A Descriptor Systems Toolbox for MATLAB

A. Varga

German Aerospace Center (DLR) - Oberpfaffenhofen
Institute of Robotics and Mechatronics
D-82234 Wessling, Germany
Andras.Varga@dlr.de

Abstract

We describe a recently developed DESCRIPTOR SYSTEMS Toolbox implemented under MATLAB. This Toolbox relies on the object oriented approach for control systems analysis and design provided within the standard CONTROL Toolbox of MATLAB. The basic approach to develop the DESCRIPTOR SYSTEMS Toolbox was to exploit the powerful matrix and system object manipulation features of MATLAB via flexible and functionally rich high level *m*-functions, while simultaneously enforcing highly efficient and numerically sound structure exploiting computations via the *mex*-function technology of MATLAB to solve critical numerical problems. The *mex*-functions are based on FORTRAN codes from LAPACK and SLICOT.

1 Why a DESCRIPTOR SYSTEMS Toolbox?

It is well-known that a descriptor system of the form

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

with E square and possibly singular and with $A - \lambda E$ a regular matrix pencil, is the most general description for a linear time-invariant continuous-time system. Such systems arise frequently from modelling interconnected systems even with standard tools like Simulink (recall the "algebraic loop" warning). Descriptor models are also common in modelling constrained mechanical systems (e.g., contact problems). Moreover, the descriptor representation is necessary to perform some operations even with standard systems like conjugation or inversion. Discrete-time descriptor representations are frequently used to model economic processes.

The DESCRIPTOR SYSTEMS Toolbox is primarily intended to provide an extended functionality for the CONTROL Toolbox of MATLAB by allowing the manipulation of descriptor systems, the most general class of linear system models. Although these models are formally supported in the CONTROL Toolbox, systems with singular E are not allowed. This is why some functions in the DESCRIPTOR SYSTEMS Toolbox represent simply extensions of functions already present in the CONTROL Toolbox. The other functions are new and allow, for the first time, a convenient user-friendly operation to solve the most complicated dynamic analysis problems, as for example, the determination of complete Kronecker-structure of a linear pencil.

The DESCRIPTOR SYSTEMS Toolbox is also useful for manipulating rational and polynomial matrices. Recall that each rational matrix $R(\lambda)$ can be seen as the *transfer-function matrix* (TFM) of a continuous- or discrete-time descriptor system. Thus, each $R(\lambda)$ can be equivalently realized by a descriptor system quadruple $(A - \lambda E, B, C, D)$ satisfying

$$R(\lambda) = C(\lambda E - A)^{-1}B + D,$$

where $\lambda = s$ for a continuous-time descriptor realization or $\lambda = z$ for discrete-time descriptor realization. It is widely accepted that most of numerical operations on rational matrices, and in particular on polynomial matrices, are best done by manipulating instead the matrices of the corresponding descriptor systems descriptions. Many operations on standard matrices have nice generalizations for rational matrices. Straightforward generalizations are the rank, determinant, inverse and several generalized inverses. The conjugate transposition of a complex matrix M^* generalizes to the conjugation of a rational matrix $R^*(\lambda)$, where $R^*(s) = R^T(-s)$ or $R^*(z) = R^T(1/z)$, while the full-rank, inner-outer and spectral factorizations can be seen as generalizations of the familiar LU, QR and Cholesky factorizations, respectively. Many aspects for scalar polynomials and rational functions, as for example, poles and zeros, minimum degree coprime factorizations, normalized coprime factorizations or spectral factorization have nontrivial generalizations for polynomial and rational matrices.

2 The Toolbox

The DESCRIPTOR SYSTEMS Toolbox provides many useful functions for manipulating generalized linear systems with rational or polynomial transfer-function ma-

trices. This toolbox relies on the object oriented approach for control systems analysis and design provided within the standard CONTROL Toolbox of MATLAB. The basic approach to develop the DESCRIPTOR SYSTEMS Toolbox was to exploit the powerful matrix and system object manipulation features of MATLAB via flexible and functionally rich collection of *m*-functions, while simultaneously enforcing highly efficient and numerically sound computations via *mex*-functions to solve critical numerical problems. For the contents of the toolbox see Appendix B.

The high level *m*-functions of the toolbox call a relatively small set of powerful *mex*-functions implementing many of recently developed structure exploiting pencil reduction and manipulation algorithms. These functions are listed in the following table:

Name	Function
`kstair`	computation of several Kronecker-like forms [2, 16]
`gsminr`	minimal descriptor realization [14]
`gsystr`	generalized system similarity transformations
`gszero`	generalized system zeros and Kronecker structure [7]
`qzord`	real QZ-algorithm and stable/unstable separation of generalized eigenvalues [5]
`sysplace`	partial pole placement via Schur method [13]
`genleq`	generalized Sylvester and Lyapunov matrix equations [4, 12]

All these *mex*-functions are based on recently implemented FORTRAN codes from the LAPACK [1] and SLICOT [3] libraries. Note that two important computations provided by the LAPACK based *mex*-function `qzord`, namely the real QZ algorithm and the reordering of generalized real Schur forms (both not available in MATLAB) are the basic tools to solve efficiently standard and generalized Riccati equations.

Several of implemented high-level descriptor systems *m*-functions can be seen as extensions of equivalent functions provided in the standard CONTROL Toolbox of MATLAB. These are: `spole` to compute poles, `seig` to compute generalized eigenvalues, `szero` to compute system zeros, `dsminreal` to compute irreducible or minimal realizations, `sconj` to determine the conjugate descriptor system, or `sinv` to invert a descriptor system. The functionality of these functions is however much richer than that of MATLAB counterparts. For example, `spole`/`seig` computes not only the poles/eigenvalues (finite and infinite), but also the complete Kronecker structure of a given (possibly non-square) pencil $A-\lambda E$. This is also the case for the rich functionality provided by the function `szero` (see the Appendix A). Practically, this function can be seen as an universal analysis tool for standard or generalized time-invariant systems.

The function `sinv` is another example for a rich functionality. For an invertible system (i.e., with an invertible rational TFM) this function determines a descriptor system which represents the descriptor realization of the inverse TFM. Note that for standard systems, this operation is possible only if the feedthrough matrix D is invertible. If the TFM is not invertible as rational matrix or if it is non-square, `sinv` determines a descriptor realization whose TFM is a generalized inverse of the system TFM. Stable generalized inverses can be optionally determined if exist. For stabilization, a generalized pole assignment technique is employed. A lower level function `kronscf`, called by `sinv`, can be used to compute several Kronecker-like staircase forms together with the complete Kronecker structure of a linear pencil.

The DESCRIPTOR SYSTEMS Toolbox also provides useful conversion functions: `tm2dss` to compute a minimal realization of a rational/polynomial matrix [14], `dss2tm` and `dss2zpk` to evaluate the rational matrix corresponding to a descriptor representation [15], and `dss2ss` to convert a proper descriptor system to a standard system representation. A set of *m*-functions built around the *mex*-function `gsystr` are available for generalized state-space similarity transformations, as for example, for descriptor system scaling or various coordinate transformations (QR, RQ, SVD or SVD-like). For additive spectral decompositions of a given descriptor system (e.g., finite-infinite or stable-unstable) the function `specdec` is provided [6, 5].

The Toolbox contains two functions based on "last minute" methods for solving two important factorization problems of rational matrices in the most general setting: `iofac` to compute the inner-outer factorizations, and `nrcf`/`lrcf` to compute the normalized right/left coprime factorization. Based on `iofac`, the function `spinv` computes the pseudo-inverse of a rational matrix. All these functions are based on recent factorizations algorithms [8, 9].

Finally, two functions `gplace` and `gstab` are provided for generalized eigenvalue assignment [10] and stabilization, respectively. A set of five *m*-functions built around `genleq` are available to solve various generalized Sylvester and Lyapunov equations [4, 12].

3 Implementation aspects

The DESCRIPTOR SYSTEMS Toolbox supports all three basic system representations in the standard CONTROL Toolbox of MATLAB: descriptor state-space, rational

and pole/zero/gain representations. By function overloading, the same function performs (if appropriate) on all three representations. Automatic model conversions are performed when necessary and the results are provided in accordance with the original system representation. All system level functions of the toolbox are available for both continuous-time as well as discrete-time descriptor systems. For some computations (e.g., system zeros, minimal realization) separate algorithms are used according to the type of system: standard ($E = I$) or descriptor.

All used SLICOT programs are based exclusively on numerically reliable and efficient (complexity $O(n^3)$) algorithms. Most staircase reduction algorithms use the LAPACK-style incremental rank estimators in combination with QR-factorization with column pivoting. All algorithms are based on real computations. The implementations of all functions exploit the best of both MATLAB and FORTRAN programming, by trying to balance the matrix manipulation power of MATLAB with the intrinsic high efficiency of carefully implemented structure exploiting FORTRAN codes available in LAPACK and SLICOT. This approach is in our opinion a very promising way to address the development of future computer aided control engineering environments.

The Toolbox works with MATLAB 5.3 and of CONTROL Toolbox 4.2 under Windows NT/95/98. The *mex*-functions have been produced using the Compaq (former Digital) Visual Fortran V 6.0. The only used Fortran 90 features are code parts for memory allocation which however can be easily replaced by equivalent calls to memory allocation routines provided by MATLAB. Therefore it is expected that no difficulties arise when porting the *mex*-functions to Unix based systems, provided a suitable Fortran 90 compiler is available.

4 Examples

Consider the non-proper transfer-function matrix

$$G(\lambda) = \begin{bmatrix} \lambda^2 & \dfrac{\lambda}{\lambda - 1} \\ 0 & \dfrac{1}{\lambda} \end{bmatrix}$$

A minimal order descriptor system representation of $G(\lambda)$ is given by

$$\left[\begin{array}{c|c} A - \lambda E & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{ccccc|cc} -\lambda & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-\lambda & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & -\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ \hline 0 & -1 & -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

A continuous-time ($\lambda = s$) *Transfer Function* object in MATLAB can be defined with the command

```
gtf = tf({ [1 0 0], [1 0];  0,  [1]  },...
         {    1,    [1 -1]; 1, [1 0] })

Transfer function from input 1 to output...
   #1:  s^2
   #2:  0

Transfer function from input 2 to output...
          s
   #1:  -----
        s - 1
        1
   #2:  -
        s
```

The corresponding *Descriptor System* object can be defined with the command

```
gds = dss(a,b,c,d,e)

a =
             x1    x2    x3    x4    x5
       x1     0     1     0     0     0
       x2     0     1     0     0     0
       x3     0     0     1     0     0
       x4     0     0     0     1     0
       x5     0     0     0     0     1

b =
             u1    u2
       x1     0     0
       x2     0    -1
       x3     0     0
       x4     0     0
       x5     1     0

c =
             x1    x2    x3    x4    x5
       y1     0    -1    -1     0     0
       y2     1    -1     0     0     0

d =
             u1    u2
       y1     0     1
       y2     0     0

e =
             x1    x2    x3    x4    x5
       x1     1     0     0     0     0
       x2     0     1     0     0     0
       x3     0     0     0     1     0
       x4     0     0     0     0     1
       x5     0     0     0     0     0
Continuous-time system.
```

Note that E being singular, this command is not allowed in the standard MATLAB.

The poles and zeros of $G(s)$ can be computed as

```
poles = spole(gds)

poles =
          0
          1
        Inf
        Inf

zeros = szero(gds)

zeros =
        1.0000
        0.0000
       -0.0000
           Inf
```

The inverse of $G(s)$ is

$$G^{-1}(s) = \begin{bmatrix} \frac{1}{s^2} & -\frac{1}{s-1} \\ 0 & s \end{bmatrix}$$

and can be computed with either of commands

`dss2tm(sinv(gds))` or `sinv(gtf)`

obtaining

```
Transfer function from input 1 to output...
                      1
  #1:  --------------------------------
       s^2 - 1.771e-017 s - 7.025e-017
  #2:  0

Transfer function from input 2 to output...
        -1
  #1:  -----
       s - 1
  #2:  s - 7.882e-016
```

The conjugate of the transfer-function matrix $G(s)$ is

$$G^{\sim}(s) := G^T(-s) = \begin{bmatrix} s^2 & 0 \\ \frac{s}{s+1} & -\frac{1}{s} \end{bmatrix}$$

and can be computed with either of commands

`dss2tm(sconj(gds))` or `sconj(gtf)` or `gtf'`

resulting

```
Transfer function from input 1 to output...
  #1:  s^2
       s - 4.441e-016
  #2:  --------------
            s + 1

Transfer function from input 2 to output...
  #1:  0
       -1
  #2:  --
       s
```

Consider now the same transfer-function matrix, but in discrete-time case

$$G(z) = \begin{bmatrix} z^2 & \frac{z}{z-1} \\ 0 & \frac{1}{z} \end{bmatrix}.$$

Its conjugate

$$G^{\sim}(z) := G^T(1/z) = \begin{bmatrix} \frac{1}{z^2} & 0 \\ \frac{-1}{z-1} & z \end{bmatrix}$$

can be computed as follows

```
gtfd = tf({ [1 0 0], [1 0]; 0,  [1]  }, ...
          {    1,   [1 -1]; 1, [1 0] }, 1 );
gsysd = tm2dss(gtfd);
dss2tm(sconj(gsysd))

Transfer function from input 1 to output...
                      1
  #1:  --------------------------------
       z^2 - 2.215e-016 z - 9.415e-017

        -1
  #2:  -----
       z - 1

Transfer function from input 2 to output...
  #1:  0

  #2:  z + 1.11e-016

Sampling time: 1
```

The computation of the inner-outer factorization of the TFM

$$G(s) = \begin{bmatrix} \frac{s-1}{(s+2)} & 0 & \frac{s-1}{s+2} \\ \frac{s}{(s+2)} & \frac{s-2}{(s+1)^2} & \frac{s^2+2s-2}{(s+1)(s+2)} \\ \frac{1}{(s+2)} & \frac{s-2}{(s+1)^2} & \frac{2s-1}{(s+1)(s+2)} \end{bmatrix}$$

raises difficulties because $G(s)$ has a zero at ∞ and has normal rank 2. Thus, none of known standard methods are applicable to compute this factorization. Using the recently developed inner-outer factorization algorithm based on pencil manipulation technique [11] we computed the factors with the command

```
[Gi,Go] = iofac(G)
```

obtaining

$$G_i(s) = \begin{bmatrix} \dfrac{\sqrt{2}(s-1)}{(s+1)(s+2)} & -\dfrac{\sqrt{6}(s-1)}{3(s+2)} \\ \dfrac{\sqrt{2}(s-1)}{2(s+1)} & -\dfrac{\sqrt{6}}{6} \\ \dfrac{\sqrt{2}s(s-1)}{2(s+1)(s+2)} & \dfrac{\sqrt{6}(s-4)}{6(s+2)} \end{bmatrix},$$

$$G_o(s) = \begin{bmatrix} \dfrac{\sqrt{2}(s+1)}{2(s+2)} & \dfrac{\sqrt{2}}{s+1} & \dfrac{\sqrt{2}(s+3)}{2(s+2)} \\ -\dfrac{\sqrt{6}(s+1)}{2(s+2)} & \dfrac{\sqrt{6}}{(s+1)^2} & -\dfrac{\sqrt{6}(s^2+2s-1)}{2(s+1)(s+2)} \end{bmatrix},$$

where G_i is *inner* satisfying $G^{\sim}(s)G(s) = I$ and G_o is *outer* being a surjective (i.e., full row rank), stable and minimum-phase TFM.

5 Conclusions

In this paper we presented a recently developed DESCRIPTOR SYSTEMS Toolbox for MATLAB. The primary goal for developing this toolbox was to provide tools for manipulating generalized state-space systems and rational/polynomial matrices. However, the applicability of the DESCRIPTOR SYSTEMS Toolbox is certainly much wider than the intended original functionality.

One particular aspect of this toolbox is that almost all its functions are based on matrix pencils manipulations. Examples where pencil techniques play a crucial role are the computation of controllability/observability staircase forms, determination of infinite poles/zeros structure and minimal indices, performing additive spectral separations like finite-infinite or stable-unstable splitting, or conversions between descriptor representations and rational matrices. Note that, virtually, matrix pencil techniques can be used for every linear system analysis and design computation! Thus, the available pencil reduction tools are also useful to solve several difficult **standard** control problems, as for example, the solution of discrete-time Riccati equations, solution of non-standard H_∞ problems, computation of infinite zeros, determination of the Kronecker structure of the system pencil, computation of inner-outer and spectral factorizations.

One interesting aspect worth mentioning is that in solving many control problems the use of pencil methods make non-standard Riccati equations based approaches obsolete! Such equations appear in solving non-standard H_∞ problems or in computing several rational matrix factorizations (e.g., inner-outer, normalized coprime, spectral). By using deflating techniques based on appropriate pencil reductions, the solution of non-standard Riccati equations in all such problems can be avoided by solving instead reduced order standard Riccati equations (again by using pencil techniques). Notable examples are the computation of inner-outer and normalized coprime factorizations using the algorithms proposed in [11, 9].

In the future, several extensions and enhancements of the toolbox are planned, as for example, implementation of algorithms for the minimum-degree coprime factorization [10], J-lossless-outer factorization and canonical J-inner-outer factorization. Further, a new implementation of the algorithm of [16] to compute Kronecker-like forms is envisaged based exclusively on incremental rank estimation techniques.

References

[1] 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.

[2] T. Beelen and P. Van Dooren. An improved algorithm for the computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. & Appl.*, 105:9–65, 1988.

[3] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga. SLICOT – a subroutine library in systems and control theory. In B. N. Datta, editor, *Applied and Computational Control, Signals and Circuits*, vol. 1, pp. 499–539, Birkhäuser, 1999.

[4] 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.

[5] B. Kågström and P. Poromaa. Computing eigenspaces with specified eigenvalues of a regular matrix pair (A,B) and condition estimation: theory, algorithms and software. *Numer. Algorithms*, 12:369–407, 1996.

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

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

[8] C. Oară and A. Varga. Solution to the general inner-outer and spectral factorization problems. *Proc. CDC'98, Tampa, FL*, pp. 2774–2779, 1998.

[9] C. Oară and A. Varga. The general inner-outer factorization problem for discrete-time systems. *Proc. ECC'99, Karlsruhe, Germany*, 1999.

[10] C. Oară and A. Varga. Minimal degree coprime factorization of rational matrices. *SIAM J. Matrix Anal. Appl.*, 21:245–278, 1999.

[11] C. Oară and A. Varga. Computation of general inner-outer and spectral factorizations. *IEEE Trans. Autom. Control*, 45, 2000 (*to appear*).

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

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

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

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

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

A The function SZERO

```
SZERO  System transmision zeros of LTI systems.

Z = SZERO(SYS) computes the transmission zeros
of the LTI system SYS. (Z is a column vector).

Z = SZERO(SYS,TOL) uses tolerance TOL for rank
determinations.

[Z,MI] = SZERO(SYS) returns also information
on multiplicities of infinite zeros as follows:
there are MI(k) multiplicity k infinite zeros.

[Z,MI,KRONS] = SZERO(SYS) returns additionally
the normal rank, and the singular and infinite
Kronecker structure of the system pencil in
the MATLAB structure KRONS as follows:
KRONS.kronr   - right Kronecker indices
KRONS.infe    - elementary infinite blocks
KRONS.kronl   - left Kronecker indices
KRONS.nr      - normal rank of the system pencil

Z = SZERO(A,E,B,C,D)  works directly on the
state space matrices and returns the
transmission zeros of the state-space system:
  .
 Ex = Ax + Bu     or  Ex[n+1] = Ax[n] + Bu[n]
  y = Cx + Du           y[n]  = Cx[n] + Du[n]

Z = SZERO(A,E,B,C,D,TOL) uses tolerance TOL
for rank determinations.

Z = SZERO(A,E) or Z = SZERO(A,E,TOL) works on
the matrix pencil A - lambda E, where A and E
are arbitrary matrices of compatible
dimensions.

Z = SZERO(A) works on the matrix pencil
A - lambda I.
```

B Contents of Descriptor Systems Toolbox

```
Descriptor Systems Toolbox.
Version 0.6   04-August-1999

Model Conversions
dss2zpk - Conversion to zero/pole/gain form
dss2tm  - Conversion to transfer function matrix
tm2dss  - Minimal descriptor realization of a
          transfer function matrix

Operations on rational matrices/LTI systems
sconj   - Conjugate transpose
sinv    - Inverse/generalized inverse
spinv   - Pseudo-inverse
nrcf    - Normalized right coprime factorization
nlcf    - Normalized left coprime factorization
iofac   - Inner-outer factorization
specdec - Additive spectral decomposition
          (finite-infinite or stable-unstable)

Model dynamics analysis
spole   - System poles and Kronecker structure
szero   - System zeros and Kronecker structure
seig    - Eigenvalues and Kronecker structure
kronscf - Kronecker-like staircase forms

State-space transformations
dsqr,dsrq - Descriptor QR/RQ forms
dssvd     - Descriptor SVD and SVD-like forms
dsctrbf   - Descriptor controllability forms
dsobsvf   - Descriptor observability forms
dsbal     - Scaling of descriptor realizations
dsminreal - Minimal realization
dss2ss    - Conversion to standard state space

Design tools
gplace    - Generalized eigenvalue assignment
gstab     - Generalized stabilization

Generalized linear matrix equation solvers
glyap   - Continuous Lyapunov equations
gplyap  - Discrete Lyapunov equations
gplyap  - Positive continuous Lyapunov equations
gpdlyap - Positive discrete Lyapunov equations
gsylv   - Generalized Sylvester equations

Basic mex-functions
gsystr   - Generalized similarity transformations
gsminr   - Generalized system minimal realization
gszero   - System zeros and Kronecker structure
qzord    - Real QZ-algorithm and stable/unstable
           ordering of generalized eigenvalues
kstair   - Staircase reduction of a pencil
sysplace - Pole placement via Schur method
genleq   - Generalized matrix linear equations
```