Task I.A.1 - Selection of Basic Software Tools for Standard and Generalized State-space Systems and Transfer Matrix Factorizations [1]

Andras Varga[2]

June 1998

[1]This document 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/pub/WGS/REPORTS/SLWN1998-3.ps.Z`

[2]Deutsches Zentrum für Luft und Raumfahrt, Institut für Robotik und Systemdynamik, DLR Oberpfaffenhofen, Postfach 1116, D-82230 Wessling, Germany

Contents

1 Introduction

The current status of the SLICOT library covers already a large number of basic mathematical and system theoretic computations. To guarantee a proper distribution of the SLICOT library, the product has been made freely available. Recall that this activity got the highest priority in the results of a previous NICONET questionnaire. During the *Exploratory Phase* we set up an ftp site at KUL-SISTA and we adapted the SLICOT implementation and documentation standards[1]. In the period between the *Exploratory* and *Implementation Phase*, most of the 90 SLICOT routines of Release 2.0 have been upgraded to the new standards and have been made available on ftp. However, to match better the recent system theoretic advances of the last decade, new basic mathematical tools have to be added to the library. Special emphasis will go here towards basic linear algebra techniques for (i) generalized state space models and factorizations of transfer functions, and (ii) fast numerical procedures for structured matrices and perturbations. In the present working notes we only address the first of the above issues.

As Release 3.0 is basically a conversion of Release 2.0, it still lacks several useful routines of potentially large interest to users. Therefore, first we discuss some useful completions to the existing SLICOT chapters on the basis of an overview of existing routines and their functionality. Then we discuss the development of new basic software for factorization of transfer matrices as well for generalized state-space (or descriptor) systems. We address only the solution of computational problems for which reliable numerical algorithms exist and for most of which, reliable numerical software, suitable for standardization within SLICOT, is also available (either as FORTRAN programs or as MATLAB prototype functions).

The standardization of basic numerical tools will include three distinct activities. The **first** activity consists in the standardization of routines which have been already submitted for Release 2.0 of SLICOT by various contributors (for pole assignment, Lyapunov equations, Riccati equations, system norms, etc.). Because of lack of financial support, these routines have been not included in Release 3.0, although they were since long time on the lists of possible extensions.

The **second** standardization direction is of special interest in model reduction as well as in H_∞-control and consist developing a set of routines for the factorization of transfer-function matrices. The basis for this activity forms the routines available in the RASP-MODRED library [48]. Note that, the implementations of routines in RASP-MODRED are based on the linear algebra standard package LAPACK [3] and most of routines fulfill the RASP-SLICOT mutual compatibility concept established in [18]. Therefore, all routines available in RASP-MODRED are well suited for standardization in SLICOT.

The **third** activity involves the standardization of routines for descriptor systems. For this purpose, we will use mainly the routines available in the RASP-DESCRIPT library [46] (which contains routines adapted mainly from the BIMASC library [62]). Unfortunately, the routines from RASP-DESCRIPT requires a large standardization effort, because they are written according to an Implementation standard particular to RASP [17] and rely on EISPACK and LINPACK calls. Alternatively, prototype software available in MATLAB can be used as basis for standardization, although the effort starting from these codes is even higher than starting from existing Fortran implementations.

[1] report available via anonymous ftp from wgs.esat.kuleuven.ac.be/pub/WGS/reports/

2 Algorithms and Software for Basic Control Problems

Consider the linear state-space system G

$$\begin{array}{rcl} \lambda x(t) & = & Ax(t) + Bu(t) \\ y(t) & = & Cx(t) + Du(t), \end{array} \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$, and where λ is either the differential operator d/dt for a continuous-time system or the advance operator z for a discrete-time system. The system (1) will be alternatively referred to as the quadruple $G = (A, B, C, D)$ or as the triple $G = (A, B, C)$ if $D = 0$ or D is not important for the context. The *transfer function matrix* (TFM) of system (1) is the $p \times m$ proper rational matrix

$$G(\lambda) = C(\lambda I - A)^{-1}B + D. \tag{2}$$

2.1 State-Space Transformations

For an invertible matrix T, two state-space systems (A, B, C, D) and $(\widetilde{A}, \widetilde{B}, \widetilde{C}, \widetilde{D})$ related by

$$\widetilde{A} = T^{-1}AT, \quad \widetilde{B} = T^{-1}B, \quad \widetilde{C} = CT, \quad \widetilde{D} = D, \tag{3}$$

are called *similar* and the transformation (3) is called a *similarity transformation.* Note that similar state-space systems have the same TFM.

The similarity transformations are the basic preprocessing tools for most of analysis, model conversion and synthesis problems discussed in this chapter. From numerical point of view, it is important that the transformation matrix T to be a well-conditioned (ideally an orthogonal) matrix. Many useful reductions of the system matrices to special condensed forms (see later) can be done by using exclusively orthogonal transformations.

2.1.1 State-space scaling

Given a system $G = (A, B, C)$, we can try to compute a transformation matrix T in (3)to reduce the 1-norm of the transformed system matrix

$$\widetilde{\mathcal{S}}(\lambda) = \left[\begin{array}{c|c} \widetilde{A} & \widetilde{B} \\ \hline \widetilde{C} & 0 \end{array}\right].$$

Such a transformation is frequently necessary to improve the accuracy of subsequent numerical computations involving the system matrices. Note that the scaling can be equally performed only on the pairs (A, B) or (A, C) or even on a single square matrix A.

To perform such a scaling, the BALABC routine is available in RASP and is suitable for standardization purposes. **Important note.** It is generally advisable to include a scaling option in all analysis and design routines to overcome possible numerical problems caused by poor scaling of the original state-space representations. The implementation of BALABC originates from the BALANC subroutine of EISPACK, with straightforward extensions to handle upper triangular and nonsquare matrices.

2.1.2 Block-diagonal form of a matrix

Consider the similarity transformation (3) on the system $G = (A, B, C)$, such that the resulting state matrix $\widetilde{A}$ is in a *block-diagonal form* (BDF)

$$\widetilde{A} = T^{-1}AT = \mathrm{diag}(A_1, \ldots, A_k) \tag{4}$$

with the matrices $\widetilde{B}$ and $\widetilde{C}$ partitioned accordingly

$$\widetilde{B} = T^{-1}B = [\, B_1^T, \ldots, B_k^T \,]^T, \qquad \widetilde{C} = CT = [\, C_1, \ldots, C_k \,]. \tag{5}$$

This partition of system matrices is equivalent with the additive decomposition $G = \sum_{i=1}^{k} G_i$, where $G_i(\lambda) = C_i(\lambda I - A_i)^{-1}B_i$, for $i = 1, \ldots, k$. Such an additive decomposition of G is useful for many control computations, as for instance [49]: discretization of continuous-time systems, computation of frequency responses, modal approach to model reduction [54], evaluation of the transfer-function matrix of large scale systems.

To compute the BDF of a matrix, the BDIAG routine is available in RASP and is suitable for standardization purposes. This routine is based on an efficient algorithm proposed in [5]. Besides simplicity and computational efficiency, the main advantage of this algorithm is its ability to keep under control the condition number of the transformation matrix. This allows the computation of the BDF with a prescribed accuracy loss and thus represents a major numerical enhancement over the diagonalization algorithms based on eigenvector computations. BDIAG calls a special purpose Sylvester equation solver SYLSM which stops computations when the computed solution exceeds a given magnitude. A possible improvement of BDIAG could be to determine the blocks such that any two blocks have no common eigenvalues. This feature is important for the modal approach for model reduction.

The transformation computed by BDIAG can be applied by calling the SIMEQ routine from RASP. This routine basically performs a coordinate transformation as in (3) or its inverse. It is suited for standardization for SLICOT.

2.1.3 Additive decomposition of transfer matrices

In several applications, like model reduction of unstable systems or computation of the Hankel-norm of an unstable system it is necessary to use a stable/unstable additive spectral decomposition of a transfer matrix G as $G = G_1 + G_2$, where G_1 and G_2 are determined such that G_1 has only poles in the stable region and G_2 has exclusively poles outside that region.

For $G = (A, B, C)$ the main computation involves the determination of a transformation matrix T such that the transformed system has the form

$$\left[\begin{array}{c|c} T^{-1}AT & T^{-1}B \\ \hline CT & 0 \end{array}\right] := \left[\begin{array}{cc|c} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ \hline C_1 & C_2 & 0 \end{array}\right],$$

where A_1 and A_2 contain the systems poles lying in the stable and unstable regions, respectively. The additive terms are then defined by $G_1 := (A_1, B_1, C_1)$ and $G_2 := (A_2, B_2, C_2)$. The subroutine SADSDC in RASP-MODRED can be used to perform the above decomposition by reducing the system state-matrix A to a block-diagonal form. This routine even allows to consider more general stability domains for both continuous- and discrete-time systems. SADSDC calls another RASP routine SRSFOD which performs the stable and anti-stable or fast and slow modes separations of a system on the basis of the ordered Schur form of the state matrix A. On his turn, SRSFOD calls the SEOR1 routine to perform the various reorderings of the eigenvalues in a RSF matrix. The computation of additive decompositions is based on an algorithm explained in [35].

2.1.4 Orthogonal reduction of system matrices

Given the system $G = (A, B, C)$, the RASP subroutine SRSFDC performs the orthogonal state-space coordinate transformation

$$\widetilde{A} = Q^T AQ, \quad \widetilde{B} = Q^T B, \quad \widetilde{C} = CQ, \tag{6}$$

where Q is an orthogonal matrix determined to reduce A to a real Schur form $\widetilde{A}$. Such a reduction is frequently necessary as a preprocessing step in the routines for balancing related model reduction.

Another useful orthogonal similarity transformation is to reduce the pair (A, B) to the controllability staircase form

$$\widetilde{A} = Q^T AQ = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad \widetilde{B} = Q^T B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \quad \widetilde{C} = CQ,$$

where the pair (A_{11}, B_1) is controllable and the matrix $[\, B_1 \ A_{11} \,]$ is in a staircase form. Such a transformation, with optional accumulation of the transformation matrix Q, is useful in an efficient implementation of the minimal realization routine AB06MD. Note that the present version is based on using AB01ND which handles only the pair (A, B). The matrix C is not involved and thus the transformation Q must be always accumulated. An observability form equivalent of the above routine with the resulting $\widetilde{A}$in an upper block Hessenberg form would be also desirable.

The two orthogonal transformation routines TB01MD and TB01ND must be correspondingly updated to perform the transformations also on the output matrix C.

2.2 Continuous-to-Discrete Conversion

2.2.1 Computation of matrix exponentials

The BDF (4) is useful to compute the exponential of a matrix A using the simple formula

$$\exp(A) = T \exp(\widetilde{A}) T^{-1} = T \mathrm{diag}(\exp(A_1), \ldots, \exp(A_k)) T,$$

where the exponentials of diagonal blocks are evaluated by Padé approximation. This approach has been proposed in [28] and apparently is one of the best available method. Note that by using BDIAG to compute the BDF (4), the resulting blocks are in *real Schur form* (RSF) and thus a specialized version of the SLICOT routine MB05OD can be used for this purpose. For standardization purpose an implementation available in the BIMAS library, BPADE, can be also used [66]. The new SLICOT routine to compute matrix exponential is intended to supersede the MB05MD routine in SLICOT which can handle only real non-defective matrices.

2.2.2 Discretization of continuous-time models

The sampled-data system $G_d = (A_d, B_d, C)$ resulting from the discretization of a continuous-time system $G = (A, B, C)$ with a sampling period τ can be computed from the matrix identity

$$\exp\left(\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}\tau\right) = \begin{bmatrix} A_d & B_d \\ 0 & 0 \end{bmatrix}.$$

If G has been additively decomposed according to the BDF of A in (4), then the discretization can be performed at the level of the low order subsystems $G_i = (A_i, B_i, C_i)$. The corresponding sampled-data subsystem is $G_{id} = (A_{id}, B_{id}, C_i)$, where the pair (A_{id}, B_{id}) results from

$$\exp\left(\begin{bmatrix} A_i & B_i \\ 0 & 0 \end{bmatrix}\tau\right) = \begin{bmatrix} A_{id} & B_{id} \\ 0 & 0 \end{bmatrix}.$$

Then the matrices A_d and B_d of the sampled data system G_d can be assembled as

$$\begin{aligned} A_d &= \operatorname{diag}(A_{1d}, \ldots, A_{kd}) \\ B_d &= \operatorname{row}(B_{1d}, \ldots, B_{kd}). \end{aligned}$$

The above approach can serve to implement a powerful routine to be included in SLICOT to discretize continuous-time systems.

2.3 System Conversions

2.3.1 Evaluation of the transfer function matrix

Given a state-space model $G = (A, B, C, D)$ it is often necessary to determine the corresponding TFM (2). Among several algorithms for this computation, the most reliable one seems to be the poles-zeros method of Varga and Sima [65]. This algorithm determines each element of $G(\lambda)$ as aproduct of a scalar gain with a monic polynomial for the corresponding zeros divided by a monic polynomial for the corresponding poles. Using already available software in SLICOT, it is possible to standardize the TSMT routine from BIMASC to include it in SLICOT. The standardization involves besides adapting of interfaces, also replacing of calls to EISPACK routines by equivalent calls to LAPACK routines. Prototype software is also available in the MATLAB-toolbox HTOOLS [63].

2.3.2 Minimal state-space representation for a proper transfer matrix

Presently there is no software in SLICOT available to compute a minimal state-space realization (A, B, C, D) of a proper transfer function matrix $G(\lambda)$ satisfying (2). The existing SLICOT routine TD01OD, requires $G(\lambda)$ to be provided in a factorized form $G(\lambda) = M(\lambda)^{-1}N(\lambda)$ or

$G(\lambda) = N(\lambda)M(\lambda)^{-1}$, where $N(\lambda)$ and $M(\lambda)$ are polynomial matrices with $M(\lambda)$ diagonal. There is no routine available in SLICOT to convert an arbitrary proper rational matrix to this form, although this is a straightforward operation provided routines for the greatest common divisor (g.c.d.) and least common multiple (l.c.m.) of two polynomials are available.

However, there exists in BIMASC [62] three routines, RENEMC, RENEMO, and RENEM which compute respectively, a controllable, an observable or a generally non-minimal state-space realization of a given proper rational matrix. RENEMC and RENEMO rely on a g.c.d. routine, but RENEM generates the system matrices without any preprocessing. The resulting order of the non-minimal realization is usually higher than that resulted with RENEMC or RENEMO. It is possible to devise a minimal realization routine for SLICOT on the basis of these routines in combination with the minimal realization routine AB06MD or a balancing related exact model reduction procedure.

2.4 Frequency Responses

2.4.1 Computation of frequency responses

For a large order system it is possible to evaluate cheaply the frequency response $G(j\omega)$ or $G(e^{j\omega T})$ for many values of the frequency ω by using the additive decomposition (5) resulted from the BDF of the state matrix (4) [49]. For example, for a given frequency value ω, $G(j\omega)$ can be computed as

$$G(j\omega) = \sum_{i=1}^{k} G_i(j\omega), \tag{7}$$

where

$$G_i(j\omega) = C_i(j\omega I - A_i)^{-1}B_i. \tag{8}$$

Recall that A_i is already in a RSF and thus the evaluation of $G_i(j\omega)$ is computationally very cheap because of usually very low order of A_i. A subroutine SMBDFR is available for this purpose in RASP and is well suited for standardization.

2.4.2 Computation of Bode diagrams with jump-free phases

The frequency response computed by the SLICOT routines TB01RD, TC01MD, TD01MD represent the value of the transfer function matrix $G(\lambda)$ evaluated for $\lambda = j\omega$ for a continuous-time system or $\lambda = \exp(j\omega\tau)$ for a discrete-time system. This value is computed in two matrices representing the real and imaginary parts of $G(\lambda)$. The extraction of the corresponding phase information (necessary for Bode and Nichol diagrams) is done by using the **arctan2** function (in Fortran or MATLAB). This functions returns phase values lying always between $-\pi$ and π, although the system phase has frequently values outside of this interval. This leads to possible 2π jumps in the phase values, although the phase variation is continuous. To obtain nice plots, the phase continuity must be ensured by using extrapolation techniques combined with adaptive selection of frequency points. A routine which can be used to compute continuous (magnitude, phase) values of the frequency response is RPBOCD available in RASP. The standardization of this rather simple routine could rise some problems. Alternatively, the corresponding MATLAB function `bode` from MATLAB could serve as prototype software for implementation.

2.5 Controller Synthesis by Pole Assignment

We consider the following *eigenvalue assignment problem* (EAP): given the controllable matrix pair (A, B), where $A \in \mathbb{R}^{n,n}$ and $B \in \mathbb{R}^{n,m}$, determine the feedback matrix $F \in \mathbb{R}^{m,n}$ such that the closed-loop state matrix $A+BF$ has all its eigenvalues at desired locations $\Gamma = \{\lambda_1, \ldots, \lambda_n\}$ in the complex plane. We assume that Γ is symmetric with respect to the real axis. This assumption guarantees that the resulting F is real. There exist several numerically stable algorithms which can be used to solve the EAP [30, 34, 26]. All these methods are based on the orthogonal controllability staircase form of the pair (A, B) [40]. An alternative to these methods is the so-called *Schur method* proposed by Varga [42] which uses the *real Schur form* (RSF) of the matrix to accomplish the eigenvalue assignment. Although computationally more involved than the previous ones, the Schur method has the attractive feature to allow a partial pole assignment, *i.e.* it is possible to alter only those eigenvalues of A which are unsatisfactory for the closed-loop system dynamics and to keep unmodified the rest of eigenvalues. The Schur approach has been extended to generalized state-space systems [55] as well as to periodic systems [38]. Moreover the Schur method has been adapted to compute various coprime factorizations of rational matrices [51, 52]. Note that the solution of the EAP can be generally expressed in terms of $n(m-1)$ free parameters [33], thus in the multi-input case ($m > 1$) this freedom can be exploited to fulfill additional requirements.

The following FORTRAN software is available and can serve for standardization within SLICOT:

SB01BD	Multi-input pole assignment using the Schur method [42]
MEVAS	Multi-input pole assignment using the Hessenberg method [26]
POLCM	Parametric multi-input pole assignment using the Hessenberg method [33]

SB01BD and POLCM are based on LAPACK calls and thus are appropriate for standardization. However, the MEVAS code, although appeared in the TOMS collection has serious interface design deficiencies, and thus its standardization is much more involved. New algorithmic developments not supported by software are the multishift variant of the single-input EAP [58] and the parametric variant of the Schur method [59].

2.6 Riccati Solvers and Related Computations

2.6.1 Newton's method for iterative solving and refinement

The Newton's method with *exact line search* to solve the *continuous-time algebraic Riccati equation* (CARE)

$$Q + A^T X + XA - XBR^{-1}B^T X = 0 \tag{9}$$

has been proposed in [6, 8]. A routine with line search is available in [10], but needs to be standardized if included in SLICOT. A MATLAB version is also available. The routine NEWTON, which implements the *standard* Newton's method (without line search), was submitted by Byers and Barth some years ago and was intended for Release 3.0 of SLICOT.

The standard Newton's method to solve the *discrete-time algebraic Riccati equation* (DARE)

$$X = A^T XA - A^T XB(R + B^T XB)^{-1} B^T XA + Q \tag{10}$$

is the **only** reasonable method which can be really trusted and needs definitely to be implemented for SLICOT. The Newton's method can also be endowed with line search as proposed in [6]. The problem with line search in the discrete-time case is that theoretical basis has still some "holes". A MATLAB version is available and could serve as basis for a Fortran implementation.

The following routines to solve CAREs and DAREs by using the *standard* Newton method are available in BIMAS [66]:

STAC	state-feedback stabilization of linear continuous-time systems [41] (necessary to initialize NTNC)
STAD	state-feedback stabilization of linear discrete-time systems [41] (necessary to initialize NTND)
NTNC	Newton's method to solve the CARE [36]
NTND	Newton's method to solve the DARE [36]

Since the standard Newton's method is just a special case of the Newton's method with line search (where the "step size" is taken to be one), it is reasonable to implement a single routine to solve both CAREs and DAREs, optionally with or without line search. Note that at a certain stage, the line search algorithms will always do only standard Newton steps. In particular this will be true almost always if the Newton's method is used for iterative refinement. It can usually be proved that the step size converges to one (and hence line search becomes Newton's method) if the method converges at all.

2.6.2 Eigenvalues of Hamiltonian matrices and symplectic matrices/pencils

The computation of eigenvalues of Hamiltonian matrices and symplectic matrices/pencils has important applications, as for instance, in computing infinity norms of transfer matrices or in recursive methods to compute inner-outer factorizations. The following software is available to compute the eigenvalues of Hamiltonian matrices:

DHASRD	Van Loan's square reduction of Hamiltonian matrices, implicit version (submitted to TOMS)
DSHSRD	Van Loan's square reduction of skew-Hamiltonian matrices (almost ready — part of the TOMS submission)
DHAEVS	eigenvalues of Hamiltonian with Van Loan's method (explicit or implicit), (almost ready, part of TOMS submission) Includes balancing as described in [9]
DHAURV	symplectic URV–type decomposition of Hamiltonian matrix [13] (ready)
DHAEVU	eigenvalues of Hamiltonian matrix with algorithm described in [13] (ready, needs Varga's periodic QR implementation)
DHASBL	symplectic balancing as described in [6, 7] (ready by April 1)
DHAEV	driver routine for eigenvalues of Hamiltonian matrices, will be part of 2nd part of TOMS submission together with DHASBL, DHAURV, DHAEVU

For the computation of eigenvalues of symplectic matrices/pencils no software is presently available. The use of QR algorithm on the symplectic matrix or of the QZ algorithm on the symplectic pencil, although straightforward, requires delicate decisions to check if some eigenvalues are on the unit circle and therefore can not be trusted. An implementation project is the routine DSMEVV to compute the eigenvalues of a symplectic matrix by using the $S + S^{-1}$ trick and DSHSRD. This implementation may however suffer from $\sqrt{\varepsilon}$ loss of accuracy, but it still relies on a safe decision if eigenvalues are on the unit circle (expected to be ready by April 1). Alternatively the $S + S^{-1}$ trick for symplectic pencils [22, 31] might be also implemented or even the recently proposed method based on [13] may come into discussion (the theory is not ready yet). We leave these issues open for discussions.

2.6.3 Symplectic methods to solve Riccati equations

Structure preserving algorithms to solve Riccati based on the reduction of Hamiltonian matrices by using orthogonal symplectic transformations has been proposed in [2, 1] and an implementation of this method led to the OSMARE subroutine submitted to SLICOT in October 1994 and also used in [37]. A new version of OSMARE will be ready soon and will employ symplectic balancing, the best you can do for relatively modest dimensions ($n < 50$). A new solver based on new method [12] is available in a prototype MATLAB implementation. The corresponding Fortran 77 implementation is rather involved, and can not guaranteed to be ready soon. Note that this version seems to be the only reasonable method for large dimensions ($50 < n < 1000$).

The choice of symplectic methods for the discrete-time case needs to be discussed, because all known methods suffer form severe problems: either due to use of the Newton's method with line search, or of the QZ method on symplectic pencil (can be implemented efficiently using part of the structure) together with Newton for iterative refinement. (Note: a comparison done in [6] shows that Newton's method with line search outperforms the QZ method even without iterative refinement for *all* benchmark examples of [11].)

2.7 Computation of Transfer Matrix Norms

To evaluate the model reduction approximation errors, different norms of TFMs are necessary to be computed. The following routines are provided in the RASP package for this purpose:

SHANRM	computes the Hankel norm and the Hankel singular values of the stable projection of a transfer-function matrix
SL2NRM	computes the L_2- or l_2-norm of a transfer-function matrix [47]
RPHINR	computes the H_∞ norm of a stable, continuous-time transfer-function matrix

SHANRM and SL2NRM are based on LAPACK and SLICOT Release 2.0 routines and thus are well suited for standardization purposes. To compute the norms of unstable systems, routines for spectral decomposition or coprime factorization with inner denominator are called. The RPHINR routine could also serve for standardization purpose, although this would imply complete rewriting of it and substantial algorithmic improvements using the recently developed routines to compute the eigenvalues of Hamiltonian matrices. Note that in this context a routine to compute the eigenvalues of symplectic matrices/ pencils would be very helpful to compute the H_∞ norm in the discrete-time case. Alternatively, the bilinear transformation technique can be used to perform discrete-to-continuous and continuous-to-discrete conversions.

2.8 Observer Synthesis

There is no available routine in SLICOT to support observer design for linear state-space systems. Observer design for linear state estimation can be done either in form of a Kalman estimator or of a minimal order observer. The design of Kalman estimators involves the solution of appropriate filter Riccati equations (which are the duals of the Riccati equations used for the design of controllers using linear-quadratic optimization). Thus the design of Kalman observers can be done with the same programs as the design of optimal LQ controllers.

For the design of minimal order observer, usually pole assignment methods are used. An efficient method for this purpose has been proposed in [43] and relies on the methodology of Luenberger [23] combined with the Schur method for pole assignment [42]. One nice aspect of the designed observer is that the resulting observer state-matrix is in RSF. The subroutine SB07AD (a version of the SAESTM routine of BIMASC [62]) has been adapted to be included in SLICOT Release 2.0 . This routine can serve for standardization purposes for SLICOT. For discrete-time systems observers based on dead-beat control techniques could be also helpful.

2.9 Optimal Output Feedback

For the control of the linear system (1) an optimal output feedback control law

$$u(t) = Fy(t) \tag{11}$$

can be computed which minimizes the quadratic performance index

$$J = E\left\{\int_0^\infty [\,x(t)^T Qx(t) + u(t)^T Ru(t)\,]dt\right\} \tag{12}$$

in the continuous-time case, or

$$J = E\left\{\sum_{k=0}^\infty [\,x(t)^T Qx(t) + u(t)^T Ru(t)\,]\right\} \tag{13}$$

in the discrete-time case, where Q and R are symmetric matrices with $Q \geq 0$ and $R > 0$.

For the solution of this problem in general no closed form solutions exist. Thus iterative search methods must be used to compute the optimizing output feedback matrix F. For search methods based on gradient techniques it is necessary to evaluate for a given stabilizing output feedback F the corresponding values of the cost functional (19) and of its gradient with respect to F. The computations involve the solution of two Lyapunov equations.

Software for the computation of the optimal output feedback is available in RASP. The subroutine FUNGRD calculates the function value $J(F)$ and the value of the gradient $\nabla_F J(F)$ for a given feedback F. The subroutine PARLQR computes the optimal output feedback gain F which minimize the performance function $J(F)$. The computational approach and derivation of formulas for function and gradient evaluations are presented in [57], where a multi-model version of the method is also described. Both above routines are well suited for standardization within SLICOT, provided the MINPACK-2 minimization routines are freely available.

2.10 Periodic Systems

Consider the linear discrete-time periodic system of the form

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k \\ y_k &= C_k x_k + D_k u_k \end{aligned} \tag{14}$$

where the matrices $A_k \in \mathbb{R}^{n\times n}$, $B_k \in \mathbb{R}^{n\times m}$, $C_k \in \mathbb{R}^{p\times n}$ and $D_k \in \mathbb{R}^{p\times m}$ are periodic with period $K \geq 1$. Such models arise usually by the discretization of linear continuous-time periodic models which are the primary mathematical descriptions encountered in some practical applications [64]. The main advantage of using discrete-time models instead of continuous-time ones is the possibility to develop and to use efficient computational algorithms which completely parallel those for standard discrete-time systems.

Notation. For an arbitrary periodic matrix X_k of period K we use alternatively the *script* notation $\mathcal{X}$ which associates the block-diagonal matrix $\mathcal{X} = \operatorname{diag}(X_0, X_1, \ldots, X_{K-1})$ to the cyclic sequence of matrices X_k, $k = 0, \ldots, K-1$. This notation is consistent with the standard matrix operations. For example the operations with block-diagonal matrices $\mathcal{X}+\mathcal{Y}$, $\mathcal{X}\mathcal{Y}$, or $\mathcal{X}^{-1}$ can be used to express the addition, the multiplication and the inversion, respectively, performed simultaneously with all individual terms in a sequence of K matrices. We denote with $\sigma\mathcal{X}$ the K-cyclic shift $\sigma\mathcal{X} = \operatorname{diag}(X_1, \ldots, X_{K-1}, X_0)$ applied to the cyclic sequence X_k, $k = 0, \ldots, K$–1.

By using the script notation, the system (14) will be alternatively referred to as the quadruple $G = (\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$ or as the triple $G = (\mathcal{A}, \mathcal{B}, \mathcal{C})$ if $\mathcal{D} = 0$ or is not important for the context.

2.11 Periodic State-Space Transformations

Two periodic systems $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$ and $(\widetilde{\mathcal{A}}, \widetilde{\mathcal{B}}, \widetilde{\mathcal{C}}, \mathcal{D})$ related by

$$\widetilde{\mathcal{A}} = (\sigma\mathcal{T})^{-1}\mathcal{A}\mathcal{T}, \quad \widetilde{\mathcal{B}} = (\sigma\mathcal{T})^{-1}\mathcal{B}, \quad \widetilde{\mathcal{C}} = \mathcal{C}\mathcal{T}, \tag{15}$$

where $\mathcal{T}$ is an invertible matrix, are called *similar* and the transformation (15) is called a *periodic similarity transformation.*

The similarity transformations are the basic preprocessing tools for most of analysis, model conversion and synthesis problems. From numerical point of view, it is important that the periodic transformation matrix $\mathcal{T}$ to be a well-conditioned (ideally an orthogonal) periodic matrix.

2.11.1 Periodic Hessenberg and Schur Decompositions

The key role in many computational methods for periodic systems plays the *periodic Schur decomposition* (PSD) of a cyclic matrix product [14, 20]. According to [14], given the matrices A_k, $k = 0, 1, \ldots, K-1$, there exist orthogonal matrices Z_k, $k = 0, 1, \ldots, K-1$ such that $\widetilde{A}_{K-1} = Z_0^T A_{K-1} Z_{K-1}$ is in RSF and the matrices $\widetilde{A}_k = Z_{k+1}^T A_k Z_k$ for $k = 0, \ldots, K-2$ are upper triangular. Thus by using the PSD algorithm, we can determine the orthogonal matrices Z_k, $k = 0, \ldots, K$–1 to reduce the cyclic product $A_{K-1} \cdots A_1 A_0$ to the RSF without forming explicitly this product. An intermediary step in this reduction is the computation of the *periodic Hessenberg* *form* (PHF) where $\widetilde{A}_{K-1}$ is in Hessenberg form and the matrices $\widetilde{A}_k$ for $k = 0, \ldots, K-2$ are upper triangular.

LAPACK based numerical software to compute the PHF and PSD are provided in RASP. Specifically, the following available subroutines are well suited for standardization within SLICOT:

PSCHUR	computes the real Schur decomposition $T = T_1 T_2 \cdots T_p$ and the eigenvalues of the forward matrix product $A = A_1 A_2 \cdots A_p$, where T_1 is in an upper RSF and $T_2, \ldots, T_p$ are upper triangular matrices, by using orthogonal similarity transformations. Alternatively, PSCHUR can be used to compute the Schur decomposition $T = T_p \cdots T_2 T_1$ and the eigenvalues of the reverse matrix product $A = A_p \cdots A_2 A_1$
PSHESS	computes the upper Hessenberg form $H = H_1 H_2 \cdots H_p$ of the matrix product $A = A_1 A_2 \cdots A_p$, where H_1 is in an upper Hessenberg form and $H_2, \ldots, H_p$ are upper triangular matrices, by using orthogonal similarity transformations
PSHTR	generates the real orthogonal matrices Q_1, Q_2, ..., Q_p which are defined as the product of n-1 elementary reflectors of order n, $Q_j = H_j(1) H_j(2) \cdots H_j(n-1)$, as returned by PSHESS
PSHQR	computes the Schur decomposition and the eigenvalues of a product of matrices $H = H_1 H_2 \cdots H_p$, where H_1 is upper Hessenberg and $H_2, \ldots,$ H_p are upper triangular matrices

2.11.2 Periodic Lyapunov Equations

The subroutine DPLYAP is available in RASP to solve the *reverse-time discrete periodic Lyapunov equations* (RTDPLEs)

$$\mathcal{X} = \mathcal{A}^T \sigma \mathcal{X} \mathcal{A} + \mathcal{C}, \quad \mathcal{X} = \mathcal{A} \sigma \mathcal{X} \mathcal{A}^T + \mathcal{C}, \tag{16}$$

and of the related dual *forward-time discrete periodic Lyapunov equations* (FTDPLEs)

$$\sigma \mathcal{X} = \mathcal{A} \mathcal{X} \mathcal{A}^T + \mathcal{C}, \quad \sigma \mathcal{X} = \mathcal{A}^T \mathcal{X} \mathcal{A} + \mathcal{C}, \tag{17}$$

where $\mathcal{C}$ is a symmetric periodic matrix. This subroutine implements reliable numerical algorithms proposed in [60] and is well suited for standardization within SLICOT.

2.12 Optimal Periodic Output Feedback

For the control of the periodic system (14) an optimal periodic output feedback control law

$$u_k^* = F_k y_k \tag{18}$$

can be computed which minimizes the performance index

$$J = E \left\{ \frac{1}{2} \sum_{k=0}^{\infty} [\, x_k^T Q_k x_k + u_k^T R_k u_k \,] \right\}. \tag{19}$$

For the solution of this problem in general, no closed form solutions can be found even for standard state space systems. Thus iterative search methods must be used to compute the optimizing periodic output feedback matrix F_k. For search methods based on gradient techniques it is necessary to evaluate for a given stabilizing periodic output feedback F_k the corresponding values of the cost functional (19) and of its gradient with respect to F_k. Explicit formulas for this purpose have been derived and are reported in [64]. The computations involve the solution of two periodic Lyapunov equations.

Software for the computation of the optimal periodic output feedback is available in RASP. The subroutine PFUNGR calculates the function value $J(\mathcal{F})$ and the value of the gradient $\nabla_{\mathcal{F}} J(\mathcal{F})$ for a given feedback $\mathcal{F}$. The subroutine PERLQR computes the optimal periodic output feedback gain $\mathcal{F}$ which minimize the performance function $J(\mathcal{F})$. The computational approach and derivation of formulas for function and gradient evaluations are presented in [64]. Both above routines are well suited for standardization within SLICOT, provided the MINPACK-2 minimization routines are freely available.

3 Factorization of Proper Transfer Function Matrices

3.1 Coprime Factorizations

3.1.1 Left/right coprime factorizations with prescribed stability degree

For a system $G = (A, B, C, D)$ with the TFM

$$G(\lambda) = C(\lambda I - A)^{-1}B + D$$

a *left coprime factorization* (LCF) is defined as the fractional representation $G = M^{-1}N$, where N and M are stable and proper rational matrices and where there exist stable and proper rational U and V such that $NU + MV = I$. Similarly,a *right coprime factorization* (RCF) is defined as the fractional representation $G = NM^{-1}$, where N and M are stable and proper rational matrices, and where there exist stable and proper rational U and V such that $UN + VM = I$.

To determine LCFs or RCFs, where the factors N and M have a prescribed stability degree, the following routines are available in RASP-MODRED and are suitable for standardization within SLICOT:

LCFS	computes the state-space representations of the factors of a LCF with prescribed stability degree [50].
RCFS	computes the state-space representations of the factors of a RCF with prescribed stability degree [50].
LCFI	computes the state-space representation of the TFM corresponding to a LCF
RCFI	computes the state-space representation of the TFM corresponding to a RCF

3.1.2 Left/right coprime factorizations with inner denominators

A LCF $G = M^{-1}N$ or a RCF $G = NM^{-1}$ with the additional restriction that the denominator factor M is *inner* (that is, $M^{\sim}M = I$, where $M^{\sim}(s) = M^T(-s)$ for a continuous-time system and $M^{\sim}(z) = M^T(1/z)$ for a discrete-time system) are called *left coprime factorization with inner denominator* (LCFID) and *right coprime factorization with inner denominator* (RCFID), respectively. This factorizations are useful in computing L_2 and l_2 norms of unstable systems.

To determine the LCFIDs or RCFIDs, the following routines are available in RASP-MODRED and are suitable for standardization within SLICOT:

LCFID	computes the state-space representations for the factors of a LCFID of a TFM [51].
RCFID	computes the state-space representations for the factors of a RCFID of a TFM [51].

3.1.3 Left/right normalized coprime factorization

Normalized coprime factorizations of proper TFMs have many important theoretical and practical applications. From theoretical part, they can be used to define topological properties for rational matrices [68]. Practical applications include model/controller reduction and robust controller design [24].

A LCF is called *normalized left coprime factorization* (NLCF) if $MM^{\sim} + NN^{\sim} = I$. Similarly, a RCF is called a *normalized right coprime factorization* (NRCF) if $M^{\sim}M + N^{\sim}N = I$.

The computation of normalized coprime factorization is based on solving a Riccati equation and the factorization formulas are described in [69] for continuous-time systems and in [15] for discrete-time systems. There is no available software for these computation, although the implementation of such a software is relatively straightforward using the existing SLICOT routines. Prototype routines in MATLAB are available for computing normalized coprime factorizations based on a more general algorithm applicable to arbitrary (possibly improper) TFMs [61].

3.2 Inner-outer Factorization

A factorization of a stable TFM G as $G = G_iG_o$, where G_i is inner and G_o is outer (full row rank, stable and minimum-phase) is useful to solve some H_∞ control problems. Algorithms to compute this factorization has been recently developed in the most general setting [29] and prototype implementations will be available soon in MATLAB. However, the underlying computational approach is based on a descriptor system approach and is very involved. Therefore, the Fortran implementation of the inner-outer factorization algorithm is questionable.

4 Algorithms and Software for Descriptor Systems

Consider the linear descriptor system G

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} , \tag{20}$$

where $E, A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$, and where λ is either the differential operator d/dt or the advance operator z, depending on the type of the system. Generally, the matrix E can be singular, but we shall assume in what follows that the pencil $\lambda E - A$ is regular, that is $\det(\lambda E - A) \not\equiv 0$. The system (20) will be alternatively referred to as the quadruple $G = (A - \lambda E, B, C, D)$ or as the triple $G = (A - \lambda E, B, C)$ if $D = 0$. Its TFM is the $p \times m$ rational matrix

$$G(\lambda) = C(\lambda E - A)^{-1}B + D. \tag{21}$$

4.1 Descriptor State-Space Transformations

Two descriptor representations $(A - \lambda E, B, C, D)$ and $(\widetilde{A} - \lambda \widetilde{E}, \widetilde{B}, \widetilde{C}, \widetilde{D})$ related by

$$\widetilde{A} - \lambda \widetilde{E} = Q(A - \lambda E)Z, \quad , \widetilde{B} = QB, \quad \widetilde{C} = CZ, \tag{22}$$

where Q and Z are square invertible matrices, are called *similar* and the transformation (22) is called a *similarity transformation*. Note that similar descriptor systems have the same TFM.

The similarity transformations are the basic preprocessing tools for most of analysis, model conversion and synthesis problems discussed in this chapter. From numerical point of view, it is important that the transformation matrices Q and Z to be well-conditioned (ideally orthogonal) matrices. Many useful reductions of the system matrices to special condensed forms (see later) can be done by using exclusively orthogonal transformations.

4.1.1 Descriptor state-space scaling

Given a descriptor system $G = (A - \lambda E, B, C)$, we compute the diagonal transformation matrices Q and Z to make the rows and columns of of the matrices of the transformed system pencil

$$\widetilde{\mathcal{S}}(\lambda) = \left[\begin{array}{c|c} \widetilde{A} - \lambda \widetilde{E} & \widetilde{B} \\ \hline \widetilde{C} & 0 \end{array}\right].$$

as close in norm to 1 as possible. Such a transformation is frequently necessary to improve the accuracy of numerical computations involving the system matrices. Note that the scaling can be equally performed only on the pairs $(A - \lambda E, B)$ or $(A - \lambda E, C)$ or even on a the pencil $A - \lambda E$.

To perform such a scaling, a routine similar to the DGGBAL from LAPACK, can be implemented along the lines of the BALABC routine (available in RASP) and intended for standardization within SLICOT. Note that it is generally advisable to include a scaling option in all analysis and design routines to overcome possible numerical problems caused by poor scaling of the original descriptor state-space representations.

4.1.2 Additive decomposition of rational matrices

In several applications, it is necessary to perform an additive decomposition of a rational matrix G as $G = G_1 + G_2$, where G_1 and G_2 are determined such that G_1 has only poles in a given region and G_2 has only poles outside that region. Useful decompositions rely on stable/unstable or finite/infinite separation of the generalized eigenvalues of the pair (A, E). For instance, the finite/infinite separation is useful to separate the proper and the polynomial part of the TFM of a given descriptor system.

For $G = (A - \lambda E, B, C)$ the main computation involves the determination of the transformation matrices Q and Z such that the transformed system has the form

$$\left[\begin{array}{c|c} Q(A-\lambda E)Z & QB \\ \hline CZ & 0 \end{array}\right] := \left[\begin{array}{cc|c} A_1 - \lambda E_1 & 0 & B_1 \\ 0 & A_2 - \lambda E_2 & B_2 \\ \hline C_1 & C_2 & 0 \end{array}\right],$$

where the pairs (A_1, E_1) and (A_2, E_2) contain the systems poles lying in the given region and outside that region, respectively. The additive terms are then defined by $G_1 := (A_1 - \lambda E_1, B_1, C_1)$ and $G_2 := (A_2 - \lambda E_2, B_2, C_2)$.

The implemention of a routine for additive decomposition using the algorithm proposed in [21], must rely on several lower level routines which are necessary to be implemented first:

- **Finite/infinite separation.** Such a routine, RPDSFI, exists in RASP-DESCRIPT and the implemented method is described in [45] (a simpler version of a more general approach proposed in [39]). RPDSFI computes the orthogonal matrices Q and Z such that

$$Q(A - \lambda E)Z = \begin{bmatrix} A_{11} - \lambda E_{11} & A_{12} - \lambda E_{12} \\ 0 & A_{22} - \lambda E_{22} \end{bmatrix}, \qquad (23)$$

  where the pair (A_{11}, E_{11}) has only finite generalized eigenvalues and the pair (A_{22}, E_{22}) has only infinite generalized eigenvalues. For standardization in SLICOT it would be useful a more versatile separation routine, taking into account other potential applications, as for instance factorizations of rational matrices [58]. Specifically, a separation of the non-dynamic and dynamic infinite poles would be also desirable, covering any permutation of the diagonal blocks of following more structured pencil:

$$Q(A - \lambda E)Z = \begin{bmatrix} A_f - \lambda E_f & * & * \\ 0 & A_\infty - \lambda E_\infty & * \\ 0 & 0 & A_n \end{bmatrix},$$

  where $A_f - \lambda E_f$ contains the finite poles of the system, $A_\infty - \lambda E_\infty$ contains the dynamic (impulsive) modes of the system, and A_n is nonsingular, corresponding to the non-dynamic poles of the system. The computation different ordering combinations can be determined by calling the SLICOT routines MB04SD and MB04TD in combination with a pertransposition routine.

- **Reordering of finite generalized eigenvalues.** A reordering routine, RPDSOS, similar to SEOR1, is available in RASP to perform various separations of the spectrum of the pair (A, E). Similarly with SEOR1, the standardized version of this routine must be more versatile than the corresponding LAPACK routine DGGES, allowing separations with respect to any translation of the imaginary axis or with respect to an arbitrary circle in the origin.

- **Solution of generalized Sylvester equations.** After the separation of spectrum of the pair (A, E) as in (23), the off-diagonal term $A_{12} - \lambda E_{12}$ can be zeroed by pre- and post-multiplication of (23) with $\begin{bmatrix} I & -L \\ 0 & I \end{bmatrix}$ and $\begin{bmatrix} I & R \\ 0 & I \end{bmatrix}$, respectively, where L and R satisfy the generalized Sylvester equation

$$\begin{array}{rcl} A_{11}R - LA_{22} & = & -A_{12} \\ E_{11}R - LE_{22} & = & -E_{12}. \end{array}$$

Software for this computation is available in LAPACK and can be readily used to perform the above computations.

4.1.3 Orthogonal reduction of system matrices

Given the system $G = (A - \lambda E, B, C)$, the following RASP-DESCRIPT subroutines performs a similarity transformation (22), where the matrices Q and Z are orthogonal:

RPDSQR	reduces a descriptor system to the QR-coordinate form with $\widetilde{E} = QE$ upper triangular and $Z = I$
RPDSRQ	reduces a descriptor system to the RQ-coordinate form with $\widetilde{E} = EZ$ upper triangular and $Q = I$
RPDSSV	reduces a descriptor system to the singular value coordinate form with $\widetilde{E} = QEZ$ diagonal, having its singular values as diagonal elements
SRSET	reduces a descriptor system to a singular value like coordinate form with $\widetilde{E} = QEZ$ in a complete orthogonal decomposition form

All these routines could serve for standardization within SLICOT.

Another useful orthogonal similarity transformation is to reduce the pair $(A - \lambda E, B)$ to the descriptor controllability staircase form

$$\widetilde{A} - \lambda\widetilde{E} = Q(A - \lambda E)Z = \begin{bmatrix} A_{11} - \lambda E_{11} & A_{12} - \lambda E_{12} \\ 0 & A_{22} - \lambda E_{22} \end{bmatrix}, \quad \widetilde{B} = Q^T B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \tag{24}$$

where the pair $(A_{11} - \lambda E_{11}, B_1)$ is controllable and the pencil $[\, B_1 \; A_{11} - \lambda E_{11} \,]$ is in a staircase form. The pair (A_{22}, E_{22}) contains the finite uncontrollable generalized eigenvalues of the pair (A, E). Such a transformation, with optional accumulation of the transformation matrices Q and Z, is useful in an efficient implementation of a minimal realization routine for descriptor systems. An observability form equivalent of the above routine would be also desirable. Note that by interchanging the roles of A and E, the same method can be used to separate the uncontrollable infinite generalized eigenvalues simultaneously with the nonzero finite uncontrollable eigenvalues. The routine RPDSCF is available in RASP-DESCRIPT for the above computation and is based on an algorithm proposed in [44]. This routine can be used as starting point for a standardization within SLICOT and basis to implement a minimal realization routine for descriptor systems.

4.2 Descriptor System Conversions

4.2.1 Evaluation of the transfer function matrix

Given a descriptor state-space model $G = (A - \lambda E, B, C, D)$ it is often necessary to determine the corresponding TFM (21). One of the most reliable methods available for this computations seems to be the poles-zeros method of Varga [45]. This algorithm determines each element of $G(\lambda)$ as a product of a scalar gain with a monic polynomial for the corresponding zeros divided by a monic polynomial for the corresponding poles. A routine RPDSDT, implementing the above algorithm, is available in RASP-DESCRIPT and can constitute the starting point for a standardization effort. However, a more robust implementation of the poles-zeros algorithm implies to eliminate the infinite poles and infinite zeros as well using a specially devised approach to avoid the unreliable detection of infinite eigenvalues among the computed whole set of generalized eigenvalues.

4.2.2 Irreducible descriptor representation

Given a descriptor system $G = (A - \lambda E, B, C)$ an *irreducible* realization of least order $G = (\widehat{A} - \lambda\widehat{E}, \widehat{B}, \widehat{C})$ having the same TFM can be determined by eliminating successively the uncontrollable and unobservable finite and infinite poles using the controllability staircase algorithm of [44]. The subroutine RPDSIR, based on this approach, is available in RASP-DESCRIPT and could serve as basis for standardization. An enhancement of this routine is possible by also using the algorithm in [58] to remove the uncontrollable/unobservable infinite eigenvalues.

4.2.3 Minimal order descriptor representation for a rational transfer matrix

The construction of a minimal order descriptor representation of a rational matrix is possible using computational methods for standard state-space systems [67]. For a given rational matrix $G(\lambda)$ we compute first the additive separation

$$G(\lambda) = G_{prop}(\lambda) + G_{pol}(\lambda),$$

where $G_{prop}(\lambda)$ and $G_{pol}(\lambda)$ are the proper and the polynomial parts, respectively, of $G(\lambda)$. Then compute the standard state-space realizations

$$G_{prop}(\lambda) = C_f(\lambda I - A_f)^{-1}B_f + D, \quad \lambda^{-1}G_{pol}(\lambda^{-1}) = C_\infty(\lambda I - E_\infty)^{-1}B_\infty$$

and assemble the matrices of the descriptor system $G = (A - \lambda E, B, C, D)$ as

$$A = \left[\begin{array}{cc} A_f & 0 \\ 0 & I \end{array}\right], \quad E = \left[\begin{array}{cc} I & 0 \\ 0 & E_\infty \end{array}\right], \quad B = \left[\begin{array}{c} B_f \\ B_\infty \end{array}\right], \quad C = \left[\begin{array}{cc} C_f & C_\infty \end{array}\right].$$

This approach is simple to be implemented provided a minimal realization subroutine for proper systems will be available in SLICOT.

Alternatively, it is possible to generate ad-hoc non-minimal state-space representations for the proper and the polynomial parts

$$G_{prop}(\lambda) = C_f(\lambda I - A_f)^{-1}B_f + D, \quad G_{pol}(\lambda) = C_\infty(\lambda E_\infty - A_\infty)^{-1}B_\infty$$

and to assemble the matrices of the descriptor system $G = (A - \lambda E, B, C, D)$ as

$$A = \left[\begin{array}{cc} A_f & 0 \\ 0 & A_\infty \end{array}\right], \quad E = \left[\begin{array}{cc} I & 0 \\ 0 & E_\infty \end{array}\right], \quad B = \left[\begin{array}{c} B_f \\ B_\infty \end{array}\right], \quad C = \left[\begin{array}{cc} C_f & C_\infty \end{array}\right].$$

Then a minimal realization routine for descriptor systems can be used to determine a minimal order descriptor realization for G. A routine RPDSTD to generate a non-minimal descriptor representation of an arbitrary rational matrix is available in RASP-DESCRIPT. This routines calls another subroutine RPDSPF which computes a descriptor representation for a polynomial matrix. Both routines can serve for standardization within SLICOT.

4.2.4 Descriptor representation for a polynomial matrix model

Besides descriptor and transfer matrix representations of generalized systems, another description arises in practice, namely the *polynomial matrix model*

$$\begin{array}{rcl} T(\lambda)\xi(t) & = & U(\lambda)u(t) \\ y(t) & = & V(\lambda)\xi(t) + W(\lambda)u(t), \end{array} \tag{25}$$

where ξ is the so called *internal state* and $T(\lambda) \in \mathbb{R}^{q\times q}(\lambda)$, $U(\lambda) \in \mathbb{R}^{q\times m}(\lambda)$, $V(\lambda) \in \mathbb{R}^{p\times q}(\lambda)$, $W(\lambda) \in \mathbb{R}^{p\times m}(\lambda)$ are polynomial matrices.

A subroutine RPDSDM is available in RASP-DESCRIPT to generate a non-minimal descriptor realization for the polynomial matrix model (25) and is suitable for standardization. The resulted descriptor model can be reduced to a minimal order representation by using a descriptor minimal realization routine.

4.3 Analysis of Descriptor Systems

Typical computational analysis problems consists of determining properties as for instance stability, poles, zeros, controllability, observability, Kronecker structure of the system pencil etc. Many of the condensed form obtainable with the presented similarity transformations are useful in these computations.

The finite-infinite separation of the pair (A, E) can be used to evaluate the poles of the descriptor system. Note that multiplicity of infinite eigenvalues exceeds by one the multiplicity of infinite poles. The controllability can be determined on the basis of the staircase form (24).

Of particular interest for the analysis of descriptor systems is the computation of zeros, defined as the *Smith zeros* of the $(n+p) \times (n+m)$ system matrix pencil

$$\mathcal{S}(\lambda) = \left[\begin{array}{c|c} A - \lambda E & B \\ \hline C & D \end{array}\right].$$

A numerically reliable method to compute the system zeros can be viewed as a powerful analysis tools because practically all above mentioned properties of the system (20) can be easily assessed by computing the zeros of particular system matrices (for $p = 0$ and/or $m = 0$).

A numerically reliable general method to compute the system zeros together with the complete Kronecker structure of $S(\lambda)$ has been proposed in [27]. Although no software for this algorithm exists, an implementation can be derived using the recently implemented codes [56] to compute Kronecker-like forms. The most appropriate for standardization within SLICOT is the SPRED routine which computes the following decomposition of the system pencil

$$Q^T \mathcal{S}(\lambda) Z = \left[\begin{array}{c|cccccc} B_r & A_r - \lambda E_r & * & * & * & * \\ 0 & 0 & A_\infty - \lambda E_\infty & * & * & * \\ 0 & 0 & 0 & D_i & * & * \\ 0 & 0 & 0 & 0 & A_f - \lambda E_f & * \\ 0 & 0 & 0 & 0 & 0 & A_l - \lambda E_l \\ \hline 0 & 0 & 0 & 0 & 0 & C_l \end{array}\right] \qquad (26)$$

where:

(a) the pencil $[\ B_r \ \ A_r - \lambda E_r\ ]$ contains the right Kronecker structure of $\mathcal{S}(\lambda)$ and E_r is invertible and upper-triangular; the pair $(B_r, A_r - \lambda E_r)$ is controllable and the pencil $[\ B_r \ \ A_r - \lambda E_r\ ]$ is in the controllability staircase form.

(b) the *regular* pencil $A_\infty - \lambda E_\infty$ together with D_i contain the infinity Kronecker structure of $\mathcal{S}(\lambda)$; A_∞ and D_i are invertible and upper-triangular, and E_∞ is nilpotent and upper-triangular.

(c) the *regular* pencil $A_f - \lambda E_f$ contains the finite Kronecker structure of $\mathcal{S}(\lambda)$ and E_f is invertible and upper-triangular.

(d) the pencil $\left[\begin{array}{c} A_l - \lambda E_l \\ C_l \end{array}\right]$ contains the left Kronecker structure of $\mathcal{S}(\lambda)$ and E_l is invertible and upper-triangular; the pair $(C_l, A_l - \lambda E_l)$ is observable and the pencil $\left[\begin{array}{c} A_l - \lambda E_l \\ C_l \end{array}\right]$ is in the observability staircase form.

Excepting the finite eigenvalues structure, the above form contains identical structural information as the Kronecker canonical form. The detailed structure of the subpencils of this form is given in [53]. The associated dimensional index sets determine the minimal indices and the infinite structure of the system pencil $\mathcal{S}(\lambda)$.

The subroutine SPRED is well suited for standardization within SLICOT and offers an alternative method to compute the Kronecker-like forms. A set of similar routines belonging to RASP to compute various Kronecker-like forms of lower complexity is described in [53] and could serve for standardization purposes too.

4.4 Generalized Lyapunov Equations

The subroutine DGLP [32] is available for the solution of the *generalized continuous-time Lyapunov equation* (GCLE) in one the forms

$$AXE^T + EXA^T + C = 0, \quad A^T XE + E^T XA + C = 0$$

and of the *generalized discrete-time Lyapunov equation* (GDLE) in one of the forms

$$AXA^T - EXE^T + C = 0, \quad A^T XA - E^T XE + C = 0$$

where $A, E, C \in \mathbb{R}^{n\times n}$, C is symmetric, and E is nonsingular. This subroutine implements reliable numerical algorithms proposed in [16] and is well suited for standardization within SLICOT.

For the case of non-negative solution, when the pair (A, E) has stable generalized eigenvalues, that is, all eigenvalues lie in the open left half plane in the continuous-time case or inside the unite circle in the discrete-time case, and the matrix C has the form $C = R^T R$, a similar routine DGLPHM, exists. This routine implements an extension of the Hammarling's method [19] for standard nonnegative Lyapunov equations to the above more general case [32] and is well suited for standardization within SLICOT.

4.5 Generalized Riccati Equations

It is straightforward to extend the existing software in SLICOT (subroutines SB02ND and SB02OD) to solve the *generalized continuous-time algebraic Riccati equation* (GCARE)

$$Q + A^T XE + E^T XA - (L + XB)R^{-1}(L + XB)^T = 0 \tag{27}$$

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

$$A^T XA - E^T XE - (L + A^T XB)(R + B^T XB)^{-1}(L + A^T XB)^T + Q = 0, \tag{28}$$

where the E matrix is non-singular. Algorithms for this purpose are described in [4, 25].

References

[1] G. Ammar and P. Benner. Osmare: A Fortran 77 implementation of the orthogonal symplectic multishift method for continuous-time algebraic Riccati equations. Preprint, Zentrum für Technomathematik, FB 3 – Mathematik und Informatik, Universität Bremen, 28334 Bremen, FRG, 1998. In preparation.

[2] G.S. Ammar, P. Benner, and V. Mehrmann. A multishift algorithm for the numerical solution of algebraic Riccati equations. *Electr. Trans. Num. Anal.*, 1:33–48, 1993.

[3] E. Anderson, Z. Bai, J. Bishop, J. Demmel, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide, Second Edition.* SIAM, Philadelphia, 1995.

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

[5] C. Bavely and G. W. Stewart. An algorithm for computing reducing subspaces by block diagonalization. *SIAM J. Numer. Anal.*, 16:359–367, 1979.

[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. Symplectic balancing of Hamiltonian matrices. Technical report, Zentrum für Technomathematik, Fachbereich 3 – Mathematik und Informatik, Universität Bremen, 28334 Bremen (Germany), February 1998.

[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):101–108, 1998.

[9] P. Benner, R. Byers, and E. Barth. HAMEV and SQRED: Fortran 77 subroutines for computing the eigenvalues of Hamiltonian matrices using Van Loan's square reduced method. Technical Report SFB393/96-06, Fak. f. Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1996. Available from `http://www.tu-chemnitz.de/sfb393/sfb96pr.html`.

[10] P. Benner, R. Byers, E. S. Quintana-Ortí, and G. Quintana-Ortí. Parallel solvers for algebraic Riccati equations based on Newton's method. Technical report, Zentrum für Technomathematik, Fachbereich 3 – Mathematik und Informatik, Universität Bremen, 28334 Bremen (Germany), 1998. In preparation.

[11] P. Benner, A. J. Laub, and V. Mehrmann. A collection of benchmark examples for the numerical solution of algebraic Riccati equations II: Discrete-time case. Technical Report SPC 95_23, Fak. f. Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1995. Available from `http://www.tu-chemnitz.de/sfb393/spc95pr.html`.

[12] P. Benner, V. Mehrmann, and H. Xu. A new method for computing the stable invariant subspace of a real Hamiltonian matrix. *J. Comput. Appl. Math.*, 86:17–43, 1997.

[13] P. Benner, V. Mehrmann, and H. Xu. A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils. *Numer. Math.*, 78(3):329–358, 1998.

[14] A. W. Bojanczyk, G. Golub, and P. Van Dooren. The periodic Schur decomposition. Algorithms and applications. In F. T. Luk, editor, *Proceedings SPIE Conference*, volume 1770, pages 31–42, July 1992.

[15] P. M. M. Bongers and P. S. C. Heuberger. Discrete normalized coprime factorization. In A. Bensoussan and J. L. Lions, editors, *Proc. 9th INRIA Conf. Analysis and Optimization of Systems*, volume 144 of *Lect. Notes Control and Inf. Scie.*, pages 307–313. Springer-Verlag, Berlin, 1990.

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

[17] G. Grübel and H.-D. Joos. The control systems engineering numerical subroutine library RASP. Technical Report TR R14–90, DLR - German Aerospace Research Establishment, D-82230 Wessling, 1990.

[18] G. Grübel, A. Varga, A. van den Boom, and A. J. Geurts. Towards a coordinated development of numerical CACSD software: the RASP/SLICOT compatibility concept. In *Prepr. of IEEE/IFAC Symp. CACSD'94, Tucson, Arizona*, pages 499–504, 1994.

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

[20] J. J. Hench and A. J. Laub. Numerical solution of the discrete-time periodic Riccati equation. *IEEE Trans. Autom. Control*, 39:1197–1210, 1994.

[21] B. Kågström and P. Van Dooren. Additive decomposition of a transfer function with respect to a specified region. In *Proc. MTNS Symp., Brussels*, 1989.

[22] W.-W. Lin. A new method for computing the closed loop eigenvalues of a discrete-time algebraic Riccati equation. *Linear Algebra Appl.*, 6:157–180, 1987.

[23] D. G. Luenberger. An introduction to observers. *IEEE Trans. Autom. Control*, 16:596–602, 1971.

[24] D. C. McFarlane and K. Glover. *Robust Controller Design Using Normalized Coprime Factor Plant Descriptions*, volume 138 of *Lect. Notes in Control and Inf. Science*. Springer-Verlag, Berlin, 1989.

[25] V. L. Mehrmann. *The Autonomous Linear Quadratic Control Problem*, volume 163 of *Lect. Notes Contr. Inf. Scie.* Springer Verlag, Berlin, 1991.

[26] G. S. Miminis and C. C. Paige. A direct algorithm for pole assignment of time-invariant multi-input linear systems using state feedback. *Automatica*, 24:343–356, 1988.

[27] P. Misra, P. Van Dooren, and A. Varga. Computation of structural invariants of generalized state-space systems. *Automatica*, 30:1921–1936, 1994.

[28] C. B. Moler and C. F. Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20:801–836, 1978.

[29] C. Oara and A. Varga. Inner-outer factorization of rational matrices: the general case. In *Proceedings of the MTNS'98, Padova*, 1998.

[30] R. V. Patel and P. Misra. Numerical algorithm for eigenvalue assignment by state feedback. *Proc. IEEE*, 72:1755–1764, 1984.

[31] R.V. Patel. On computing the eigenvalues of a symplectic pencil. *Linear Algebra Appl.*, 188/189:591–611, 1993. See also: Proc. CDC-31, Tuscon, AZ, 1992, pp. 1921–1926.

[32] T. Penzl. Numerical solution of generalized Lyapunov equations. Preprint SFB393/96-02, Technical University Chemnitz, May 1996.

[33] P. H. Petkov, M. M. Konstantinov, D. W. Gu, and I. Postlethwaite. Optimal eigenstructure assignment of linear systems. Technical report, 93-64, Department of Engineering, Leicester University, UK, 1993.

[34] P. Hr. Petkov, N. D. Christov, and M. M. Konstantinov. A computational algorithm for pole assignment of linear multiinput systems. *IEEE Trans. Autom. Control*, AC-31:1755–1764, 1986.

[35] M. G. Safonov, E. A. Jonckheere, M. Verma, and D. J. N. Limebeer. Synthesis of positive real multivariable feedback systems. *Int. J. Control*, 45:817–842, 1987.

[36] V. Sima. On the real Schur form in linear control system design. *Rev. Roum. Scie. Techn. – Electrotech. et Energ.*, 25, 1980.

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

[38] J. Sreedhar and P. Van Dooren. Pole placement via the periodic Schur decomposition. In *Proc. 1993 American Control Conference, San Francisco, CA*, pages 1563–1567, 1993.

[39] P. Van Dooren. Computing the eigenvalues of a polynomial matrix. In *Proc. of the IBM-NFWO Symp., Brussels, Belgium*, pages 213–223, 1979.

[40] P. Van Dooren. The generalized eigenstructure problem in linear systems theory. *IEEE Trans. Autom. Control*, 26:111–129, 1981.

[41] A. Varga. On stabilization algorithms for linear time-invariant systems. *Rev. Roum. Scie. Techn. – Electrotech. et Energ.*, 26:115–124, 1981.

[42] A. Varga. A Schur method for pole assignment. *IEEE Trans. Autom. Control*, AC-26:517–519, 1981.

[43] A. Varga. Computer aided design of robust compensators by pole assignment. In *Preprints of SOCOCO'82 Symp., Madrid*, 1982.

[44] A. Varga. Computation of irreducible generalized state-space realizations. *Kybernetika*, 26:89–106, 1989.

[45] A. Varga. Computation of transfer function matrices of generalized state-space models. *Int. J. Control*, 50:2543–2561, 1989.

[46] A. Varga. Numerical algorithms and software tools for analysis and modelling of descriptor systems. In *Prepr. of 2nd IFAC Workshop on System Structure and Control, Prague, Czechoslovakia*, pages 392–395, 1992.

[47] A. Varga. On computing 2-norms of transfer function matrices. In *Proc. 1992 American Control Conference, Chicago, Illinois*. Elsevier, Amsterdam, 1992.

[48] A. Varga. *RASP Model Order Reduction Programs*. University of Bochum and DLR-Oberpfaffenhofen, TR R88-92, August 1992.

[49] A. Varga. Computational techniques based on the block-diagonal form for solving large systems modeling problems. In *Proc. 1993 IEEE Conference on Aerospace Control Systems, Westlake Village, CA*, pages 693–697, 1993.

[50] A. Varga. Coprime factors model reduction based on accuracy enhancing techniques. *Systems Analysis Modelling and Simulation*, 11:303–311, 1993.

[51] A. Varga. A Schur method for computing coprime factorizations with inner denominators and applications in model reduction. In *Proc. 1993 American Control Conference, San Francisco, CA*, pages 2130–2131, 1993.

[52] A. Varga. Generalized Schur methods to compute coprime factorizations of rational matrices. In *Proc. 1st Asian Control Conference, Tokyo, Japan*, volume 3, pages 89–92, 1994.

[53] A. Varga. Computation of Kronecker-like forms of a system pencil: Applications, algorithms and software. TR R181-95, DLR-Oberpfaffenhofen, Inst. Robotics and System Dynamics, January 1995.

[54] A. Varga. Enhanced modal approach for model reduction. *Mathematical Modelling of Systems*, 1:91–105, 1995.

[55] A. Varga. On stabilization of descriptor systems. *Systems & Control Letters*, 24:133–138, 1995.

[56] 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.

[57] A. Varga. Optimal output feedback control: a multi-model approach. In *Proc. CACSD'96 Symposium, Dearborn, MI*, pages 327–332, 1996.

[58] A. Varga. Computation of coprime factorizations of rational matrices. *Lin. Alg. & Appl.*, 271:83–115, 1997.

[59] A. Varga. Parametric methods for pole assignment. In *Proc. ECC'97, Brussels*, 1997.

[60] A. Varga. Periodic Lyapunov equations: some applications and new algorithms. *Int. J. Control*, 67:69–87, 1997.

[61] A. Varga. Computation of normalized coprime factorizations of rational matrices. *Systems & Control Lett.*, 1998. (to appear).

[62] A. Varga and A.Davidoviciu. BIMASC - A package of Fortran subprograms for analysis, modelling, design and simulation of control systems. In *Proc. of 3-rd IFAC/IFIP Symp. CADCE'85, Copenhagen, Denmark*. Pergamon Press, Oxford, 1986.

[63] A. Varga and V. Ionescu. HTOOLS - A Toolbox for solving H_∞ and H_2 synthesis problems. In *Proc. of IFAC/IMACS Symp. on Computer Aided Design of Control Systems, Swansea, UK*, pages 508–511, July 1991.

[64] A. Varga and S. Pieters. Gradient-based approach to solve optimal periodic output feedback control problems. *Automatica*, 34(4), 1998.

[65] A. Varga and V. Sima. A numerically stable algorithm for transfer-function matrix evaluation. *Int. J. Control*, 33:1123–1133, 1981.

[66] A. Varga and V. Sima. BIMAS - A basic mathematical package for computer aided systems analysis and design. In J. Gerter and L. Keviczky, editors, *Proc. of 9-th IFAC World Congress, Budapest, Hungary*, 1985.

[67] G. Verghese, P. Van Dooren, and T. Kailath. Properties of the system matrix of a generalized state-space system. *Int. J. Control*, 30:235–243, 1979.

[68] M. Vidyasagar. *Control System Synthesis: A Factorization Approach.* The MIT Press, Cambridge, MA, 1985.

[69] M. Vidyasagar. Normalized coprime factorization for nonstrictly proper systems. *IEEE Trans. Autom. Control*, 33:300–301, 1988.

A List of Routines to be Standardized for Basic Control Problems

A.1 Mathematical Routines

Name	Function
MB03RD	computes the bloc diagonal form of a square matrix
MB03QD	reorders the eigenvalues of a real Schur matrix according to several reordering criteria
MB03SD	computes the periodic Schur decomposition of a matrix product
MB03TD	computes the periodic Hessenberg decomposition of a matrix product
MB03VD	generates the real orthogonal matrices returned by PSHESS
MB03WD	computes the periodic Schur decomposition of a matrix product in a periodic Hessenberg form
MB03XD	reorders the generalized eigenvalues of a matrix pair (A, E) in a generalized real Schur form according to several reordering criteria
MB03UD	computes all, or part, of the singular value decomposition of an upper triangular matrix
MB05PD	computes the matrix exponential using the block diagonal form reduction

A.2 Transformation Routines

Name	Function
TB01ID	performs the scaling of a state-space model
TB01JD	applies a general similarity transformation
TB01KD	computes the terms G_1 and G_2 of an additive spectral decomposition of a transfer-function matrix G with respect to a specified region of the complex plane
TB01LD	performs an orthogonal similarity trasformation to reduce the system state matrix to an ordered real Schur form
TB01OD	computes the orthogonal controllability staircase form of a state-space model
TB01RD	computes the orthogonal observability staircase form of a state-space model
TB01WD	performs an orthogonal similarity trasformation to reduce the system state matrix to the real Schur form

A.3 Analysis Routines

Name	Function
AB13AD	computes the Hankel norm and the Hankel singular values of the stable projection of a transfer-function matrix
AB13BD	computes the L_2- or l_2-norm of a transfer-function matrix
AB13CD	computes the H_∞ norm of a stable/unstable transfer-function matrix

A.4 Synthesis Routines

Name	Function
SB01BD	performs multi-input pole assignment using the Schur method
SB01DD	performs parametric multi-input pole assignment using the Hessenberg method
SB01ED	computes a stabilizing state feedback for a state-space system
SB02ED	solves the CARE or DARE using Newton's method
SB02FD	condition estimation of CARE or DARE
SB02PD	solves the CARE using symplectic method
SB02KD	symplectic balancing
SB02LD	driver routine for eigenvalues of Hamiltonian matrices
SB02PD	solves the CARE using symplectic method
SB03OD	solves for $X = \mathrm{op}(U)'\mathrm{op}(U)$ either the stable non-negative definite continuous-time Lyapunov equation $\mathrm{op}(A)'X + X\mathrm{op}(A) = -\sigma^2\mathrm{op}(B)'\mathrm{op}(B)$ or the convergent non-negative definite discrete-time Lyapunov equation $\mathrm{op}(A)'X\mathrm{op}(A) - X = -\sigma^2\mathrm{op}(B)'\mathrm{op}(B)$, where $\mathrm{op}(K) = K$ or K'.

A.5 Factorization Routines

Name	Function
SB08AD	computes the state-space representations of the factors of a LCF with prescribed stability degree
SB08BD	computes the state-space representations of the factors of a RCF with prescribed stability degree
SB08CD	computes the state-space representations of the factors of a LCFID of a TFM
SB08DD	computes the state-space representations of the factors of a RCFID of a TFM
SB08GD	computes the state-space representation of the TFM corresponding to a LCF
SB08HD	computes the state-space representation of the TFM corresponding to a RCF

B List of Descriptor System Routines

B.1 Transformation Routines

Name	Function
TG01AD	performs the scaling of a descriptor system model
TG01CD	reduces a descriptor system to the QR-coordinate form with $\widetilde{E} = QE$ upper triangular and $Z = I$
TG01DD	reduces a descriptor system to the RQ-coordinate form with $\widetilde{E} = EZ$ upper triangular and $Q = I$
TG01ED	reduces a descriptor system to the singular value coordinate form with $\widetilde{E} = QEZ$ diagonal, having its singular values as diagonal elements
TG01FD	reduces a descriptor system to a singular value like coordinate form with $\widetilde{E} = QEZ$ in a complete orthogonal decomposition form
TG01HD	computes the controllability staircase form of a descriptor system
TG01ID	computes the observability staircase form of a descriptor system
TG01JD	computes an irreducible descriptor representation from a non-minimal one
TG01KD	computes a descriptor system staircase form exhibiting uncontrollable infinite eigenvalues

B.2 Analysis Routines

Name	Function
AG08BD	computes the zeros and the Kronecker structure of a descriptor system pencil

B.3 Synthesis Routines

Name	Function
SG02AD	solves GCAREs and GDAREs
SG03AD	solves GCLEs and GDLEs
SG03BD	solves non-negative GCLEs and GDLEs