation solvers.
We include in this group solvers both for CARE and DARE.

In the continuous-time case, PLILCO provides solvers based on three different methods, these are, Newton's method, the matrix sign function, and the matrix disk function. Moreover, in the standard case, three different variants are proposed for Newton's method depending on the Lyapunov solver that is employed. Thus, we have the following CARE solvers:

- `PDGECRNW`. Newton's method with the Newton iteration for solving the Lyapunov equations.
- `PDGECRNS`. Newton's method with the Newton-Schulz iteration for solving the Lyapunov equations.
- `PDGECRHA`. Newton's method with the Halley iteration for solving the Lyapunov equations.
- `PDGECRSG`. The matrix sign function method.
- `PDGECRDK`. The matrix disk function method.

Similarly, we have the following generalized CARE solvers:

- `PDGGCRNW`. Newton's method with the generalized Newton iterative scheme for solving the generalized Lyapunov equations.
- `PDGGCRSG`. The generalized matrix sign function method.
- `PDGGCRDK`. The matrix disk function method.

PLILCO also includes the following solvers for DARE (two routines) and generalized DARE (two routines):

- `PDGEDRSM`. Newton's method with the Smith iteration for solving the discrete-time Lyapunov equations.
- `PDGEDRDK`. The matrix disk function method for DARE.
- `PDGGDRSM`. Newton's method with the Smith iteration for solving the discrete-time generalized Lyapunov equations.
- `PDGGDRDK`. The matrix disk function method for generalized DARE.

Table 3 lists the PLILCO routines in the RIC group.

Group STF: Feedback stabilization of LTI systems.

This group includes routines for partial and complete (state feedback) stabilization of LTI systems. The routines in this group use the linear matrix equations solvers in group LME to deal with the different equations arising in standard and generalized, continuous-time and discrete-time LTI systems. PLILCO thus includes several state feedback stabilizing routines, which differ in the linear matrix equation that has to be solved and, therefore, the iteration employed. The feedback stabilization of continuous-time LTI systems can be obtained by means of routines:

- `PDGECFNW`. The Newton iteration.
- `PDGECFNS`. The Newton-Schulz iteration.

Type of Equation	Problem	
	Standard	Generalized
CARE	`PDGECRNW`	`PDGGCRNW`
	`PDGECRNS`	
	`PDGECRHA`	
	`PDGECRSG`	`PDGGCRSG`
	`PDGECRDK`	`PDGGCRDK`
DARE	`PDGEDRSM`	`PDGGDRSM`
	`PDGEDRDK`	`PDGGDRDK`

Table 3: PLILCO routines in the RIC group.

Type of LTI system	Problem	
	Standard	Generalized
continuous-time	`PDGECFNW`	`PDGGSTNW`
	`PDGECFNS`	
	`PDGECFHA`	
discrete-time	`PDGEDFSM`	

Table 4: PLILCO routines in STF group.

- `PDGECFHA`. The Halley iteration.
- `PDGGCFNW`. The generalized Newton iteration.

In the discrete-time case, the unique routine available so far is the following:

- `PDGEDFSM`. The Smith iteration.

Table 4 lists the names of the routines in group STF.

Future extensions of PLILCO will include at least two more groups:

<u>Group MRD: Model reduction of LTI systems.</u>

<u>Group H2I: Computation of H_2- and H_∞-controllers.</u>

	DGEMM (Mflops)	Latency (sec.)	Bandwith (Mbit/sec.)
IBM SP2	200	30×10^{-6}	90
Cray T3E	400	50×10^{-6}	166

Table 5: Basic performance parameters of the parallel architectures.

4 Preliminary Results

In this section we present some of the preliminary results obtained with the PLILCO routines on different parallel architectures. Specifically, we report results for the Lyapunov equation solver `PDGECLNW` and the generalized Lyapunov equation solver `PDGGCLNW`.

As target parallel distributed memory architectures we evaluate our algorithms on an IBM SP2 and a Cray T3E. In both cases we use the native BLAS, the MPI communication library, and the LAPACK, BLACS, and ScaLAPACK libraries [2, 16] to ensure the portability of the algorithms.

The IBM SP2 that we used consists of 80 RS/6000 nodes at 120 MHz, and 256 MBytes RAM per processor. Internally, the nodes are connected by a TB3 high performance switch. The Cray T3E-600 has 60 DEC Alpha EV5 nodes at 300 MHz, and 128 Mbytes RAM per processor. The communication network has a bidimensional torus topology.

Table 5 reports the performance of the Level-3 BLAS matrix product (in Mflops, or millions of floating-point arithmetic operations per second), and the latency and bandwidth for the communication system of each platform.

In both matrix equations, the coefficient matrix A is generated with random uniform entries. This matrix is stabilized by a shift of the eigenvalues ($A := A - \|A\|_F I_n$ in the continuous-time case and $A := A/\|A\|_F$ in the discrete-time case). In case a LTI system is required, $\hat{A} - \lambda \hat{E}$ is obtained from as $\hat{A} = R$, $\hat{E} = Q^H$ where Q and R are obtained from a QR factorization A. The solution matrix X is set to a matrix with all entries equal to one and matrix Q is then chosen to satisfy the corresponding linear matrix equation.

All experiments were performed using Fortran 77 and IEEE double-precision arithmetic ($\varepsilon \approx 2.2 \times 10^{-16}$). In our examples, the solution is obtained with the accuracy that could be expected from the conditioning of the problem. A more detailed study of the accuracy of these solvers is beyond the scope of this paper. For details and numerical examples demonstrating the performance and numerical reliability of the proposed equation solvers, see [9, 10, 11, 12, 14, 15].

The figures show the Mflops ratio per node when the number of nodes is increased and the ratio n/p is kept constant. Thus, we are measuring the scalability of our parallel routines. The results in the figures are averaged for 5 executions on different randomly generated matrices. In these figures the solid line indicates the maximum attainable real performance (that of `DGEMM`) and the dashed line represents the performance of the corresponding linear matrix equation solver.

Figure 3 reports the Mflops ratio per node for routine `PDGECLNW` on the Cray T3E platform and routine `PDGGCLNW` on the IBM SP2 platform.

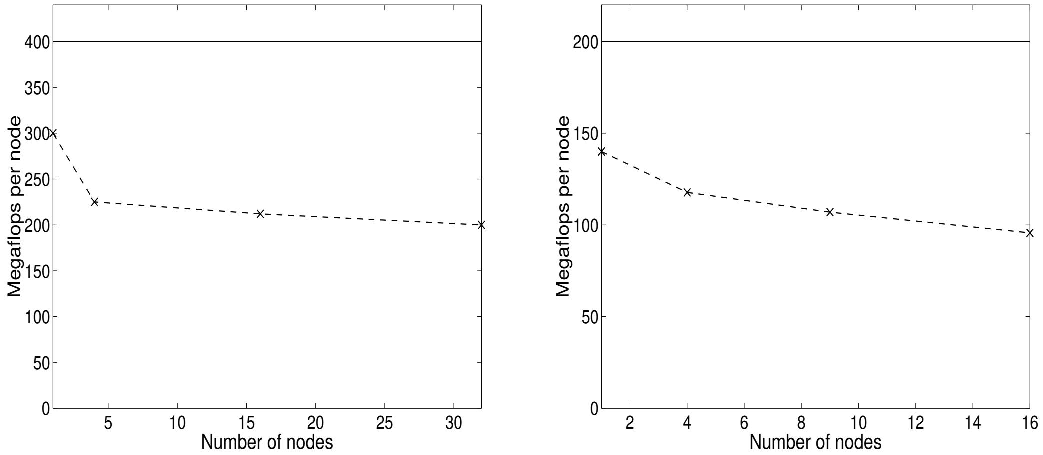

Figure 3: Mflop ratio for routine `PDGECLNW` on the Cray T3E with $n/p = 750$ (left), and routine `PDGGCLNW` on the IBM SP2, with $n/p = 1000$ (right).

Both figures show similar results. The performance per node of the algorithms decreases when the number of processors is increased from 1 to 4 due to the communication overhead of the parallel algorithm. However, as the number of processors is further increased, the performance only decreases slightly showing the scalability of the solvers.

5 Concluding Remarks

We have described the development of a software library for solving the computational problems that arise in analysis and synthesis of linear control systems. The library is intended for solving medium-size and large-scale problems and the numerical results demonstrate its performance on shared and distributed memory parallel architectures. The portability of the library is ensured by using the PBLAS, BLACS, and ScaLAPACK. It is hoped that this high-performance computing approach will enable users to deal with large scale problems in linear control theory.

We hope that the subroutine library PLILCO described in the paper will serve as a basic layer of a parallel version of the Subroutine Library in Systems and Control Theory, SLICOT [13], developed within the *Numerics in Control Network* NICONET and proposed in [17].

Acknowledgments

The work reported in this paper was also partially supported by the DAAD programme *Acciones Integradas Hispano-Alemanas*. Enrique S. Quintana-Ortí and Gregorio Quintana-Ortí were also supported by the Spanish CICYT Project TIC96-1062-C03-03.

References

[1] B.D.O. Anderson and J.B. Moore. *Optimal Control – Linear Quadratic Methods.* Prentice-Hall, Englewood Cliffs, NJ, 1990.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide.* SIAM, Philadelphia, PA, second edition, 1995.

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

[4] 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.

[5] R.H. Bartels and G.W. Stewart. Solution of the matrix equation $AX + XB = C$: Algorithm 432. *Comm. ACM*, 15:820–826, 1972.

[6] 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.

[7] 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.

[8] 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.

[9] 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. Berichte aus der Technomathematik, Report 98–05, FB3 – Mathematik und Informatik, Universität Bremen, 28334 Bremen (Germany), August 1998. Available from `http://www.math.uni-bremen.de/zetem/berichte.html`.

[10] P. Benner, M. Castillo, E.S. Quintana-Ortí, and V. Hernández. Parallel partial stabilizing algorithms for large linear control systems. *J. Supercomputing*, to appear.

[11] P. Benner, J.M. Claver, and E.S. Quintana-Ortí. Efficient solution of coupled Lyapunov equations via matrix sign function iteration. In A. Dourado et al., editor, *Proc.* 3rd *Portuguese Conf. on Automatic Control CONTROLO'98,* Coimbra, pages 205–210, 1998.

[12] P. Benner, J.M. Claver, and E.S. Quintana-Ortí. Parallel distributed solvers for large stable generalized Lyapunov equations. *Parallel Processing Letters*, to appear.

[13] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga. SLICOT - a subroutine library in systems and control theory. *Applied and Computational Control, Signals, and Circuits*, to appear. *See also:* NICONET Report 97–03, available from `http://www.win.tue.nl/niconet/NIC2/reports.html`.

[14] P. Benner and E.S. Quintana-Ortí. Solving stable generalized Lyapunov equations with the matrix sign function. *Numerical Algorithms*, to appear.

[15] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Solving linear matrix equations via rational iterative schemes. In preparation.

[16] 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.

[17] I. Blanquer, D. Guerrero, V. Hernandez, E. Quintana-Ortí, and P. Ruiz. Parallel-SLICOT implementation and documentation standards. SLICOT Working Note 1998–1, `http://www.win.tue.nl/niconet/`, September 1998.

[18] D. Boley and R. Maier. A parallel QR algorithm for the unsymmetric eigenvalue problem. Technical Report TR-88-12, Univ. of Minn. at Minneapolis, Dept. of Computer Science, 1988.

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

[20] E.J. Davison and F.T. Man. The numerical solution of $A'Q + QA = -C$. *IEEE Trans. Automat. Control*, AC-13:448–449, 1968.

[21] 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.

[22] J.D. Gardiner and A.J. Laub. Parallel algorithms for algebraic Riccati equations. *Internat. J. Control*, 54:1317–1333, 1991.

[23] J.D. Gardiner, A.J. Laub, J.J. Amato, and C.B. Moler. Solution of the Sylvester matrix equation $AXB + CXD = E$. *ACM Trans. Math. Software*, 18:223–231, 1992.

[24] J.D. Gardiner, M.R. Wette, A.J. Laub, J.J. Amato, and C.B. Moler. Algorithm 705: A Fortran-77 software package for solving the Sylvester matrix equation $AXB^T + CXD^T = E$. *ACM Trans. Math. Software*, 18:232–238, 1992.

[25] G. H. Golub, S. Nash, and C. F. Van Loan. A Hessenberg–Schur method for the problem $AX + XB = C$. *IEEE Trans. Automat. Control*, AC-24:909–913, 1979.

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

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

[28] S. J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. Numer. Anal.*, 2:303–323, 1982.

[29] G. A. Geist, R. C. Ward, G. J. Davis, R. E. Funderlic. Finding eigenvalues and eigenvectors of unsymmetric matrices using a Hypercube multiprocessor. In G. Fox, editor, *Proc. 3th Conference on Hypercube Concurrent Computers and Appl.*,, pages 1577–1582, 1988.

[30] G. Henry and R. van de Geijn. Parallelizing the QR algorithm for the unsymmetric algebraic eigenvalue problem: Myths and reality. *SIAM J. Sci. Comp.*, 17:870–883, 1997.

[31] G. Henry, D. Watkins, and J. Dongarra. A parallel implementation of the nonsymmetric QR algorithm for distributed memory architectures. Technical Report CS-97-352, University of Tenneesee at Knoxville, 1997.

[32] G.A. Hewer. An iterative technique for the computation of steady state gains for the discrete optimal regulator. *IEEE Trans. Automat. Control*, AC-16:382–384, 1971.

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

[34] C. Kenney, A.J. Laub, and M. Wette. A stability-enhancing scaling procedure for Schur-Riccati solvers. *Sys. Control Lett.*, 12:241–250, 1989.

[35] D. L. Kleinman. On an iterative technique for Riccati equation computations. *IEEE Trans. Automat. Control*, AC-13:114–115, 1968.

[36] P. Lancaster and L. Rodman. *The Algebraic Riccati Equation.* Oxford University Press, Oxford, 1995.

[37] A.J. Laub. A Schur method for solving algebraic Riccati equations. *IEEE Trans. Automat. Control*, AC-24:913–921, 1979.

[38] A.J. Laub. Algebraic aspects of generalized eigenvalue problems for solving Riccati equations. In C.I. Byrnes and A. Lindquist, editors, *Computational and Combinatorial Methods in Systems Theory*, pages 213–227. Elsevier (North-Holland), 1986.

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

[40] 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.

[41] V. Mehrmann. A step toward a unified treatment of continuous and discrete time control problems. *Linear Algebra Appl.*, 241–243:749–779, 1996.

[42] T. Pappas, A.J. Laub, and N.R. Sandell. On the numerical solution of the discrete-time algebraic Riccati equation. *IEEE Trans. Automat. Control*, AC-25:631–641, 1980.

[43] T. Penzl. Numerical solution of generalized Lyapunov equations. *Adv. Comp. Math.*, 8:33–48, 1997.

[44] 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).

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

[46] G. Schelfhout. *Model Reduction for Control Design*. PhD thesis, Dept. Electrical Engineering, KU Leuven, 3001 Leuven–Heverlee, Belgium, 1996.

[47] V. Sima. *Algorithms for Linear-Quadratic Optimization*, volume 200 of *Pure and Applied Mathematics*. Marcel Dekker, Inc., New York, NY, 1996.

[48] R.A. Smith. Matrix equation $XA + BX = C$. *SIAM J. Appl. Math.*, 16(1):198–201, 1968.

[49] G. W. Stewart. A parallel implementation of the QR algorithm. *Parallel Computing*, 5:187–196, 1987.

[50] X. Sun and E.S. Quintana-Ortí. Spectral division methods for block generalized Schur decompositions. PRISM Working Note #32, 1996. Available from `http://www-c.mcs.anl.gov/Projects/PRISM`.

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

[52] A. Varga. A note on Hammarling's algorithm for the discrete Lyapunov equation. *Sys. Control Lett.*, 15(3):273–275, 1990.

[53] A. Varga. Computation of Kronecker-like forms of a system pencil: Applications, algorithms and software. In *Proc. CACSD'96 Symposium, Dearborn, MI*, pages 77–82, 1996.

[54] K. Zhou, J.C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice-Hall, Upper Saddle River, NJ, 1996.