Descriptor System Tools (**DSTOOLS**)

User's Guide

Andreas Varga*

September 30, 2018

Abstract

The Descriptor System Tools (**DSTOOLS**) is a collection of MATLAB functions for the operation on and manipulation of rational transfer function matrices via their descriptor system realizations. The **DSTOOLS** collection relies on the Control System Toolbox and several mex-functions based on the Systems and Control Library SLICOT. Many of the implemented functions are based on the computational procedures described in Chapter 10 of the book: "A. Varga, Solving Fault Diagnosis Problems – Linear Synthesis Techniques, Springer, 2017". This document is the User's Guide for the version V0.74 of **DSTOOLS**. First, we present the mathematical background on rational matrices and descriptor systems. Then, we give in-depth information on the command syntax of the main computational functions. Several examples illustrate the use of the main functions of **DSTOOLS**.

*Andreas Varga lives in Gilching, Germany. *E-mail address*: varga.andreas@gmail.com, *URL*: https://sites.google.com/view/andreasvarga/home

Contents

Notations and Symbols

$\emptyset$	empty set
$\mathbb{C}$	field of complex numbers
$\mathbb{R}$	field of real numbers
$\mathbb{C}_s$	stability domain (i.e., open left complex half-plane in continuous-time or open unit disk centered in the origin in discrete-time)
$\partial\mathbb{C}_s$	boundary of stability domain (i.e., extended imaginary axis with infinity included in continuous-time, or unit circle centered in the origin in discrete-time)
$\overline{\mathbb{C}}_s$	closure of $\mathbb{C}_s$: $\overline{\mathbb{C}}_s = \mathbb{C}_s \cup \partial\mathbb{C}_s$
$\mathbb{C}_u$	open instability domain: $\mathbb{C}_u := \mathbb{C} \setminus \overline{\mathbb{C}}_s$
$\overline{\mathbb{C}}_u$	closure of $\mathbb{C}_u$: $\overline{\mathbb{C}}_u := \mathbb{C}_u \cup \partial\mathbb{C}_s$
$\mathbb{C}_g$	"good" domain of $\mathbb{C}$
$\mathbb{C}_b$	"bad" domain of $\mathbb{C}$: $\mathbb{C}_b = \mathbb{C} \setminus \mathbb{C}_g$
s	complex frequency variable in the Laplace transform: $s = \sigma + \mathrm{i}\omega$
z	complex frequency variable in the Z-transform: $z = \mathrm{e}^{sT}$, T – sampling time
λ	complex frequency variable: $\lambda = s$ in continuous-time or $\lambda = z$ in discrete-time
$\bar{\lambda}$	complex conjugate of the complex number λ
$\mathbb{R}(\lambda)$	field of real rational functions in indeterminate λ
$\mathbb{R}(\lambda)$	set of rational matrices in indeterminate λ with real coefficients and unspecified dimensions
$\mathbb{R}(\lambda)^{p\times m}$	set of $p \times m$ rational matrices in indeterminate λ with real coefficients
$\mathbb{R}[\lambda]$	ring of real rational polynomials in indeterminate λ
$\mathbb{R}[\lambda]$	set of polynomial matrices in indeterminate λ with real coefficients and unspecified dimensions
$\mathbb{R}[\lambda]^{p\times m}$	set of $p \times m$ polynomial matrices in indeterminate λ with real coefficients
$\delta(G(\lambda))$	McMillan degree of the rational matrix $G(\lambda)$
$G^{\sim}(\lambda)$	Conjugate of $G(\lambda) \in \mathbb{R}(\lambda)$: $G^{\sim}(s) = G^T(-s)$ in continuous-time and $G^{\sim}(z) = G^T(1/z)$ in discrete-time
ℓ_2	Banach-space of square-summable sequences
$\mathcal{L}_2$	Lebesgue-space of square-integrable functions
$\mathcal{H}_2$	Hardy-space of square-integrable complex-valued functions analytic in $\mathbb{C}_u$
$\mathcal{L}_\infty$	Space of complex-valued functions bounded and analytic in $\partial\mathbb{C}_s$
$\mathcal{H}_\infty$	Hardy-space of complex-valued functions bounded and analytic in $\mathbb{C}_u$
$\|G\|_2$	$\mathcal{H}_2$- or $\mathcal{L}_2$-norm of the transfer function matrix $G(\lambda)$
$\|G\|_\infty$	$\mathcal{H}_\infty$- or $\mathcal{L}_\infty$-norm of the transfer function matrix $G(\lambda)$
$\|G\|_{\infty/2}$	either the $\mathcal{H}_\infty$- or $\mathcal{H}_2$-norm of the transfer function matrix $G(\lambda)$
$\|G\|_H$	Hankel norm of the transfer function matrix $G(\lambda)$
$\delta_\nu(G_1, G_2)$	ν-gap distance between the transfer function matrices $G_1(\lambda)$ and $G_2(\lambda)$
$\mathrm{col}_i(M)$	the i-th column of the matrix M
$\mathrm{row}_i(M)$	the i-th row of the matrix M
M^T	transpose of the matrix M
M^P	pertranspose of the matrix M
M^{-1}	inverse of the matrix M
M^{-T}	transpose of the inverse matrix M^{-1}

M^{-L}	left inverse of the matrix M (i.e., $M^{-L}M = I$)
M^{-R}	right inverse of the matrix M (i.e., $MM^{-R} = I$)
$M^{\dagger}$	Moore-Penrose pseudo-inverse of the matrix M
$\overline{\sigma}(M)$	largest singular value of the matrix M
$\underline{\sigma}(M)$	least singular value of the matrix M
$\mathcal{N}(M)$	kernel (or right nullspace) of the matrix M
$\mathcal{N}_L(G(\lambda))$	left kernel (or left nullspace) of $G(\lambda) \in \mathbb{R}(\lambda)$
$\mathcal{N}_R(G(\lambda))$	right kernel (or right nullspace) of $G(\lambda) \in \mathbb{R}(\lambda)$
$\mathcal{R}(M)$	range (or image space) of the matrix M
$\Lambda(A)$	set of eigenvalues of the matrix A
$\Lambda(A, E)$	set of generalized eigenvalues of the pair (A, E)
$\Lambda(A - \lambda E)$	set of eigenvalues of the pencil $A - \lambda E$
$\mathbf{u}$	unit roundoff of the floating-point representation
$\mathcal{O}(\epsilon)$	quantity of order of ϵ
I_n or I	identity matrix of order n or of an order resulting from context
e_i	the i-th column of the (known size) identity matrix
$0_{m\times n}$ or 0	zero matrix of size $m \times n$ or of a size resulting from context
$\operatorname{span} M$	span (or linear hull) of the columns of the matrix M

Acronyms

GCARE	Generalized continuous-time algebraic Riccati equation
GDARE	Generalized discrete-time algebraic Riccati equation
GRSD	Generalized real Schur decomposition
GRSF	Generalized real Schur form
LCF	Left coprime factorization
LDP	Least distance problem
LTI	Linear time-invariant
MIMO	Multiple-input multiple-output
RCF	Right coprime factorization
RSF	Real Schur form
SISO	Single-input single-output
SVD	Singular value decomposition
TFM	Transfer function matrix

1 Introduction

The Descriptor System Tools (**DSTOOLS**) is a collection of MATLAB functions for the operation on and manipulation of rational transfer function matrices via their descriptor system realizations. The initial version V0.5 of **DSTOOLS** covers the main computations encountered in the synthesis approaches of linear residual generation filters for continuous- or discrete-time linear systems, described in the Chapter 10 of the author's book [59]:

> Andreas Varga, *Solving Fault Diagnosis Problems - Linear Synthesis Techniques*, vol. 84 of Studies in Systems, Decision and Control, Springer International Publishing, xxviii+394, 2017.

The functions of the **DSTOOLS** collection rely on the *Control System Toolbox* [25] and several mex-functions based on the Systems and Control Library SLICOT [4]. The current release of **DSTOOLS** is version V0.74, dated July 30, 2019. **DSTOOLS** is distributed as a free software via the Bitbucket repository.[1] The codes have been developed under MATLAB 2015b and have been also tested with MATLAB 2016a through 2019a. To use the functions of **DSTOOLS**, the *Control System Toolbox* must be installed in MATLAB running under 64-bit Windows 7, 8, 8.1 or 10.

This document describes version V0.74 of the **DSTOOLS** collection. It will be continuously extended in parallel with the implementation of new functions. The book [59] represents an important complementary documentation for the **DSTOOLS** collection: it describes the mathematical background on rational matrices and descriptor systems, and gives detailed descriptions of many of the underlying procedures. Additionally, the M-files of the functions are self-documenting and a detailed documentation can be obtained online by typing help with the M-file name. Please cite the current version of **DSTOOLS** as follows:

> A. Varga. DSTOOLS – The Descriptor System Tools for MATLAB, 2018.
> `https://sites.google.com/site/andreasvargacontact/home/software/dstools`.

[1] `https://bitbucket.org/DSVarga/dstools`

2 Background Material on Generalized System Representations

In this section we give background information on two system representations of linear time-invariant systems, namely the input-output representation via rational transfer function matrices and the generalized state-space representation, also known as descriptor system representation. Since each rational matrix can be interpreted as the transfer function matrix of a descriptor system, the manipulation of rational matrices can be alternatively performed via their descriptor representations, using numerically reliable computational algorithms.

The treatment in depth of most of concepts was not possible in the restricted size of this guide. The equivalence theory of linear matrix pencils is covered in [13]. The material on rational matrices is covered in several textbooks, of which we mention only the two widely cited books of Kailath [22] and Vidyasagar [63]. Linear descriptor systems (also known in the literature as linear differential-algebraic-equations-based systems or generalized state-space systems or singular systems), are discussed, to different depths and with different focus, in several books [5, 7, 24, 9].

2.1 Rational Transfer Function Matrices

Transfer functions are used to describe the input-output behaviour of *single-input single-output* (SISO) *linear time-invariant* (LTI) systems by relating the input and output variables via a gain depending on a frequency variable. For a SISO system with input $u(t)$ and output $y(t)$ depending on the continuous time variable t, let $\mathbf{u}(s) := \mathcal{L}(u(t))$ and $\mathbf{y}(s) := \mathcal{L}(y(t))$ denote the Laplace transformed input and output, respectively. Then, the transfer function of the continuous-time LTI system is defined as

$$g(s) := \frac{\mathbf{y}(s)}{\mathbf{u}(s)}$$

and relates the input and output in the form

$$\mathbf{y}(s) = g(s)\mathbf{u}(s)\,.$$

The complex variable $s = \sigma + j\omega$, has for $\sigma = 0$ the interpretation of a complex frequency. If the time variable has a discrete variation with equally spaced values with increments given by a sampling-period T, then the transfer function of the discrete-time system is defined using the $\mathcal{Z}$-transforms of the input and output variables $\mathbf{u}(z) := \mathcal{Z}(u(t))$ and $\mathbf{y}(z) := \mathcal{Z}(y(t))$, respectively, as

$$g(z) := \frac{\mathbf{y}(z)}{\mathbf{u}(z)}$$

and relates the input and output in the form

$$\mathbf{y}(z) = g(z)\mathbf{u}(z)\,.$$

The complex variable z is related to the complex variable s as $z = e^{sT}$. We will use the variable λ to denote either the s or z complex variables, depending on the context, continuous- or discrete-time, respectively. Throughout this guide, bolded variables as $\mathbf{u}(\lambda)$ and $\mathbf{y}(\lambda)$ will be used to denote either the Laplace- or $\mathcal{Z}$-transformed quantities of the corresponding time-variables $u(t)$

and $y(t)$. Furthermore, we will restrict our discussion to rational transfer functions $g(\lambda)$ which can be expressed as a ratio of two polynomials with real coefficients

$$g(\lambda) = \frac{\alpha(\lambda)}{\beta(\lambda)} = \frac{a_m\lambda^m + a_{m-1}\lambda^{m-1} + \cdots + a_1\lambda + a_0}{b_n\lambda^n + b_{n-1}\lambda^{n-1} + \cdots + b_1\lambda + b_0}\,, \tag{1}$$

with $a_m \neq 0$ and $b_n \neq 0$. Thus, $g(\lambda) \in \mathbb{R}(\lambda)$, where $\mathbb{R}(\lambda)$ is the field of real rational functions.

Transfer function matrices are used to describe the input-output behaviour of *multi-input multi-output* (MIMO) LTI systems by relating the input and output variables via a matrix of gains depending on a frequency variable. Consider a MIMO system with m inputs $u_1(t)$, ..., $u_m(t)$, which form the m-dimensional input vector $u(t) = [\,u_1(t), \ldots, u_m(t)\,]^T$, and p outputs $y_1(t)$, ..., $y_p(t)$, which form the p-dimensional output vector $y(t) = [\,y_1(t), \ldots, y_p(t)\,]^T$. For a continuous dependence of $u(t)$ and $y(t)$ on the time variable t, let $\mathbf{u}(s)$ and $\mathbf{y}(s)$ be the Laplace-transformed input and output vectors, respectively, while in the case of a discrete dependence on t, we denote $\mathbf{u}(z)$ and $\mathbf{y}(z)$ the $\mathcal{Z}$-transformed input and output vectors, respectively. We denote with λ the frequency variable, which is either s or z, depending on the nature of the time variation, continuous or discrete, respectively. Let $G(\lambda)$ be the $p \times m$ *transfer function matrix* (TFM) defined as

$$G(\lambda) = \begin{bmatrix} g_{11}(\lambda) & \cdots & g_{1m}(\lambda) \\ \vdots & \ddots & \vdots \\ g_{p1}(\lambda) & \cdots & g_{pm}(\lambda) \end{bmatrix},$$

which relates relates the m-dimensional input vector u to the p-dimensional output vector y in the form

$$\mathbf{y}(\lambda) = G(\lambda)\mathbf{u}(\lambda)\,.$$

The element $g_{ij}(\lambda)$ describes the contribution of the j-th input $u_j(t)$ to the i-th output $y_i(t)$. We assume that each matrix entry $g_{ij}(\lambda) \in \mathbb{R}(\lambda)$ and thus it can be expressed as ratio of two polynomials $\alpha_{ij}(\lambda)$ and $\beta_{ij}(\lambda)$ with real coefficients as $g_{ij}(\lambda) = \alpha_{ij}(\lambda)/\beta_{ij}(\lambda)$ of the form (1).

Each TFM $G(\lambda)$ belongs to the set of rational matrices with real coefficients, thus having elements in the field of real rational functions $\mathbb{R}(\lambda)$. Polynomial matrices, having elements in the ring of polynomials with real coefficients $\mathbb{R}[\lambda]$, can be assimilated in a natural way with special rational matrices with all elements having 1 as denominators. Let $\mathbb{R}(\lambda)^{p\times m}$ and $\mathbb{R}[\lambda]^{p\times m}$ denote the sets of $p \times m$ rational and polynomial matrices with real coefficients, respectively. To simplify the notation, we will also use $G(\lambda) \in \mathbb{R}(\lambda)$ or $G(\lambda) \in \mathbb{R}[\lambda]$ if the dimensions of $G(\lambda)$ are not relevant or are clear from the context.

A rational matrix $G(\lambda) \in \mathbb{R}(\lambda)$ is called *proper* if $\lim_{\lambda\to\infty} G(\lambda) = D$, with D having a finite norm. Otherwise, $G(\lambda)$ is called *improper*. If $D = 0$, then $G(\lambda)$ is *strictly proper*. An invertible $G(\lambda)$ is *biproper* if both $G(\lambda)$ and $G^{-1}(\lambda)$ are proper. A polynomial matrix $U(\lambda) \in \mathbb{R}[\lambda]$ is called *unimodular* if is invertible and its inverse $U^{-1}(\lambda) \in \mathbb{R}[\lambda]$ (i.e., is a polynomial matrix). The determinant of a unimodular matrix is therefore a constant.

The degree of a rational matrix $G(\lambda)$, also known as the McMillan degree, is defined in Section 2.5. We only give here the definition of the degree of a rational vector $v(\lambda)$. For this, we express first $v(\lambda)$ in the form $v(\lambda) = \widetilde{v}(\lambda)/d(\lambda)$, where $d(\lambda)$ is the monic least common multiple of all denominator polynomials of the elements of $v(\lambda)$ and $\widetilde{v}(\lambda)$ is the corresponding polynomial vector $\widetilde{v}(\lambda) := d(\lambda)v(\lambda)$. Then, $\deg\, v(\lambda) = \max(\deg\, \widetilde{v}(\lambda), \deg\, v(\lambda))$.

2.2 Descriptor Systems

A descriptor system is a generalized state-space representation of the form

$$\begin{array}{rcl} E\lambda x(t) & = & Ax(t) + Bu(t), \\ y(t) & = & Cx(t) + Du(t), \end{array} \tag{2}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the input vector, and $y(t) \in \mathbb{R}^p$ is the output vector, and where λ is either the differential operator $\lambda x(t) = \frac{\mathrm{d}}{\mathrm{d}t}x(t)$ for a continuous-time system or the advance operator $\lambda x(t) = x(t+1)$ for a discrete-time system. In all what follows, we assume E is square and possibly singular, and the pencil $A - \lambda E$ is regular (i.e., $\det(A - \lambda E) \not\equiv 0$). If $E = I_n$, we call the representation (2) a *standard state-space system*. The corresponding input-output representation of the descriptor system (2) is

$$\mathbf{y}(\lambda) = G(\lambda)\mathbf{u}(\lambda)\,, \tag{3}$$

where, depending on the system type, $\lambda = s$, the complex variable in the Laplace transform for a continuous-time system, or $\lambda = z$, the complex variable in the $\mathcal{Z}$-transform for a discrete-time system, $\mathbf{y}(\lambda)$ and $\mathbf{u}(\lambda)$ are the Laplace- or $\mathcal{Z}$-transformed output and input vectors, respectively, and $G(\lambda)$ is the rational TFM of the system, defined as

$$G(\lambda) = C(\lambda E - A)^{-1}B + D\,. \tag{4}$$

We alternatively denote descriptor systems of the form (2) with the quadruple $(A-\lambda E, B, C, D)$ or a standard state-space system with (A, B, C, D) (if $E = I_n$), and use the notation

$$G(\lambda) := \left[\begin{array}{c|c} A - \lambda E & B \\ \hline C & D \end{array}\right], \tag{5}$$

to relate the TFM $G(\lambda)$ to a particular descriptor system realization as in (2).

It is well known that a descriptor system representation of the form (2) is the most general description for a linear time-invariant system. Continuous-time descriptor systems arise frequently from modelling interconnected systems containing algebraic loops or constrained mechanical systems which describe contact phenomena. Discrete-time descriptor representations are frequently used to model economic processes.

The manipulation of rational matrices can be easily performed via their descriptor representations. The main result which allows this is the following [62]:

Theorem 1. For any rational matrix $G(\lambda) \in \mathbb{R}(\lambda)^{p\times m}$, there exist $n \geq 0$ and the real matrices $E, A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times m}$, with $A - \lambda E$ regular, such that (4) holds.

The descriptor realization $(A - \lambda E, B, C, D)$ of a given rational matrix $G(\lambda)$ is not unique. For example, if U and V are invertible matrices of the size n of the square matrix E, then two descriptor realizations $(A - \lambda E, B, C, D)$ and $(\widetilde{A} - \lambda\widetilde{E}, \widetilde{B}, \widetilde{C}, \widetilde{D})$ related by a *system similarity transformation* of the form

$$(\widetilde{A} - \lambda\widetilde{E}, \widetilde{B}, \widetilde{C}, \widetilde{D}) = (UAV - \lambda UEV, UB, CV, D)\,, \tag{6}$$

have the same TFM $G(\lambda)$. Moreover, among all possible realizations of a given $G(\lambda)$, with different sizes n, there exist realizations which have the least dimension. A descriptor realization

$(A-\lambda E, B, C, D)$ of the rational matrix $G(\lambda)$ is called *minimal* if the dimension n of the square matrices E and A is the least possible one. The minimal realization of a given $G(\lambda)$ is also not unique, since two minimal realizations related by a system similarity transformation as in (6) correspond to the same $G(\lambda)$.

A minimal descriptor system realization $(A-\lambda E, B, C, D)$ is characterized by the following five conditions [61].

Theorem 2. A descriptor system realization $(A-\lambda E, B, C, D)$ of order n is minimal if the following conditions are fulfilled:

$$
\begin{aligned}
&(i) \quad \operatorname{rank} \begin{bmatrix} A-\lambda E & B \end{bmatrix} = n, \quad \forall \lambda \in \mathbb{C}, \\
&(ii) \quad \operatorname{rank} \begin{bmatrix} E & B \end{bmatrix} = n, \\
&(iii) \quad \operatorname{rank} \begin{bmatrix} A-\lambda E \\ C \end{bmatrix} = n, \quad \forall \lambda \in \mathbb{C}, \\
&(iv) \quad \operatorname{rank} \begin{bmatrix} E \\ C \end{bmatrix} = n, \\
&(v) \quad A\mathcal{N}(E) \subseteq \mathcal{R}(E).
\end{aligned}
$$

The conditions (i) and (ii) are known as *finite* and *infinite controllability*, respectively. A system or, equivalently, the pair $(A-\lambda E, B)$, is called *finite controllable* if it fulfills (i), *infinite controllable* if it fulfills (ii), and *controllable* if it fulfills both (i) and (ii). Similarly, the conditions (iii) and (iv) are known as *finite* and *infinite observability*, respectively. A system or, equivalently, the pair $(A-\lambda E, C)$, is called *finite observable* if it fulfills (iii), *infinite observable* if it fulfills (iv), and *observable* if it fulfills both (iii) and (iv). The condition (v) expresses the absence of non-dynamic modes. A descriptor realization which satisfies only $(i)-(iv)$ is called *irreducible* (also weakly minimal). The numerical computation of irreducible realizations is addressed in [44] (see also [39] for alternative approaches).

2.3 Linear Matrix Pencils

Linear matrix pencils of the form $M-\lambda N$, where M and N are $m \times n$ matrices with elements in $\mathbb{C}$, play an important role in the theory of generalized LTI systems. In what follows we shortly review the equivalence theory of linear matrix pencils. For more details on this topic see [13].

The pencil $M-\lambda N$ is called *regular* if $m=n$ and $\det(M-\lambda N) \not\equiv 0$. Otherwise, the pencil is called *singular*. Two pencils $M-\lambda N$ and $\widetilde{M}-\lambda\widetilde{N}$ with $M, N, \widetilde{M}, \widetilde{N} \in \mathbb{C}^{m\times n}$ are *strictly equivalent* if there exist two invertible matrices $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ such that

$$
U(M-\lambda N)V = \widetilde{M}-\lambda\widetilde{N}. \tag{7}
$$

For a regular pencil, the strict equivalence leads to the (complex) *Weierstrass canonical form*, which is instrumental to characterize the dynamics of generalized systems.

Lemma 1. Let $M-\lambda N$ be an arbitrary regular pencil with $M, N \in \mathbb{C}^{n\times n}$. Then, there exist invertible matrices $U \in \mathbb{C}^{n\times n}$ and $V \in \mathbb{C}^{n\times n}$ such that

$$
U(M-\lambda N)V = \begin{bmatrix} J_f - \lambda I & 0 \\ 0 & I - \lambda J_\infty \end{bmatrix}, \tag{8}
$$

where J_f is in a (complex) Jordan canonical form

$$J_f = \operatorname{diag}\left(J_{s_1}(\lambda_1), J_{s_2}(\lambda_2), \ldots, J_{s_k}(\lambda_k)\right) , \tag{9}$$

with $J_{s_i}(\lambda_i)$ an elementary $s_i \times s_i$ Jordan block of the form

$$J_{s_i}(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}$$

and J_∞ is nilpotent and has the (nilpotent) Jordan form

$$J_\infty = \operatorname{diag}\left(J_{s_1^\infty}(0), J_{s_2^\infty}(0), \ldots, J_{s_h^\infty}(0)\right) . \tag{10}$$

The Weierstrass canonical form (8) exhibits the finite and infinite eigenvalues of the pencil $M - \lambda N$. The finite eigenvalues are λ_i, for $i = 1, \ldots, k$. Overall, by including all multiplicities, there are $n_f = \sum_{i=1}^{k} s_i$ *finite eigenvalues* and $n_\infty = \sum_{i=1}^{h} s_i^\infty$ *infinite eigenvalues*. Infinite eigenvalues with $s_i^\infty = 1$ are called *simple infinite eigenvalues*. We can also express the rank of N as

$$\operatorname{rank} N = n_f + \operatorname{rank} J_\infty = n_f + \sum_{i=1}^{h} (s_i^\infty - 1) = n_f + n_\infty - h = n - h.$$

If M and N are real matrices, then there exist real matrices U and V such that the pencil $U(M - \lambda N)V$ is in a *real Weierstrass canonical form*, where the only difference is that J_f is in a *real* Jordan form [20, Section 3.4]. In this form, the elementary real Jordan blocks correspond to pairs of complex conjugate eigenvalues.

If $M - \lambda N = A - \lambda I$ (e.g., the pole pencil for a standard state-space system), then all eigenvalues are finite and J_f in the Weierstrass form is simply the (real) Jordan form of A. The transformation matrices can be chosen such that $U = V^{-1}$.

The eigenvalue structure of a regular pencil $A - \lambda E$ is completely described by the Weierstrass canonical form (see Lemma 1). However, the computation of this canonical form involves the use of (potentially ill-conditioned) general invertible transformations, and therefore numerical reliability cannot be guaranteed. Fortunately, the computation of Weierstrass canonical form can be avoided in almost all computations, and alternative "less" condensed forms can be employed instead, which can be computed by exclusively employing orthogonal similarity transformations. The *generalized real Schur decomposition* (GRSD) of a matrix pair (A, E) reveals the eigenvalues of the regular pencil $A - \lambda E$, by determining the *generalized real Schur form* (GRSF) of the pair (A, E) (a quasi-triangular–triangular form) using orthogonal similarity transformations on the pencil $A - \lambda E$. The main theoretical result regarding the GRSD is the following theorem.

Theorem 3. Let $A - \lambda E$ be an $n \times n$ regular pencil, with A and E real matrices. Then, there exist orthogonal transformation matrices Q and Z such that

$$S - \lambda T := Q^T (A - \lambda E) Z = \begin{bmatrix} S_{11} & \cdots & S_{1k} \\ & \ddots & \vdots \\ 0 & & S_{kk} \end{bmatrix} - \lambda \begin{bmatrix} T_{11} & \cdots & T_{1k} \\ & \ddots & \vdots \\ 0 & & T_{kk} \end{bmatrix} , \tag{11}$$

where each diagonal subpencil $S_{ii} - \lambda T_{ii}$, for $i = 1, \ldots, k$, is either of dimension 1×1 in the case of a finite real or infinite eigenvalue of the pencil $A - \lambda E$ or of dimension 2×2, with T_{ii} upper triangular, in the case of a pair of finite complex conjugate eigenvalues of $A - \lambda E$.

The pair (S, T) in (11) is in a GRSF and the eigenvalues of $A - \lambda E$ (or the generalized eigenvalues of the pair (A, E)) are given by

$$\Lambda(A - \lambda E) = \bigcup_{i=1}^{k} \Lambda(S_{ii} - \lambda T_{ii}) \,.$$

If $E = I$, then we can always choose $Q = Z$, $T = I$ and S is the *real Schur form* (RSF) of A.

The order of eigenvalues (and thus of the associated pairs of diagonal blocks) of the reduced pencil $S - \lambda T$ is arbitrary. The reordering of the pairs of diagonal blocks (thus also of corresponding eigenvalues) can be done by interchanging two adjacent pairs of diagonal blocks of the GRSF. For the swapping of such two pairs of blocks orthogonal similarity transformations can be used. Thus, any arbitrary reordering of pairs of blocks (and thus of the corresponding eigenvalues) can be achieved in this way. An important application of this fact is the computation of orthogonal bases for the deflating subspaces of the pencil $A - \lambda E$ corresponding to a particular eigenvalue or a particular set of eigenvalues.

For a general (singular) pencil, the strict equivalence leads to the (complex) *Kronecker canonical form*, which is instrumental to characterize the zeros and singularities of a descriptor system.

Lemma 2. Let $M - \lambda N$ be an arbitrary pencil with $M, N \in \mathbb{C}^{m \times n}$. Then, there exist invertible matrices $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ such that

$$U(M - \lambda N)V = \begin{bmatrix} K_r(\lambda) & & \\ & K_{reg}(\lambda) & \\ & & K_l(\lambda) \end{bmatrix}, \tag{12}$$

where:

1) The full row rank pencil $K_r(\lambda)$ has the form

$$K_r(\lambda) = \operatorname{diag}\left(L_{\epsilon_1}(\lambda), L_{\epsilon_2}(\lambda), \cdots, L_{\epsilon_{\nu_r}}(\lambda)\right), \tag{13}$$

with $L_i(\lambda)$ $(i \geq 0)$ an $i \times (i+1)$ bidiagonal pencil of form

$$L_i(\lambda) = \begin{bmatrix} -\lambda & 1 & & \\ & \ddots & \ddots & \\ & & -\lambda & 1 \end{bmatrix}; \tag{14}$$

2) The regular pencil $K_{reg}(\lambda)$ is in a Weierstrass canonical form

$$K_{reg}(\lambda) = \begin{bmatrix} \widetilde{J}_f - \lambda I & \\ & I - \lambda \widetilde{J}_\infty \end{bmatrix}, \tag{15}$$

with $\widetilde{J}_f$ in a (complex) Jordan canonical form as in (9) and with $\widetilde{J}_\infty$ in a nilpotent Jordan form as in (10);

3) The full column rank $K_l(\lambda)$ has the form

$$K_l(\lambda) = \operatorname{diag}\left(L_{\eta_1}^T(\lambda), L_{\eta_2}^T(\lambda), \cdots, L_{\eta_{\nu_l}}^T(\lambda)\right). \tag{16}$$

As it is apparent from (12), the Kronecker canonical form exhibits the right and left singular structures of the pencil $M - \lambda N$ via the full row rank block $K_r(\lambda)$ and full column rank block $K_l(\lambda)$, respectively, and the eigenvalue structure via the regular pencil $K_{reg}(\lambda)$. The full row rank pencil $K_r(\lambda)$ is $n_r \times (n_r + \nu_r)$, where $n_r = \sum_{i=1}^{\nu_r} \epsilon_i$, the full column rank pencil $K_l(\lambda)$ is $(n_l + \nu_l) \times n_l$, where $n_l = \sum_{j=1}^{\nu_l} \eta_j$, while the regular pencil $K_{reg}(\lambda)$ is $n_{reg} \times n_{reg}$, with $n_{reg} = \tilde{n}_f + \tilde{n}_\infty$, where $\tilde{n}_f$ is the number of finite eigenvalues in $\Lambda(\widetilde{J}_f)$ and $\tilde{n}_\infty$ is the number of infinite eigenvalues in $\Lambda(I - \lambda\widetilde{J}_\infty)$ (or equivalently the number of null eigenvalues in $\Lambda(\widetilde{J}_\infty)$). The $\epsilon_i \times (\epsilon_i + 1)$ blocks $L_{\epsilon_i}(\lambda)$ with $\epsilon_i \geq 0$ are the right elementary Kronecker blocks, and ϵ_i, for $i = 1, \ldots, \nu_r$, are called the *right Kronecker indices*. The $(\eta_i + 1) \times \eta_i$ blocks $L_{\eta_i}^T(\lambda)$ with $\eta_i \geq 0$ are the left elementary Kronecker blocks, and η_i, for $i = 1, \ldots, \nu_l$, are called the *left Kronecker indices*. The normal rank r of the pencil $M - \lambda N$ results as

$$r := \operatorname{rank}(M - \lambda N) = n_r + \tilde{n}_f + \tilde{n}_\infty + n_l.$$

If $M - \lambda N$ is regular, then there are no left- and right-Kronecker structures and the Kronecker canonical form is simply the Weierstrass canonical form.

Remark 1. By additional column permutations of the block $K_r(\lambda)$ and row permutations of the block $K_l(\lambda)$ (which can be included in the left and right transformations matrices U and V) we can bring these blocks to the alternative forms

$$K_r(\lambda) = \begin{bmatrix} B_r & A_r - \lambda I_{n_r} \end{bmatrix}, \qquad K_l(\lambda) = \begin{bmatrix} A_l - \lambda I_{n_l} \\ C_l \end{bmatrix}, \tag{17}$$

where the pair (A_r, B_r) is in a Brunovsky controllable form

$$A_r = \begin{bmatrix} A_{r,1} & & & \\ & A_{r,2} & & \\ & & \ddots & \\ & & & A_{r,\nu_r} \end{bmatrix}, \qquad B_r = \begin{bmatrix} b_{r,1} & & & \\ & b_{r,2} & & \\ & & \ddots & \\ & & & b_{r,\nu_r} \end{bmatrix},$$

with $A_{r,i}$ an $\varepsilon_i \times \varepsilon_i$ matrix and $b_{r,i}$ an $\varepsilon_i \times 1$ column vector of the forms

$$A_{r,i} = \begin{bmatrix} 0 & I_{\varepsilon_i - 1} \\ 0 & 0 \end{bmatrix} = J_{\varepsilon_i}(0), \qquad b_{r,i} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

and the pair (A_l, C_l) is in a Brunovsky observable form

$$A_l = \begin{bmatrix} A_{l,1} & & & \\ & A_{l,2} & & \\ & & \ddots & \\ & & & A_{l,\nu_l} \end{bmatrix}, \qquad C_l = \begin{bmatrix} c_{l,1} & & & \\ & c_{l,2} & & \\ & & \ddots & \\ & & & c_{l,\nu_l} \end{bmatrix}, \tag{18}$$

with $A_{l,i}$ an $\eta_i \times \eta_i$ matrix and $c_{l,i}$ a $1 \times \eta_i$ row vector of the forms

$$A_{l,i} = \begin{bmatrix} 0 & 0 \\ I_{\eta_i - 1} & 0 \end{bmatrix} = J_{\eta_i}^T(0), \qquad c_{l,i} = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}.$$

□

The computation of the Kronecker-canonical form may involve the use of ill-conditioned transformations and, therefore, is potentially numerically unstable. Fortunately, alternative staircase forms, called *Kronecker-like forms*, allow to obtain basically the same (or only a part of) structural information on the pencil $M - \lambda N$ by employing exclusively orthogonal transformations (i.e., $U^T U = I$ and $V^T V = I$).

The following result concerns with one of the main Kronecker-like forms.

Theorem 4. Let $M \in \mathbb{R}^{m \times n}$ and $N \in \mathbb{R}^{m \times n}$ be arbitrary real matrices. Then, there exist orthogonal $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$, such that

$$U(M - \lambda N)V = \begin{bmatrix} M_r - \lambda N_r & * & * \\ 0 & M_{reg} - \lambda N_{reg} & * \\ 0 & 0 & M_l - \lambda N_l \end{bmatrix}, \tag{19}$$

where

1) The $n_r \times (m_r + n_r)$ pencil $M_r - \lambda N_r$ has full row rank, n_r, for all $\lambda \in \mathbb{C}$ and is in a *controllability staircase form*

$$M_r - \lambda N_r = \begin{bmatrix} B_r & A_r - \lambda E_r \end{bmatrix}, \tag{20}$$

with $B_r \in \mathbb{R}^{n_r \times m_r}$, $A_r, E_r \in \mathbb{R}^{n_r \times n_r}$, and E_r invertible.

2) The $n_{reg} \times n_{reg}$ pencil $M_{reg} - \lambda N_{reg}$ is regular and its eigenvalues are the eigenvalues of pencil $M - \lambda N$. The pencil $M_{reg} - \lambda N_{reg}$ may be chosen in a GRSF, with arbitrary ordered diagonal blocks.

3) The $(p_l + n_l) \times n_l$ pencil $M_l - \lambda N_l$ has full column rank, n_l, for all $\lambda \in \mathbb{C}$ and is in an *observability staircase form*

$$M_l - \lambda N_l = \begin{bmatrix} A_l - \lambda E_l \\ C_l \end{bmatrix}, \tag{21}$$

with $C_l \in \mathbb{R}^{p_l \times n_l}$, $A_l, E_l \in \mathbb{R}^{n_l \times n_l}$, and E_l invertible.

The generalized controllability staircase form (20) is defined with

$$\left[\begin{array}{c|cccc} B_r & A_r \end{array}\right] = \left[\begin{array}{c|cccc} A_{1,0} & A_{1,1} & A_{12} & \cdots & A_{1,k-1} & A_{1,k} \\ 0 & A_{2,1} & A_{22} & \cdots & A_{2,k-1} & A_{2,k} \\ 0 & 0 & A_{32} & \cdots & A_{3,k-1} & A_{3,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{k,k-1} & A_{k,k} \end{array}\right], \tag{22}$$

where $A_{j,j-1} \in \mathbb{R}^{\nu_j \times \nu_{j-1}}$, with $\nu_0 = m_r$, are full row rank matrices for $j = 1, \ldots, k$, and the resulting upper triangular matrix E_r has a similar block partitioned form

$$E_r = \begin{bmatrix} E_{1,1} & E_{1,2} & \cdots & E_{1,k-1} & E_{1,k} \\ 0 & E_{2,2} & \cdots & E_{2,k-1} & E_{2,k} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & E_{k-1,k-1} & E_{k-1,k} \\ 0 & 0 & \cdots & 0 & E_{k,k} \end{bmatrix}, \tag{23}$$

where $E_{j,j} \in \mathbb{R}^{\nu_j \times \nu_j}$. The resulting block dimensions ν_j, $j = 0, 1, \ldots, k$, satisfy

$$m_r = \nu_0 \geq \nu_1 \geq \cdots \geq \nu_k > 0\,.$$

The dimensions ν_i, $i = 1, \ldots, k$, of the diagonal blocks of $A_r - \lambda E_r$ completely determine the right Kronecker structure of $M - \lambda N$ as follows: there are $\nu_{i-1} - \nu_i$ blocks $L_{i-1}(\lambda)$ of size $(i-1) \times i$, for $i = 1, \ldots, k$.

The generalized observability staircase form (21) is

$$\left[\begin{array}{c} A_l \\ \hline C_l \end{array}\right] = \left[\begin{array}{ccccc} A_{\ell,\ell} & A_{\ell,\ell-1} & \cdots & A_{\ell,2} & A_{\ell,1} \\ A_{\ell-1,\ell} & A_{\ell-1,\ell-1} & \cdots & A_{\ell-1,2} & A_{\ell-1,1} \\ 0 & A_{\ell-2,\ell-1} & \cdots & A_{\ell-2,2} & A_{\ell-2,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & A_{1,2} & A_{1,1} \\ \hline 0 & 0 & \cdots & 0 & A_{0,1} \end{array}\right], \tag{24}$$

where $A_{j-1,j} \in \mathbb{R}^{\mu_{j-1} \times \mu_j}$, with $\mu_0 = p_l$, are full column rank matrices for $j = 1, \ldots, \ell$, and the resulting upper triangular matrix E_o has a similar block partitioned form

$$E_l = \begin{bmatrix} E_{\ell,\ell} & E_{\ell,\ell-1} & \cdots & E_{\ell,2} & E_{\ell,1} \\ 0 & E_{\ell-1,\ell-1} & \cdots & E_{\ell-1,2} & E_{\ell-1,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & E_{2,2} & E_{2,1} \\ 0 & 0 & \cdots & 0 & E_{1,1} \end{bmatrix}, \tag{25}$$

with $E_{j,j} \in \mathbb{R}^{\mu_j \times \mu_j}$. The resulting block dimensions μ_j, $j = 0, 1, \ldots, \ell$, satisfy

$$p_l = \mu_0 \geq \mu_1 \cdots \geq \mu_\ell > 0\,.$$

The dimensions μ_i, $i = 1, \ldots, \ell$ of the diagonal blocks of $A_l - \lambda E_l$ in the observability staircase form completely determine the left Kronecker structure of $M - \lambda N$ as follows: there are $\mu_{i-1} - \mu_i$ blocks $L_{i-1}^T(\lambda)$ of size $i \times (i-1)$, $i = 1, \ldots, \ell$. We have $n_r = \sum_{i=1}^{k} \nu_i$ and $n_l = \sum_{i=1}^{\ell} \mu_i$, and the normal rank of $M - \lambda N$ is $n_r + n_{reg} + n_l$.

Algorithms for the computation of Kronecker-like forms of linear pencils, using SVD-based rank determinations, have been proposed in [38, 8]. Albeit numerically reliable, these algorithms have a computational complexity $\mathcal{O}(n^4)$, where n is the minimum of row or column dimensions of the pencil. More efficient algorithms of complexity $\mathcal{O}(n^3)$ have been proposed in [3, 48, 30], which rely on using QR decompositions with column pivoting for rank determinations.

2.4 Minimal Nullspace Bases

A p-dimensional rational vector $v(\lambda) \in \mathbb{R}(\lambda)^p$ can be seen as either a $1 \times p$ or a $p \times 1$ rational matrix. A set of rational vectors $V(\lambda) := \{v_1(\lambda), \ldots, v_k(\lambda)\}$ is said to be *linearly dependent* over the field $\mathbb{R}(\lambda)$ if there exists k rational functions $\gamma_i(\lambda) \in \mathbb{R}(\lambda)$, $i = 1, \ldots, k$, with $\gamma_i(\lambda) \neq 0$ for at least one i, such that, the linear combination

$$\sum_{i=1}^{k} \gamma_i(\lambda) v_i(\lambda) = 0\,. \tag{26}$$

The set of vectors $V(\lambda)$ is *linearly independent* over $\mathbb{R}(\lambda)$ if (26) implies that $\gamma_i(\lambda) = 0$ for each $i = 1, \ldots, k$. It is important to note, that a linearly dependent set $V(\lambda)$ over $\mathbb{R}(\lambda)$, can be still linearly independent over another field (e.g., the field of reals if $\gamma_i \in \mathbb{R}$).

The *normal rank* of a rational matrix $G(\lambda) \in \mathbb{R}(\lambda)^{p\times m}$, which we also denote by $\operatorname{rank} G(\lambda)$, is the maximal number of linearly independent rows (or columns) over the field of rational functions $\mathbb{R}(\lambda)$. It can be shown that the normal rank of $G(\lambda)$ is the maximally possible rank of the complex matrix $G(\lambda)$ for all values of $\lambda \in \mathbb{C}$ such that $G(\lambda)$ has finite norm. This interpretation provides a simple way to determine the normal rank as the maximum of the rank of $G(\lambda)$ for a few random values of the frequency variable λ.

It is easy to check that the set of p-dimensional rational vectors $\mathbb{R}(\lambda)^p$ forms a vector space with scalars defined over $\mathbb{R}(\lambda)$. If $\mathcal{V}(\lambda) \subset \mathbb{R}(\lambda)^p$ is a vector space, then there exists a set of linearly independent rational vectors $V(\lambda) := \{v_1(\lambda), \ldots, v_{n_b}(\lambda)\} \subset \mathcal{V}(\lambda)$ such that any vector in $\mathcal{V}(\lambda)$ is a linear combination of the vectors in $V(\lambda)$ (equivalently, any set of $n_b + 1$ vectors, including an arbitrary vector from $\mathcal{V}(\lambda)$ and the n_b vectors in $V(\lambda)$, is linearly dependent). The set $V(\lambda)$ is called a *basis* of the vector space $\mathcal{V}(\lambda)$ and n_b is the dimension of $\mathcal{V}(\lambda)$. With a slight abuse of notation, we denote $V(\lambda)$ the matrix formed of the n_b stacked row vectors

$$V(\lambda) = \begin{bmatrix} v_1(\lambda) \\ \vdots \\ v_{n_b}(\lambda) \end{bmatrix}$$

or the n_b concatenated column vectors

$$V(\lambda) = \begin{bmatrix} v_1(\lambda) & \cdots & v_{n_b}(\lambda) \end{bmatrix}.$$

Interestingly, as basis vectors we can always use polynomial vectors since we can replace each vector $v_i(\lambda)$ of a rational basis, by $v_i(\lambda)$ multiplied with the least common multiple of the denominators of the components of $v_i(\lambda)$.

The use of polynomial bases allows to define the main concepts related to so-called minimal bases. Let n_i be the degree of the i-th polynomial vector $v_i(\lambda)$ of a polynomial basis $V(\lambda)$ (i.e., n_i is the largest degree of the components of $v_i(\lambda)$). Then, $n := \sum_{i=1}^{n_b} n_i$ is, by definition, the *degree* of the polynomial basis $V(\lambda)$. A *minimal polynomial basis* of $\mathcal{V}(\lambda)$ is one for which n has the least achievable value. For a minimal polynomial basis, n_i, for $i = 1, \ldots, n_b$, are called the *row* or *column minimal indices* (also known as *left* or *right Kronecker indices*, respectively). Two important examples are the left and right nullspace bases of a rational matrix, which are shortly discussed below.

Let $G(\lambda)$ be a $p \times m$ rational matrix $G(\lambda)$ whose normal rank is $r < \min(p, m)$. It is easy to show that the set

$$\mathcal{N}_L(G(\lambda)) := \{v(\lambda) \in \mathbb{R}(\lambda)^{1\times p} \mid v(\lambda)G(\lambda) = 0\}$$

is a linear space called the *left nullspace* of $G(\lambda)$. Analogously,

$$\mathcal{N}_R(G(\lambda)) := \{v(\lambda) \in \mathbb{R}(\lambda)^{m\times 1} \mid G(\lambda)v(\lambda) = 0\}$$

is a linear space called the *right nullspace* of $G(\lambda)$. The dimension of $\mathcal{N}_L(G(\lambda))$ $[\mathcal{N}_R(G(\lambda))]$ is $p - r$ $[m - r]$, and therefore, there exist $p - r$ $[m - r]$ linearly independent polynomial vectors which form a minimal polynomial basis for $\mathcal{N}_L(G(\lambda))$ $[\mathcal{N}_R(G(\lambda))]$. Let $n_{l,i}$ $[n_{r,i}]$ be the *left* [*right*] *minimal indices* and let $n_l := \sum_{i=1}^{p-r} n_{l,i}$ $\left[n_r := \sum_{i=1}^{m-r} n_{r,i}\right]$ be the least possible degree of the left [right] nullspace basis. The least left and right degrees n_l and n_r, respectively, play an important role in relating the main structural elements of rational matrices (see the discussion of poles and zeros in Section 2.5).

Some properties of minimal polynomial bases are summarized below for the left nullspace bases. Similar results can be given for the right nullspace bases.

Lemma 3. Let $G(\lambda)$ be a $p \times m$ rational matrix of normal rank r and let $N_l(\lambda)$ be a $(p-r) \times p$ minimal polynomial basis of the left nullspace $\mathcal{N}_L(G(\lambda))$ with left minimal (or left Kronecker) indices $n_{l,i}$, $i = 1, \ldots, p - r$. Then the following holds:

1. The left minimal indices are unique up to permutations (i.e., if $\widetilde{N}_l(\lambda)$ is another minimal polynomial basis, then $N_l(\lambda)$ and $\widetilde{N}_l(\lambda)$ have the same left minimal indices).

2. $N_l(\lambda)$ is irreducible, having full row rank for all $\lambda \in \mathbb{C}$ (i.e., $N_l(\lambda)$ has no finite zeros, see Section 2.5).

3. $N_l(\lambda)$ is row reduced (i.e., the leading row coefficient matrix formed from the coefficients of the highest row degrees has full row rank.)

An irreducible and row-reduced polynomial basis is actually a minimal polynomial basis. Irreducibility implies that any polynomial vector $v(\lambda)$ in the space spanned by the rows of $N_l(\lambda)$, can be expressed as a linear combination of basis vectors $v(\lambda) = \phi(\lambda)N_l(\lambda)$, with $\phi(\lambda)$ being a polynomial vector. In particular, assuming the rows of $N_l(\lambda)$ are ordered such that $n_{l,i} \leq n_{l,i+1}$ for $i = 1, \ldots, p - r - 1$, then for any $v(\lambda)$ of degree $n_{l,i}$, the corresponding $\phi(\lambda)$ has as its j-th element a polynomial of degree at most $n_{l,i} - n_{l,j}$ for $j = 1, \ldots, i$, and the rest of components are zero. This property allows to easily generate left polynomial annihilators of given degrees.

Minimal polynomial bases allow to easily build *simple minimal proper rational bases*. These are proper rational bases having the property that the sum of degrees of the rows [columns] is equal to the least left [right] degree of a minimal polynomial basis n_l $[n_r]$. A simple minimal proper rational left nullspace basis with arbitrary poles can be constructed by forming $\widetilde{N}_l(\lambda) := M(\lambda)N_l(\lambda)$ with

$$M(\lambda) = \operatorname{diag}\left(1/d_1(\lambda), \ldots, 1/d_{p-r}(\lambda)\right), \tag{27}$$

where $d_i(\lambda)$ is a polynomial of degree $n_{l,i}$ with arbitrary roots. Since $N_l(\lambda)$ is row reduced, it follows that $D_l := \lim_{\lambda\to\infty} \widetilde{N}_l(\lambda)$ has full row rank (i.e., $\widetilde{N}_l(\lambda)$ has no infinite zeros, see Section 2.5). A simple minimal proper rational left nullspace basis allows to generate, in a

straightforward manner, left rational annihilators of given McMillan degrees by forming linear combinations of the basis vectors in $\widetilde{N}_l(\lambda)$ using specially chosen rational vectors $\phi(\lambda)$ (see Section 2.5 for the definition of the McMillan degree of a rational matrix). The concept of simple minimal proper rational basis has been introduced in [42] as the natural counterpart of a minimal polynomial basis.

Let $G(\lambda)$ be a $p \times m$ rational matrix of normal rank r and let $(A - \lambda E, B, C, D)$ be an irreducible descriptor realization of $G(\lambda)$. To determine a basis $N_l(\lambda)$ of the left nullspace of $G(\lambda)$, we can exploit the simple fact (see [62, Theorem 2]) that $N_l(\lambda)$ is a left minimal nullspace basis of $G(\lambda)$ if and only if, for a suitable $M_l(\lambda)$,

$$Y_l(\lambda) := [\, M_l(\lambda) \; N_l(\lambda) \,] \tag{28}$$

is a left minimal nullspace basis of the associated system matrix pencil

$$S(\lambda) := \begin{bmatrix} A - \lambda E & B \\ C & D \end{bmatrix}. \tag{29}$$

Thus, to compute $N_l(\lambda)$ we can determine first a left minimal nullspace basis $Y_l(\lambda)$ for $S(\lambda)$ and then $N_l(\lambda)$ results as

$$N_l(\lambda) = Y_l(\lambda) \begin{bmatrix} 0 \\ I_p \end{bmatrix}.$$

By duality, if $Y_r(\lambda)$ is a right minimal nullspace basis for $S(\lambda)$, then a right minimal nullspace basis of $G(\lambda)$ is given by

$$N_r(\lambda) = \begin{bmatrix} 0 & I_m \end{bmatrix} Y_r(\lambda).$$

The Kronecker canonical form (12) of the system matrix pencil $S(\lambda)$ in (29) allows to easily determine left and right nullspace bases of $G(\lambda)$. A numerically reliable computational approach to compute proper minimal nullspace bases of rational matrices is described in [51] and relies on using the Kronecker-like form (19) of the system matrix pencil, which can be determined by using exclusively orthogonal similarity transformations. The computation of simple minimal proper nullspace bases is described in [56].

2.5 Poles and Zeros

For a scalar rational function $g(\lambda) \in \mathbb{R}(\lambda)$, the values of λ for which $g(\lambda)$ is infinite are called the *poles* of $g(\lambda)$. If $g(\lambda) = \alpha(\lambda)/\beta(\lambda)$ has the form in (1), then the n roots λ_i^p, $i = 1, \ldots, n$, of $\beta(\lambda)$ are the *finite poles* of $g(\lambda)$, while if $m < n$, there are also, by convention, $n-m$ *infinite poles*. The values of λ for which $g(\lambda) = 0$ are called the *zeros* of $g(\lambda)$. The m roots λ_i^z, $i = 1, \ldots, m$, of $\alpha(\lambda)$ are the *finite zeros* of $g(\lambda)$, while if $n < m$, there are also, by convention, $m - n$ *infinite zeros*. It follows that the number of poles is equal to the number of zeros and is equal to $\max(m, n)$, the degree of $g(\lambda)$. The rational function $g(\lambda)$ in (1) can be equivalently expressed in terms of its finite poles and zeros in the factorized form

$$g(\lambda) = k_g \frac{\prod_{i=1}^{m}(\lambda - \lambda_i^z)}{\prod_{i=1}^{n}(\lambda - \lambda_i^p)}, \tag{30}$$

where $k_g = a_m/b_n$. If $g(\lambda)$ is the transfer function of a SISO LTI system, then we will always assume that $g(\lambda)$ is in a minimal cancelled (irreducible) form, that is, the polynomials $\alpha(\lambda)$

and $\beta(\lambda)$ in (1) have 1 as greatest common divisor. Equivalently, the two polynomials have no common roots, and therefore no pole-zero cancellation may occur in (30). Two such polynomials are called *coprime*.

In studying the stability of systems, the poles play a primordial role. Their real parts, in the case of a continuous-time system, or moduli, in the case of a discrete-time system, determine the asymptotic (exponential) decay or divergence speed of the system output. A SISO LTI system with the transfer function $g(\lambda)$ is *exponentially stable* (or equivalently $g(\lambda)$ is stable) if $g(\lambda)$ is proper and has all poles in the appropriate *stability domain* $\mathbb{C}_s$. The system is *unstable* if it has at least one pole outside of the stability domain and *anti-stable* if all poles lie outside of the stability domain. Poles inside the stability domain are called *stable poles*, while those outside of the stability domain are called *unstable poles*. For continuous-time systems the stability domain is the open left half complex plane $\mathbb{C}_s = \{s \in \mathbb{C} : \Re(s) < 0\}$, while for discrete-time systems the stability domain is the open unit disk $\mathbb{C}_s = \{z \in \mathbb{C} : |z| < 1\}$. We denote by $\partial\mathbb{C}_s$ the boundary of the stability domain. For continuous-time systems, the boundary of the stability domain is the extended imaginary axis (i.e., including the point at infinity) $\partial\mathbb{C}_s = \{\infty\} \cup \{s \in \mathbb{C} : \Re(s) = 0\}$, while for discrete-time systems the boundary of the stability domain is the unit circle $\partial\mathbb{C}_s = \{z \in \mathbb{C} : |z| = 1\}$. We denote $\overline{\mathbb{C}}_s = \mathbb{C}_s \cup \partial\mathbb{C}_s$ the closure of the stability domain. The *instability domain* of poles we denote by $\overline{\mathbb{C}}_u$ and is the complement of $\mathbb{C}_s$ in $\mathbb{C}$, $\overline{\mathbb{C}}_u = \mathbb{C} \setminus \mathbb{C}_s$. It is also the closure of the set denoted by $\mathbb{C}_u$, which for a continuous-time system is the open right-half plane $\mathbb{C}_u = \{s \in \mathbb{C} : \Re(s) > 0\}$, while for a discrete-time systems is the exterior of the unit circle $\mathbb{C}_u = \{z \in \mathbb{C} : |z| > 1\}$. The *stability degree* of poles is defined as the largest real part of the poles in the continuous-time case, or the largest absolute value of the poles in the discrete-time case.

Let $\mathbb{R}_s(\lambda)$ be the set of proper stable transfer functions having poles only in $\mathbb{C}_s$. A transfer function $g(\lambda) \in \mathbb{R}_s(\lambda)$ having only zeros in $\mathbb{C}_s$ is called *minimum-phase*. Otherwise it is called *non-minimum-phase*. The zeros of $g(\lambda)$ in $\mathbb{C}_s$ are called *minimum-phase zeros*, while those outside $\mathbb{C}_s$ are called *non-minimum-phase zeros*.

There are no straightforward generalizations of poles and zeros of scalar rational functions to the rational matrix case. Instrumental for a rigorous definition are two canonical forms: the *Smith form* for polynomial matrices and the *Smith-McMillan form* for rational matrices. For polynomial matrices we have the following important result.

Lemma 4. Let $P(\lambda) \in \mathbb{R}[\lambda]^{p\times m}$ be any polynomial matrix. Then, there exist unimodular matrices $U(\lambda) \in \mathbb{R}[\lambda]^{p\times p}$ and $V(\lambda) \in \mathbb{R}[\lambda]^{m\times m}$ such that

$$U(\lambda)P(\lambda)V(\lambda) = S(\lambda) := \begin{bmatrix} \alpha_1(\lambda) & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \alpha_2(\lambda) & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \cdots & \alpha_r(\lambda) & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \tag{31}$$

and $\alpha_i(\lambda)$ divides $\alpha_{i+1}(\lambda)$ for $i = 1, \ldots, r-1$.

The polynomial matrix $S(\lambda)$ is called the *Smith form* of $P(\lambda)$ and r is the normal rank of $P(\lambda)$. The diagonal elements $\alpha_1(\lambda), \ldots, \alpha_r(\lambda)$ are called the *invariant polynomials* of $P(\lambda)$. The roots of the polynomials $\alpha_i(\lambda)$, for $i = 1, \ldots, r$, are called the *finite zeros* of the polynomial

matrix $P(\lambda)$. To each distinct finite zero λ_z of $P(\lambda)$, we can associate the multiplicities $\sigma_i(\lambda_z) \geq 0$ of root λ_z in each of the polynomials $\alpha_i(\lambda)$, for $i = 1, \ldots, r$. By convention, $\sigma_i(\lambda_z) = 0$ if λ_z is not a root of $\alpha_i(\lambda)$. The divisibility properties of $\alpha_i(\lambda)$ imply that

$$0 \leq \sigma_1(\lambda_z) \leq \sigma_2(\lambda_z) \leq \cdots \leq \sigma_r(\lambda_z)\,.$$

Any rational matrix $G(\lambda)$ can be expressed as

$$G(\lambda) = \frac{P(\lambda)}{d(\lambda)},$$

where $d(\lambda)$ is the monic least common multiple of the denominator polynomials of the entries of $G(\lambda)$, and $P(\lambda) := d(\lambda)G(\lambda)$ is a polynomial matrix. Then, we have the following straightforward extension of Lemma 4 to rational matrices.

Lemma 5. Let $G(\lambda) \in \mathbb{R}(\lambda)^{p\times m}$ be any rational matrix. Then, there exist unimodular matrices $U(\lambda) \in \mathbb{R}[\lambda]^{p\times p}$ and $V(\lambda) \in \mathbb{R}[\lambda]^{m\times m}$ such that

$$U(\lambda)G(\lambda)V(\lambda) = H(\lambda) := \begin{bmatrix} \frac{\alpha_1(\lambda)}{\beta_1(\lambda)} & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \frac{\alpha_2(\lambda)}{\beta_2(\lambda)} & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \cdots & \frac{\alpha_r(\lambda)}{\beta_r(\lambda)} & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}, \tag{32}$$

with $\alpha_i(\lambda)$ and $\beta_i(\lambda)$ coprime for $i = 1, \ldots, r$ and $\alpha_i(\lambda)$ divides $\alpha_{i+1}(\lambda)$ and $\beta_{i+1}(\lambda)$ divides $\beta_i(\lambda)$ for $i = 1, \ldots, r-1$.

The rational matrix $H(\lambda)$ is called the *Smith–McMillan form* of $G(\lambda)$ and r is the normal rank of $G(\lambda)$. The Smith-McMillan form is a powerful conceptual tool which allows to define rigorously the notions of poles and zeros of MIMO LTI systems and to establish several basic factorization results of rational matrices.

The roots of the numerator polynomials $\alpha_i(\lambda)$, for $i = 1, \ldots, r$, are called the *finite zeros* of the rational matrix $G(\lambda)$ and the roots of the denominator polynomials $\beta_i(\lambda)$, for $i = 1, \ldots, r$, are called the *finite poles* of the rational matrix $G(\lambda)$. To each finite λ_z, which is a zero or a pole of $G(\lambda)$ (or both), we can associate its multiplicities $\{\sigma_1(\lambda_z), \ldots, \sigma_r(\lambda_z)\}$, where $\sigma_i(\lambda_z)$ is the multiplicity of λ_z either as a pole or a zero of the ratio $\alpha_i(\lambda)/\beta_i(\lambda)$, for $i = 1, \ldots, r$. By convention, we use negative values for poles and positive values for zeros. The divisibility properties of $\alpha_i(\lambda)$ and $\beta_i(\lambda)$ imply that

$$\sigma_1(\lambda_z) \leq \sigma_2(\lambda_z) \leq \cdots \leq \sigma_r(\lambda_z)\,.$$

The r-tuple of multiplicities $\{\sigma_1(\lambda_z), \ldots, \sigma_r(\lambda_z)\}$ completely characterizes the *pole-zero structure* of $G(\lambda)$ in λ_z.

The relative degrees of $\alpha_i(\lambda)/\beta_i(\lambda)$ do not provide the correct information on the multiplicity of infinite zeros and poles. This is because the used unimodular transformations may have poles and zeros at infinity. To overcome this, the multiplicity of zeros and poles at infinity are defined

in terms of multiplicities of poles and zeros of $G(1/\lambda)$ in $\lambda_z = 0$. This allows to define the multiplicities of *infinite zeros* of a polynomial matrix $P(\lambda)$ as the multiplicities of the null poles in the Smith-McMillan for of $P(1/\lambda)$.

The *McMillan degree* of a rational matrix $G(\lambda)$, usually denoted by $\delta(G(\lambda))$, is the number n_p of its poles, both finite and infinite, counting all multiplicities. If n_z is the number of zeros (finite and infinite, counting all multiplicities), then we have the following important structural relation for any rational matrix [62]

$$n_p = n_z + n_l + n_r, \tag{33}$$

where n_l and n_r are the least degrees of the minimal polynomial bases for the left and right nullspaces of $G(\lambda)$, respectively.

For a given rational matrix $G(\lambda)$, any (finite or infinite) pole of its elements $g_{ij}(\lambda)$, is also a pole of $G(\lambda)$. Therefore, many notions related to poles of SISO LTI systems introduced previously can be extended in a straightforward manner to MIMO LTI systems. For example, the notion of properness of $G(\lambda)$ can be equivalently defined as the nonexistence of infinite poles in the elements of $G(\lambda)$ (i.e., $G(\lambda)$ has only finite poles). The notion of *exponential stability* of a LTI system with a proper TFM $G(\lambda)$ can be defined as the lack of unstable poles in all elements of $G(\lambda)$ (i.e., all poles of $G(\lambda)$ are in the stable domain $\mathbb{C}_s$). A TFM $G(\lambda)$ with only stable poles is called *stable*. Otherwise, $G(\lambda)$ is called *unstable*. A similar definition applies for the *stability degree* of poles.

The zeros of $G(z)$ are also called the *transmission zeros* of the corresponding LTI system. A proper and stable $G(z)$ is *minimum-phase* if all its finite zeros are stable. Otherwise, it is called *non-minimum-phase*.

Consider the irreducible descriptor system $(A - \lambda E, B, C, D)$ with the corresponding TFM $G(\lambda) \in \mathbb{R}(\lambda)^{p\times m}$. Two pencils play a fundamental role in defining the main structural elements of the rational matrix $G(\lambda)$. The regular *pole pencil*

$$P(\lambda) := A - \lambda E \tag{34}$$

characterizes the pole structure of $G(\lambda)$, exhibited by the Weierstrass canonical form of the pole pencil $P(\lambda)$. The (singular) system matrix pencil

$$S(\lambda) := \begin{bmatrix} A - \lambda E & B \\ C & D \end{bmatrix} \tag{35}$$

characterizes the zero structure of $G(\lambda)$, as well as the right- and left-singular structures of $G(\lambda)$, which are exhibited by the Kronecker canonical form of the system matrix pencil $S(\lambda)$.

The main structural results for the rational TFM $G(\lambda)$ can be stated in terms of its irreducible descriptor system realization $(A - \lambda E, B, C, D)$ of order n. The following facts rely on the Weierstrass canonical form (8) of the pole pencil $P(\lambda)$ in (34) (see Lemma 1) and the Kronecker canonical form (12) of the system matrix pencil $S(\lambda)$ in (35) (see Lemma 2):

1) The *finite poles* of $G(\lambda)$ are the finite eigenvalues of the pole pencil $P(\lambda)$ and are the eigenvalues (counting multiplicities) of the matrix J_f in the Weierstrass canonical form (8) of the pencil $P(\lambda)$.

2) The *infinite poles* of $G(\lambda)$ have multiplicities defined by the multiplicities of the infinite eigenvalues of the pole pencil $P(\lambda)$ minus 1 and are the dimensions minus 1 of the nilpotent Jordan blocks in the matrix J_∞ in the Weierstrass canonical form (8) of the pencil $P(\lambda)$.

3) The *finite zeros* of $G(\lambda)$ are the n_f finite eigenvalues (counting multiplicities) of the system matrix pencil $S(\lambda)$ and are the eigenvalues (counting multiplicities) of the matrix $\widetilde{J}_f$ in the Kronecker canonical form (12) of the pencil $S(\lambda)$.

4) The *infinite zeros* of $G(\lambda)$ have multiplicities defined by the multiplicities of the infinite eigenvalues of the system matrix pencil $S(\lambda)$ minus 1 and are the dimensions minus 1 of the nilpotent Jordan blocks in the matrix $\widetilde{J}_\infty$ in the Kronecker canonical form (12) of the pencil $S(\lambda)$.

5) The *left minimal indices* of $G(\lambda)$ are pairwise equal to the left Kronecker indices of $S(\lambda)$ and are the row dimensions ϵ_i of the blocks $L_{\epsilon_i}(\lambda)$ for $i = 1, \ldots, \nu_r$ in the Kronecker canonical form (12) of the pencil $S(\lambda)$.

6) The *right minimal indices* of $G(\lambda)$ are pairwise equal to the right Kronecker indices of $S(\lambda)$ and are the column dimensions η_i of the blocks $L_{\eta_i}^T(\lambda)$ for $i = 1, \ldots, \nu_l$ in the Kronecker canonical form (12) of the pencil $S(\lambda)$.

7) The *normal rank* of $G(\lambda)$ is $r = \operatorname{rank} S(\lambda) - n = n_r + \tilde{n}_f + \tilde{n}_\infty + n_l - n$.

These facts allow to formulate simple conditions to characterize some pole-zero related properties, such as, properness, stability or minimum-phase of an irreducible descriptor system $(A - \lambda E, B, C, D)$ in terms of the eigenvalues of the pole and system matrix pencils. The descriptor system $(A - \lambda E, B, C, D)$ is *proper* if all infinite eigenvalues of the regular pencil $A - \lambda E$ are simple (i.e., the system has no infinite poles). It is straightforward to show using the Weierstrass canonical form of the pencil $A - \lambda E$, that any irreducible proper descriptor system can be always reduced to a minimal order descriptor system, with the descriptor matrix E invertible, or even to a standard state-space representation with $E = I$. The irreducible descriptor system $(A - \lambda E, B, C, D)$ is *improper* if the regular pencil $A - \lambda E$ has at least one infinite eigenvalue which is not simple (i.e, has at least one infinite pole). A *polynomial* descriptor system is one for which $A - \lambda E$ has only infinite eigenvalues of which at least one is not simple (i.e, has only infinite poles). The concept of stability involves naturally the properness of the system. The irreducible descriptor system $(A - \lambda E, B, C, D)$ is *exponentially stable* if it has only finite poles and all poles belong to the stable region $\mathbb{C}_s$ (the pencil $A - \lambda E$ still can have simple infinite eigenvalues). The irreducible descriptor system $(A - \lambda E, B, C, D)$ is *unstable* if it has at least one finite pole outside of the stability domain or at least one infinite pole. The finite poles (or finite eigenvalues) inside the stability domain are called *stable poles* (*stable eigenvalues*), while the poles lying outside of the stability domain are called *unstable poles*. The irreducible descriptor system $(A - \lambda E, B, C, D)$ is *minimum-phase* if it has only finite zeros and all finite zeros belong to the stable region $\mathbb{C}_s$.

To check the finite controllability condition (i) and finite observability condition (iii) of Theorem 2, it is sufficient to check that

$$\operatorname{rank} \begin{bmatrix} A - \lambda_i E & B \end{bmatrix} = n \tag{36}$$

and, respectively,

$$\operatorname{rank} \begin{bmatrix} A - \lambda_i E \\ C \end{bmatrix} = n \tag{37}$$

for all distinct finite eigenvalues λ_i of the regular pencil $A - \lambda E$. A finite eigenvalue λ_i is *controllable* if (36) is fulfilled, and *uncontrollable* otherwise. Similarly, a finite eigenvalue λ_i

is *observable* if (37) is fulfilled, and *unobservable* otherwise. If the rank conditions (36) are fulfilled for all $\lambda_i \in \overline{\mathbb{C}}_u$ we call the descriptor system $(A-\lambda E, B, C, D)$ (or equivalently the pair $(A-\lambda E, B)$) *finite stabilizable*. Finite stabilizability guarantees the existence of a state-feedback matrix $F \in \mathbb{R}^{m\times n}$ such that all finite eigenvalues of $A+BF-\lambda E$ lie in $\mathbb{C}_s$. If the rank conditions (37) are fulfilled for all $\lambda_i \in \overline{\mathbb{C}}_u$ we call the descriptor system $(A-\lambda E, B, C, D)$ (or equivalently the pair $(A-\lambda E, C)$) *finite detectable*. Finite detectability guarantees the existence of an output-injection matrix $K \in \mathbb{R}^{n\times p}$ such that all finite eigenvalues of $A+KC-\lambda E$ lie in $\mathbb{C}_s$.

The notion of strong stabilizability is related to the existence of a state-feedback matrix F such that all finite eigenvalues of $A+BF-\lambda E$ lie in $\mathbb{C}_s$ and all infinite eigenvalues of $A+BF-\lambda E$ are simple. The necessary and sufficient conditions for the existence of such an F is the *strong stabilizability* of the pair $(A-\lambda E, B)$, that is: (1) the finite stabilizability of the pair $(A-\lambda E, B)$; and (2) $\mathrm{rank}[\, E \;\; AN_\infty \;\; B\,] = n$, where the columns of N_∞ form a basis of $\mathcal{N}(E)$. Similarly, strong detectability is related to the existence of an output-injection matrix K such that all finite eigenvalues of $A+KC-\lambda E$ lie in $\mathbb{C}_s$ and all infinite eigenvalues of $A+KC-\lambda E$ are simple. The necessary and sufficient conditions for the existence of such a K is the *strong detectability* of the pair $(A-\lambda E, C)$, that is: (1) the finite detectability of the pair $(A-\lambda E, C)$; and (2) $\mathrm{rank}[\, E^T \;\; A^T L_\infty \;\; C^T\,] = n$, where the columns of L_∞ for a basis of $\mathcal{N}(E^T)$.

2.6 Range and Coimage Space Bases

For any $p \times m$ real rational matrix $G(\lambda)$ of normal rank r, there exists a *full rank factorization* of $G(\lambda)$ of the form

$$G(\lambda) = R(\lambda)X(\lambda), \tag{38}$$

where $R(\lambda)$ is a $p \times r$ full column rank rational matrix and $X(\lambda)$ is a $r \times m$ full row rank rational matrix. This factorization generalizes the full-rank factorization of constant matrices, and, similarly to the constant case, it is not unique. Indeed, for any $r \times r$ invertible rational matrix $M(\lambda)$, $G(\lambda) = \widetilde{R}(\lambda)\widetilde{X}(\lambda)$, with $\widetilde{R}(\lambda) = R(\lambda)M(\lambda)$ and $\widetilde{X} = M^{-1}(\lambda)X(\lambda)$, is also a full rank factorization of $G(\lambda)$.

Using (38), it is straightforward to show that $G(\lambda)$ and $R(\lambda)$ have the same range space over the rational functions

$$\mathcal{R}(G(\lambda)) = \mathcal{R}(R(\lambda)).$$

For this reason, with a little abuse of language, we will call $R(\lambda)$ the range (or image) matrix of $G(\lambda)$ (or simply the range of $G(\lambda)$). Since $R(\lambda)$ has normal rank r, its columns form a set of r basis vectors of $\mathcal{R}(G(\lambda))$.

The poles of $R(\lambda)$ can be chosen (almost) arbitrary, by replacing $R(\lambda)$ with $\widetilde{R}(\lambda)$, the numerator factor of right coprime factorization $R(\lambda) = \widetilde{R}(\lambda)M^{-1}(\lambda)$, where $\widetilde{R}(\lambda)$ and $M(\lambda)$ have only poles in a given complex domain $\mathbb{C}_g$. From the full-rank factorization (38) results that each zero of $G(\lambda)$ is either a zero of $R(\lambda)$ or of $X(\lambda)$. Therefore, it is possible to construct full-rank factorizations with $R(\lambda)$ having as zeros a symmetric subset of zeros of $G(\lambda)$. A minimal McMillan degree basis results if $R(\lambda)$ has no zeros. Of special interest in some applications are bases which are inner, such that $R(\lambda)$ is stable and satisfies $R^{\sim}(\lambda)R(\lambda) = I_r$. Applications of such bases are the computation of inner-outer factorizations (see Section 2.9) and of the normalized coprime factorizations (see Section 2.8). Methods to compute various full rank factorizations are discussed in [57], based on techniques developed in [32] and [29].

Similarly, for any $p \times m$ real rational matrix $G(\lambda)$ of normal rank r, there exists a *full rank factorization* of $G(\lambda)$ of the form

$$G(\lambda) = X(\lambda)R(\lambda), \tag{39}$$

where $R(\lambda)$ is a $r \times m$ full row rank rational matrix and $X(\lambda)$ is a $p \times r$ full column rank rational matrix. Using (39), it is straightforward to show that $G(\lambda)$ and $R(\lambda)$ have the same coimage space (or row space) over the rational functions, which can be also expressed as

$$\mathcal{R}(G^T(\lambda)) = \mathcal{R}(R^T(\lambda)).$$

Since $R(\lambda)$ has normal rank r, its rows form a set of r basis vectors of the row space of $G(\lambda)$ (i.e., the coimage of $G(\lambda)$). As in the case of the range space bases, the coimage space basis $R(\lambda)$ of $G(\lambda)$ can be determined to have (almost) arbitrarily chosen poles and the zeros of $R(\lambda)$ include an arbitrary symmetric subset of zeros of $G(\lambda)$. Moreover, $R(\lambda)$ can be chosen stable, without zeros and coinner, satisfying $R(\lambda)R^{\sim}(\lambda) = I_r$.

2.7 Additive Decompositions

Let $G(\lambda)$ be the transfer function matrix of a LTI system and let $\mathbb{C}_g$ be a domain of interest of the complex plane $\mathbb{C}$ for the poles of $G(\lambda)$ (e.g., a certain stability domain). Define $\mathbb{C}_b := \mathbb{C} \setminus \mathbb{C}_g$, the complement of $\mathbb{C}_g$ in $\mathbb{C}$. Since $\mathbb{C}_g$ and $\mathbb{C}_b$ are disjoint, each pole of any element $g_{ij}(\lambda)$ of $G(\lambda)$ lies either in $\mathbb{C}_g$ or in $\mathbb{C}_b$. Therefore, using the well-known partial fraction decomposition results of rational functions, $G(\lambda)$ can be additively decomposed as

$$G(\lambda) = G_1(\lambda) + G_2(\lambda), \tag{40}$$

where $G_1(\lambda)$ has only poles in $\mathbb{C}_g$, while $G_2(\lambda)$ has only poles in $\mathbb{C}_b$. For such a decomposition of $G(\lambda)$ we always have that

$$\delta\left(G(\lambda)\right) = \delta\left(G_1(\lambda)\right) + \delta\left(G_2(\lambda)\right) .$$

For example, if $\mathbb{C}_g = \mathbb{C}\setminus\{\infty\}$ and $\mathbb{C}_b = \{\infty\}$, then (40) represents the additive decomposition of a possibly improper rational matrix as the sum of its proper and polynomial parts. This decomposition, in general, is not unique, because an arbitrary constant term can be always added to one term and subtracted from the other one. Another frequently used decomposition is the stable-unstable decomposition of proper rational matrices, when $\mathbb{C}_g = \mathbb{C}_s$ (stability region) and $\mathbb{C}_b = \mathbb{C}_u$ (instability region).

Let $G(\lambda) = (A - \lambda E, B, C, D)$ be a descriptor system representation of $G(\lambda)$. Using a general similarity transformation using two invertible matrices Q and Z, we can determine an equivalent representation of $G(\lambda)$ with partitioned system matrices of the form

$$G(\lambda) = \left[\begin{array}{c|c} QAZ - \lambda QEZ & QB \\ \hline CZ & D \end{array}\right] = \left[\begin{array}{cc|c} A_1 - \lambda E_1 & 0 & B_1 \\ 0 & A_2 - \lambda E_2 & B_2 \\ \hline C_1 & C_2 & D \end{array}\right], \tag{41}$$

where $\Lambda(A_1 - \lambda E_1) \subset \mathbb{C}_1$ and $\Lambda(A_2 - \lambda E_2) \subset \mathbb{C}_2$. It follows that $G(\lambda)$ can be additively decomposed as

$$G(\lambda) = G_1(\lambda) + G_2(\lambda), \tag{42}$$

where

$$G_1(\lambda) = \left[\begin{array}{c|c} A_1 - \lambda E_1 & B_1 \\ \hline C_1 & D \end{array}\right], \quad G_2(\lambda) = \left[\begin{array}{c|c} A_2 - \lambda E_2 & B_2 \\ \hline C_2 & 0 \end{array}\right], \tag{43}$$

and $G_1(\lambda)$ has only poles in $\mathbb{C}_g$, while $G_2(\lambda)$ has only poles in $\mathbb{C}_b$. For the computation of additive spectral decompositions, a numerically reliable procedure is described in [21].

2.8 Coprime Factorizations

Consider a disjunct partition of the complex plane $\mathbb{C}$ as

$$\mathbb{C} = \mathbb{C}_g \cup \mathbb{C}_b, \quad \mathbb{C}_g \cap \mathbb{C}_b = \emptyset\,, \tag{44}$$

where both $\mathbb{C}_g$ and $\mathbb{C}_b$ are symmetrically located with respect to the real axis, and such that $\mathbb{C}_g$ has at least one point on the real axis. Any rational matrix $G(\lambda)$ can be expressed in a left fractional form

$$G(\lambda) = M^{-1}(\lambda)N(\lambda)\,, \tag{45}$$

or in a right fractional form

$$G(\lambda) = N(\lambda)M^{-1}(\lambda)\,, \tag{46}$$

where both the denominator factor $M(\lambda)$ and the numerator factor $N(\lambda)$ have only poles in $\mathbb{C}_g$. These fractional factorizations over a "good" domain of poles $\mathbb{C}_g$ are important in various observer, fault detection filter, or controller synthesis methods, because they allow to achieve the placement of all poles of a TFM $G(\lambda)$ in the domain $\mathbb{C}_g$ simply, by a premultiplication or postmultiplication of $G(\lambda)$ with a suitable $M(\lambda)$.

Of special interest are the so-called coprime factorizations, where the factors satisfy additional conditions. A fractional representation of the form (45) is a *left coprime factorization* (LCF) of $G(\lambda)$ with respect to $\mathbb{C}_g$, if there exist $U(\lambda)$ and $V(\lambda)$ with poles only in $\mathbb{C}_g$ which satisfy the *Bezout identity*

$$M(\lambda)U(\lambda) + N(\lambda)V(\lambda) = I\,.$$

A fractional representation of the form (46) is a *right coprime factorization* (RCF) of $G(\lambda)$ with respect to $\mathbb{C}_g$, if there exist $U(\lambda)$ and $V(\lambda)$ with poles only in $\mathbb{C}_g$ which satisfy

$$U(\lambda)M(\lambda) + V(\lambda)N(\lambda) = I\,.$$

An important class of coprime factorizations is the class of coprime factorizations with minimum-degree denominators. Recall that $\delta\,(G(\lambda))$, the McMillan degree of $G(\lambda)$, is defined as the number of poles of $G(\lambda)$, both finite and infinite, counting all multiplicities. It follows that for any $G(\lambda)$ we have $\delta\,(G(\lambda)) = n_g + n_b$, where n_g and n_b are the number of poles of $G(\lambda)$ in $\mathbb{C}_g$ and $\mathbb{C}_b$, respectively. The denominator factor $M(\lambda)$ has the minimum-degree property if $\delta\,(M(\lambda)) = n_b$.

A square TFM $G(\lambda)$ is *all-pass* if $G^{\sim}(\lambda)G(\lambda) = I$. If $G(\lambda)$ is a stable TFM and satisfies $G^{\sim}(\lambda)G(\lambda) = I$ then it is called an *inner* TFM, while if it satisfies $G(\lambda)G^{\sim}(\lambda) = I$ it is called a *coinner* TFM. Note that an inner or coinner TFM must not be square, but must have full column rank (injective) or full row rank (surjective), respectively. It is remarkable, that each proper TFM $G(\lambda)$ without poles on the boundary of stability domain $\partial\mathbb{C}_s$ has a stable LCF of the form (45) or a stable RCF of the form (46) with the denominator factor $M(\lambda)$ inner. As

before, the minimum McMillan degree of the inner denominator $M(\lambda)$ is equal to the number of the unstable poles of $G(\lambda)$.

A special class of coprime factorizations consists of the so-called normalized coprime factorizations. For an arbitrary $p \times m$ rational matrix $G(\lambda)$, the *normalized left coprime factorization* of $G(\lambda)$ is

$$G(\lambda) = M^{-1}(\lambda)N(\lambda), \tag{47}$$

where $N(\lambda)$ and $M(\lambda)$ are stable TFMs and $[\, N(\lambda)\ M(\lambda)\,]$ is *coinner*, that is

$$N(\lambda)N^{\sim}(\lambda) + M(\lambda)M^{\sim}(\lambda) = I_p. \tag{48}$$

Similarly, the *normalized right coprime factorization* of $G(\lambda)$ is

$$G(\lambda) = N(\lambda)M^{-1}(\lambda), \tag{49}$$

where $N(\lambda)$ and $M(\lambda)$ are stable TFMs and $\left[\begin{smallmatrix} N(\lambda) \\ M(\lambda) \end{smallmatrix}\right]$ is inner, that is

$$N^{\sim}(\lambda)N(\lambda) + M^{\sim}(\lambda)M(\lambda) = I_m. \tag{50}$$

For the computation of coprime factorizations with minimum degree denominators, descriptor system representation based methods have been proposed in [49, 58], by using iterative pole dislocation techniques developed in the spirit of the approach described in [40]. Alternative, non-iterative approaches to compute minimum degree coprime factorizations with inner denominators have been proposed in [31, 29]. For the computation of normalized coprime factorizations computational methods have been proposed in [28] (see also [57]).

2.9 Inner-Outer and Spectral Factorizations

A proper and stable TFM $G(\lambda)$ is *outer* if it is minimum-phase and full row rank (surjective), and is *co-outer* if it is minimum-phase and full column rank (injective). A full row rank (full column rank) proper and stable TFM $G(\lambda)$ is *quasi-outer* (*quasi-co-outer*) if it has only zeros in $\overline{\mathbb{C}}_s$ (i.e, in the stability domain and its boundary). Any stable TFM $G(\lambda)$ without zeros in $\partial\mathbb{C}_s$ has an *inner–outer factorization*

$$G(\lambda) = G_i(\lambda)G_o(\lambda)\,, \tag{51}$$

with $G_i(\lambda)$ inner and $G_o(\lambda)$ outer. Similarly, $G(\lambda)$ has a *co-outer–coinner factorization*

$$G(\lambda) = G_{co}(\lambda)G_{ci}(\lambda)\,, \tag{52}$$

with $G_{co}(\lambda)$ co-outer and $G_{ci}(\lambda)$ coinner. Any stable TFM $G(\lambda)$ has an *inner–quasi-outer factorization* of the form (51), where $G_i(\lambda)$ is inner and $G_o(\lambda)$ is quasi-outer, and also has a *quasi-co-outer–coinner factorization* of the form (52), where $G_{ci}(\lambda)$ is coinner and $G_{co}(\lambda)$ is quasi-co-outer.

In some applications, instead of the (compact) inner-outer factorization (51), an alternative (extended) factorization with square inner factor is desirable. The *extended inner-outer factorization* and *extended inner–quasi-outer factorization* have the form

$$G(\lambda) = \left[\begin{array}{cc} G_i(\lambda) & G_i^{\perp}(\lambda) \end{array}\right] \left[\begin{array}{c} G_o(\lambda) \\ 0 \end{array}\right] = U(\lambda) \left[\begin{array}{c} G_o(\lambda) \\ 0 \end{array}\right], \tag{53}$$

where $G_i^{\perp}(\lambda)$ is the inner orthogonal complement of $G_i(\lambda)$ such that $U(\lambda) := \left[G_i(\lambda)\ G_i^{\perp}(\lambda)\right]$ is square and inner. Similarly, the *extended co-outer–coinner factorization* and *extended quasi-co-outer–coinner factorization* have the form

$$G(\lambda) = \begin{bmatrix} G_{co}(\lambda) & 0 \end{bmatrix} \begin{bmatrix} G_{ci}(\lambda) \\ G_{ci}^{\perp}(\lambda) \end{bmatrix} = \begin{bmatrix} G_{co}(\lambda) & 0 \end{bmatrix} V(\lambda)\,, \tag{54}$$

where $G_{ci}^{\perp}(\lambda)$ is the coinner orthogonal complement of $G_i(\lambda)$ such that $V(\lambda) := \begin{bmatrix} G_{ci}(\lambda) \\ G_{ci}^{\perp}(\lambda) \end{bmatrix}$ is square and coinner (thus also inner).

The extended inner-outer factorization (53) of a TFM $G(\lambda)$ can be interpreted as a generalization of the orthogonal QR-factorization of a real matrix. The inner factor $U(\lambda) = \left[G_i(\lambda)\ G_i^{\perp}(\lambda)\right]$ can be seen as the generalization of an orthogonal matrix. Its role in an inner-outer factorization is twofold: to compress the given $G(\lambda)$ to a full row rank TFM $G_o(\lambda)$ and to dislocate all zeros of $G(\lambda)$ lying in $\mathbb{C}_u$ into positions within $\mathbb{C}_s$, which are symmetric (in a certain sense) with respect to $\partial\mathbb{C}_s$. A factorization of $G(\lambda)$ as in (53), where only the first aspect (i.e., the row compression) is addressed is called an *extended QR-like factorization* of the rational matrix $G(\lambda)$ and produces a compressed full row rank $G_o(\lambda)$, which contains all zeros of $G(\lambda)$. The similar factorization in (54) with $V(\lambda)$ inner, which only addresses the column compression aspect for $G(\lambda)$, is called an *extended RQ-like factorization* of $G(\lambda)$. Applications of these factorizations are in computing the pseudo-inverse of a rational matrix [32] (see also Example 9).

The outer factor $G_o(\lambda)$ of a TFM $G(\lambda)$ without zeros in $\partial\mathbb{C}_s$ satisfies

$$G^{\sim}(\lambda)G(\lambda) = G_o^{\sim}(\lambda)G_o(\lambda)$$

and therefore, it is a solution of the *minimum-phase right spectral factorization* problem. Similarly, the co-outer factor $G_{co}(\lambda)$ of a TFM $G(\lambda)$ without zeros in $\partial\mathbb{C}_s$ satisfies

$$G(\lambda)G^{\sim}(\lambda) = G_{co}(\lambda)G_{co}^{\sim}(\lambda)$$

and therefore, it is a solution of the *minimum-phase left spectral factorization* problem.

Combining the LCF with inner denominator and the inner-outer factorization, we have for an arbitrary $G(\lambda)$, without poles and zeros on the boundary of the stability domain $\partial\mathbb{C}_s$, that

$$G(\lambda) = M_i^{-1}(\lambda)N(\lambda) = M_i^{-1}(\lambda)N_i(\lambda)N_o(\lambda),$$

where $M_i(\lambda)$ and $N_i(\lambda)$ are inner and $N_o(\lambda)$ is outer. It follows that the outer factor $N_o(\lambda)$ is the solution of the *stable minimum-phase right spectral factorization* problem

$$G^{\sim}(\lambda)G(\lambda) = N_o^{\sim}(\lambda)N_o(\lambda)\,.$$

Similarly, by combining the RCF with inner denominator and the co-outer–coinner factorization we obtain

$$G(\lambda) = N(\lambda)M_i^{-1}(\lambda) = N_{co}(\lambda)N_{ci}(\lambda)M_i^{-1}(\lambda)\,,$$

with $M_i(\lambda)$ inner, $N_{ci}(\lambda)$ coinner and $N_{co}(\lambda)$ co-outer. Then, $N_{co}(\lambda)$ is the solution of the *stable minimum-phase left spectral factorization* problem

$$G(\lambda)G^{\sim}(\lambda) = N_{co}(\lambda)N_{co}^{\sim}(\lambda)\,.$$

If $G(\lambda)$ has poles or zeros on the boundary of the stability domain $\partial\mathbb{C}_s$, then we can still achieve the above factorizations by including all poles and zeros of $G(\lambda)$ in $\partial\mathbb{C}_s$ in the resulting spectral factors $N_o(\lambda)$ or $N_{co}(\lambda)$.

Assume that the stable proper TFM $G(\lambda)$ has an irreducible descriptor realization

$$G(\lambda) = \left[\begin{array}{c|c} A-\lambda E & B \\ \hline C & D \end{array}\right]. \tag{55}$$

For illustration of the inner-outer factorization approach, we only consider in detail the *standard problem* with E invertible and when $G(\lambda)$ has full column rank, in which case explicit formulas for both the inner and outer factors can be derived using the results presented in [69]. Similar dual results can be obtained if $G(\lambda)$ has full row rank. The factorization procedures to compute inner–quasi-outer factorizations in the most general setting (i.e., for $G(\lambda)$ possibly improper with arbitrary normal rank and arbitrary zeros), are described in [32] for continuous-time systems and in [29] for discrete-time systems. These methods, essentially reduce the factorization problems to the solution of standard problems, for which straightforward extensions of the results for standard systems in [69] apply.

We have the following result for a continuous-time system:

Theorem 5. Let $G(s)$ be a proper and stable, full column rank TFM without zeros in $\partial\mathbb{C}_s$, with the descriptor system realization $G(s) = (A-sE, B, C, D)$ and E invertible. Then, $G(s)$ has an inner–outer factorization $G(s) = G_i(s)G_o(s)$, with the particular realizations of the factors

$$G_i(s) = \left[\begin{array}{c|c} A+BF-sE & BH^{-1} \\ \hline C+DF & DH^{-1} \end{array}\right], \qquad G_o(s) = \left[\begin{array}{c|c} A-sE & B \\ \hline -HF & H \end{array}\right],$$

where H is an invertible matrix satisfying $D^TD = H^TH$, F is given by

$$F = -(D^TD)^{-1}(B^TX_sE + D^TC),$$

with $X_s \geq 0$ being the stabilizing solution of the generalized continuous-time algebraic Riccati equation *(GCARE)*

$$A^TXE + E^TXA - (E^TXB + C^TD)(D^TD)^{-1}(B^TXE + D^TC) + C^TC = 0.$$

The similar result for a discrete-time system is:

Theorem 6. Let $G(z)$ be a proper and stable, full column rank TFM without zeros in $\partial\mathbb{C}_s$, with the descriptor system realization $G(z) = (A-zE, B, C, D)$ and E invertible. Then, $G(z)$ has an inner–outer factorization $G(z) = G_i(z)G_o(z)$, with the particular realizations of the factors

$$G_i(z) = \left[\begin{array}{c|c} A+BF-zE & BH^{-1} \\ \hline C+DF & DH^{-1} \end{array}\right], \qquad G_o(z) = \left[\begin{array}{c|c} A-zE & B \\ \hline -HF & H \end{array}\right],$$

where H is an invertible matrix satisfying $D^TD + B^TX_sB = H^TH$, F is given by

$$F = -(H^TH)^{-1}(B^TX_sA + D^TC),$$

with $X_s \geq 0$ being the stabilizing solution of the generalized discrete-time algebraic Riccati equation *(GDARE)*

$$A^TXA - E^TXE - (A^TXB + C^TD)(D^TD + B^TXB)^{-1}(B^TXA + D^TC) + C^TC = 0.$$

When computing the *extended inner-outer factorization* (53), the inner orthogonal complement $G_i^{\perp}(\lambda)$ of $G_i(\lambda)$ is also needed. In the continuous-time case, a descriptor realization of $G_i^{\perp}(\lambda)$ is given by

$$G_i^{\perp}(s) = \left[\begin{array}{c|c} A+BF-sE & -X_s^{\dagger}E^{-T}C^TD^{\perp} \\ \hline C+DF & D^{\perp} \end{array}\right],$$

where $D^{\perp}$ is an orthogonal complement chosen such that $\left[\begin{array}{cc} DH^{-1} & D^{\perp} \end{array}\right]$ is square and orthogonal. In the discrete-time case we have

$$G_i^{\perp}(z) = \left[\begin{array}{c|c} A+BF-zE & Y \\ \hline C+DF & W \end{array}\right],$$

where Y and W satisfy

$$\begin{array}{rcl} A^TX_sY+C^TW & = & 0, \\ B^TX_sY+D^TW & = & 0, \\ W^TW+Y^TX_sY & = & I. \end{array}$$

The similar results for the co-outer–coinner factorization (or the extended co-outer–coinner factorization) can be easily obtained by considering the inner–outer factorization (or its extended version) for the dual system with the TFM $G^T(\lambda)$ having the descriptor realization $(A^T-\lambda E^T, C^T, B^T, D^T)$.

A special factorization problem encountered when solving the approximate model-matching problem by reducing it to a least distance problem (see equation (85) in Section 2.16) is the following *special left spectral factorization problem*: for a given TFM $G(\lambda)$ without poles in $\partial\mathbb{C}_s$ and a given bound $\gamma > \|G(\lambda)\|_\infty$, compute a stable and minimum-phase TFM $G_o(\lambda)$ such that

$$\gamma^2 I - G(\lambda)G^{\sim}(\lambda) = G_o(\lambda)G_o^{\sim}(\lambda)\,.$$

This computation can be addressed in two steps. In the first step, we compute a RCF $G(\lambda) = N(\lambda)M^{-1}(\lambda)$, with the denominator factor $M(\lambda)$ inner. It follows that

$$\gamma^2 I - G(\lambda)G^{\sim}(\lambda) = \gamma^2 I - N(\lambda)N^{\sim}(\lambda),$$

where $N(\lambda)$ is proper and has only poles in $\mathbb{C}_s$. In the second step, we determine the stable and minimum-phase $G_o(\lambda)$ which satisfies

$$\gamma^2 I - N(\lambda)N^{\sim}(\lambda) = G_o(\lambda)G_o^{\sim}(\lambda)\,. \tag{56}$$

The first step has been already discussed in Section 2.8, and therefore we assume that for an irreducible descriptor realization $(A-\lambda E, B, C, D)$ of $G(\lambda)$, we determined a stable $N(\lambda)$ with a descriptor realization $(\widetilde{A}-\lambda\widetilde{E}, \widetilde{B}, \widetilde{C}, \widetilde{D})$.

In the continuous-time case, we can compute the spectral factor $G_o(s)$ by using the following result, which extends to proper descriptor systems the formulas developed in [69, Corollary 13.22].

Lemma 6. Let $N(s)$ be a stable TFM and let $(\widetilde{A}-s\widetilde{E}, \widetilde{B}, \widetilde{C}, \widetilde{D})$ be its descriptor system realization. For $\gamma > \|N(s)\|_\infty$, a realization of a stable and minimum-phase spectral factor $G_o(s)$, satisfying (56) for $\lambda = s$, is given by

$$G_o(s) = \left[\begin{array}{c|c} \widetilde{A}-s\widetilde{E} & -K_sR^{1/2} \\ \hline \widetilde{C} & R^{1/2} \end{array}\right],$$

where

$$\begin{aligned} R &= \gamma^2 I - \widetilde{D}\widetilde{D}^T, \\ K_s &= (\widetilde{E}Y_s\widetilde{C}^T + \widetilde{B}\widetilde{D}^T)R^{-1}, \end{aligned}$$

and Y_s is the stabilizing solution of the GCARE

$$\widetilde{A}Y\widetilde{E}^T + \widetilde{E}Y\widetilde{A}^T + (\widetilde{E}Y\widetilde{C}^T + \widetilde{B}\widetilde{D}^T)R^{-1}(\widetilde{C}Y\widetilde{E}^T + \widetilde{D}\widetilde{B}^T) + \widetilde{B}\widetilde{B}^T = 0\,.$$

We have the following analogous result in the discrete-time case, which extends to proper descriptor system the formulas developed in [69, Theorem 21.26] for the dual special right spectral factorization problem (see below).

Lemma 7. Let $N(z)$ be a stable TFM and let $(\widetilde{A} - z\widetilde{E}, \widetilde{B}, \widetilde{C}, \widetilde{D})$ be its descriptor realization. For $\gamma > \|N(z)\|_\infty$, a realization of a stable and minimum-phase spectral factor $G_o(z)$, satisfying (56) for $\lambda = z$, is given by

$$G_o(z) = \left[\begin{array}{c|c} \widetilde{A} - \lambda\widetilde{E} & -K_s R^{1/2} \\ \hline \widetilde{C} & R^{1/2} \end{array}\right],$$

where

$$\begin{aligned} R_D &= \gamma^2 I - \widetilde{D}\widetilde{D}^T, \\ R &= R_D - \widetilde{C}Y_s\widetilde{C}^T, \\ K_s &= (\widetilde{A}Y_s\widetilde{C}^T + \widetilde{B}\widetilde{D}^T)R^{-1}, \end{aligned}$$

and Y_s is the stabilizing solution of the GDARE

$$\widetilde{A}Y\widetilde{A}^T - \widetilde{E}Y\widetilde{E}^T - (\widetilde{A}Y\widetilde{C}^T + \widetilde{B}\widetilde{D}^T)(-R_D + \widetilde{C}Y\widetilde{C}^T)^{-1}(\widetilde{C}Y\widetilde{A}^T + \widetilde{D}\widetilde{B}^T) + \widetilde{B}\widetilde{B}^T = 0\,.$$

Similar formulas to those provided by Lemma 6 and Lemma 7 can be derived for the solution of the dual *special right spectral factorization problem*

$$\gamma^2 I - N^\sim(\lambda)N(\lambda) = G_o^\sim(\lambda)G_o(\lambda).$$

.

2.10 Linear Rational Matrix Equations

For $G(\lambda) \in \mathbb{R}(\lambda)^{p\times m}$ and $F(\lambda) \in \mathbb{R}(\lambda)^{p\times q}$ consider the solution of the linear rational matrix equation

$$G(\lambda)X(\lambda) = F(\lambda)\,, \tag{57}$$

where $X(\lambda) \in \mathbb{R}(\lambda)^{m\times q}$ is the solution we seek. The existence of a solution is guaranteed if the compatibility condition for the linear system (57) is fulfilled.

Lemma 8. The rational equation (57) has a solution if and only if

$$\operatorname{rank} G(\lambda) = \operatorname{rank}\,[\,G(\lambda)\ F(\lambda)\,]\,. \tag{58}$$

Let r be the rank of $G(\lambda)$. In the most general case, the solution of (57) (if exists) is not unique and can be expressed as

$$X(\lambda) = X_0(\lambda) + N_r(\lambda)Y(\lambda), \tag{59}$$

where $X_0(\lambda)$ is a particular solution of (57), $N_r(\lambda) \in \mathbb{R}(\lambda)^{m\times(m-r)}$ is a rational matrix representing a basis of the right nullspace $\mathcal{N}_R(G(\lambda))$ (is empty if $r = m$), while $Y(\lambda) \in \mathbb{R}(\lambda)^{(m-r)\times q}$ is an arbitrary rational matrix.

An important aspect in control related applications is to establish conditions which ensure the existence of a solution $X(\lambda)$ which has only poles in a "good" domain $\mathbb{C}_g$, or equivalently, $X(\lambda)$ has no poles in the "bad" domain $\mathbb{C}_b := \mathbb{C}\setminus\mathbb{C}_g$. Such a condition can be obtained in terms of the pole-zero structures of the rational matrices $G(\lambda)$ and $[\,G(\lambda)\ F(\lambda)\,]$ at a particular value λ_z of the frequency parameter λ (see, for example, [22]).

Lemma 9. The rational equation (57) has a solution without poles in $\mathbb{C}_b$ if and only if (58) is fulfilled and the rational matrices $G(\lambda)$ and $[\,G(\lambda)\ F(\lambda)\,]$ have the same pole-zero structure for all $\lambda_z \in \mathbb{C}_b$.

The characterization provided by Lemma 9 is relevant when solving synthesis problems of fault detection filters and controllers using an *exact model-matching* approach, where the physical realizability requires the properness and stability of the solutions (i.e., $\mathbb{C}_g = \mathbb{C}_s$). For example, if $G(\lambda)$ has unstable zeros in λ_z, then $F(\lambda)$ must be chosen to have the same (or richer) zero structure in λ_z, in order to ensure the cancellation of these zeros (appearing now as unstable poles of any particular solution $X_0(\lambda)$). The fixed poles in $\mathbb{C}_b$ of any particular solution $X_0(\lambda)$ correspond to those zeros of $G(\lambda)$ for which the above condition is not fulfilled, and thus no complete cancellations take place.

In the case when $\operatorname{rank} G(\lambda) < m$, an important aspect is the exploitation of the non-uniqueness of the solution in (57) by determining a solution with the least possible McMillan degree. This problem is known in the literature as the *minimum design problem* (MDP) and primarily targets the reduction of the complexity of real-time burden when implementing filters or controllers. Of particular importance are proper and stable solutions which are suitable for a physical (causal) realization. If the minimal degree solution is not proper and stable, then it is of interest to find a proper and stable solution with the least McMillan degree. Surprisingly, this problem does not have a satisfactory procedural solution, most of proposed approaches involves parametric searches using suitably parameterized solutions of given candidate degrees.

An equivalent solvability condition to that of Lemma 8 can be derived in terms of descriptor system representations of $G(\lambda)$ and $F(\lambda)$, which we assume to be of the form

$$G(\lambda) = \left[\begin{array}{c|c} A-\lambda E & B_G \\ \hline C & D_G \end{array}\right], \qquad F(\lambda) = \left[\begin{array}{c|c} A-\lambda E & B_F \\ \hline C & D_F \end{array}\right]. \tag{60}$$

Such representations, which share the pair $(A-\lambda E, C)$, can be easily obtained by determining a descriptor realization of the compound rational matrix $[\,G(\lambda)\ F(\lambda)\,]$. It is easy to observe that any solution of (57) is also part of the solution of the linear polynomial equation

$$\left[\begin{array}{cc} A-\lambda E & B_G \\ C & D_G \end{array}\right] Y(\lambda) = \left[\begin{array}{c} B_F \\ D_F \end{array}\right], \tag{61}$$

where $Y(\lambda) = \left[\begin{smallmatrix} W(\lambda) \\ X(\lambda) \end{smallmatrix}\right]$. Therefore, alternatively to solving (57), we can solve instead (61) for $Y(\lambda)$ and compute $X(\lambda)$ as

$$X(\lambda) = [\,0\ \ I_m\,]Y(\lambda)\,. \tag{62}$$

Define the system matrix pencils corresponding to $G(\lambda)$ and the compound $[\,G(\lambda)\ F(\lambda)\,]$ as

$$S_G(\lambda) := \left[\begin{array}{cc} A-\lambda E & B_G \\ C & D_G \end{array}\right], \qquad S_{G,F}(\lambda) := \left[\begin{array}{ccc} A-\lambda E & B_G & B_F \\ C & D_G & D_F \end{array}\right]. \tag{63}$$

We have the following result similar to Lemma 8.

Lemma 10. The rational equation (57) with $G(\lambda)$ and $F(\lambda)$ having the descriptor realizations in (60) has a solution if and only if

$$\operatorname{rank} S_G(\lambda) = \operatorname{rank} S_{G,F}(\lambda)\,. \tag{64}$$

Let $\mathbb{C}_b$ be the "bad" domain of the complex plane, where the solution $X(\lambda)$ must not have poles. We have the following result similar to Lemma 9.

Lemma 11. The rational equation (57) with $G(\lambda)$ and $F(\lambda)$ having the descriptor realizations in (60) has a solution without poles in $\mathbb{C}_b$ if and only if the matrix pencils $S_G(\lambda)$ and $S_{G,F}(\lambda)$ defined in (63) fulfill (64) and have the same zero structure for all zeros of $G(\lambda)$ in $\mathbb{C}_b$.

The general solution of (57) can be expressed as in (59) where $X_0(\lambda)$ is any particular solution of (57), $N_r(\lambda)$ is a rational matrix whose columns form a basis for the right nullspace of $G(\lambda)$, and $Y(\lambda)$ is an arbitrary rational matrix with compatible dimensions. A possible approach to compute a solution $X(\lambda)$ of least McMillan degree is to determine a suitable $Y(\lambda)$ to achieve this goal. A computational procedure to determine a least McMillan degree solution $X(\lambda)$ is described in [59, Section 10.3.7] and relies on the Kronecker-like form of the associated system matrix pencil $S_G(\lambda)$ in (63), which is used to determine the particular solution $X_0(\lambda)$, the nullspace basis $N_r(\lambda)$, and, combined with the generalized minimum cover algorithm of [54], to obtain the least McMillan degree solution.

2.11 Dynamic Cover-Based Order Reduction

Let $X_1(\lambda)$ and $X_2(\lambda)$ be rational transfer function matrices. For $X_1(\lambda)$ and $X_2(\lambda)$ with the same row dimension, the *right minimal cover problem* is to find $X(\lambda)$ and $Y(\lambda)$ such that

$$X(\lambda) = X_1(\lambda) + X_2(\lambda)Y(\lambda), \tag{65}$$

and the McMillan degree of $\left[\begin{smallmatrix} X(\lambda) \\ Y(\lambda) \end{smallmatrix}\right]$ is minimal [1]. Similarly, for $X_1(\lambda)$ and $X_2(\lambda)$ with the same column dimension, the *left minimal cover problem* is to find $X(\lambda)$ and $Y(\lambda)$ such that

$$X(\lambda) = X_1(\lambda) + Y(\lambda)X_2(\lambda), \tag{66}$$

and the McMillan degree of $[\,X(\lambda)\ Y(\lambda)\,]$ is minimal [1]. Any method to solve a right minimal cover problem can be used to solve the left minimal proper problem, by applying it to the transposed matrices $X_1^T(\lambda)$ and $X_2^T(\lambda)$ to determine the transposed solution $X^T(\lambda)$ and $Y^T(\lambda)$. Therefore, in what follows we will only discuss the solution of right minimal cover problems.

By imposing additional conditions on $X_1(\lambda)$ and $X_2(\lambda)$, as well as on the class of desired solutions $X(\lambda)$ and $Y(\lambda)$, particular cover problems arise, which may require specific methods for their solution. In what follows, we only discuss the proper case, when $X_1(\lambda)$, $X_2(\lambda)$, $X(\lambda)$ and $Y(\lambda)$ are all proper. Various applications involving the solution of minimal cover problems

are discussed in [1], as for example—observer and controller synthesis using model-matching based approaches. Other problems, as for example, imposing $X(\lambda) = 0$, are equivalent to solve linear rational equations for the least McMillan degree solutions (see Section 2.10).

Assume $X_1(\lambda)$ and $X_2(\lambda)$ have the descriptor realizations

$$\left[\begin{array}{cc} X_1(\lambda) & X_2(\lambda) \end{array}\right] = \left[\begin{array}{c|cc} A-\lambda E & B_1 & B_2 \\ \hline C & D_1 & D_2 \end{array}\right], \tag{67}$$

with the descriptor pair $(A - \lambda E, [\, B_1 \;\; B_2 \,])$ controllable and E invertible. The methods we describe, use for $Y(\lambda)$ the descriptor realization

$$Y(\lambda) = \left[\begin{array}{c|c} A+B_2F-\lambda E & B_1+B_2G \\ \hline F & G \end{array}\right], \tag{68}$$

where F and G are state-feedback gain and feedforward gains, respectively, to be determined. It is straightforward to check that $X(\lambda) = X_1(\lambda)+X_2(\lambda)Y(\lambda)$ has the descriptor system realization

$$X(\lambda) := \left[\begin{array}{c|c} A+B_2F-\lambda E & B_1+B_2G \\ \hline C+D_2F & D_1+D_2G \end{array}\right]. \tag{69}$$

If the gains F and G are determined such that the pair $(A+B_2F-\lambda E, B_1+B_2G)$ is maximally uncontrollable, then the resulting realizations of $X(\lambda)$ and $Y(\lambda)$ contain a maximum number of uncontrollable eigenvalues which can be eliminated using minimal realization techniques. In some applications, the use of a strictly proper $Y(\lambda)$ (i.e., $G = 0$) is sufficient to achieve the maximal order reduction, therefore, this case will be treated explicitly in what follows.

The problem to determine the matrices F and G, which make the descriptor system pair $(A + B_2F - \lambda E, B_1 + B_2G)$ maximally uncontrollable, is essentially equivalent [27] to compute a subspace $\mathcal{V}$ having the least possible dimension and satisfying

$$(\overline{A} + \overline{B}_2F)\mathcal{V} \subset \mathcal{V}, \quad \text{span}\,(\overline{B}_1 + \overline{B}_2G) \subset \mathcal{V}\,, \tag{70}$$

where $\overline{A} := E^{-1}A$, $\overline{B}_1 := E^{-1}B_1$, and $\overline{B}_2 := E^{-1}B_2$. If we denote $\overline{\mathcal{B}}_1 = \text{span}\,\overline{B}_1$ and $\overline{\mathcal{B}}_2 = \text{span}\,\overline{B}_2$, then the above condition for $G = 0$ can be equivalently rewritten as the condition defining a *Type 1* minimum dynamic cover [11, 23]:

$$\overline{A}\mathcal{V} \subset \mathcal{V} + \overline{\mathcal{B}}_2, \qquad \overline{\mathcal{B}}_1 \subset \mathcal{V}. \tag{71}$$

If we allow for $G \neq 0$, then (70) can be equivalently rewritten as the condition defining a *Type 2* minimum dynamic cover [11, 23]:

$$\overline{A}\mathcal{V} \subset \mathcal{V} + \overline{\mathcal{B}}_2, \qquad \overline{\mathcal{B}}_1 \subset \mathcal{V} + \overline{\mathcal{B}}_2. \tag{72}$$

The underlying computational problems to solve these minimum dynamic cover problems is the following: given the controllable descriptor system pair $(A - \lambda E, B)$ with $A, E \in \mathbb{R}^{n\times n}$ and E invertible, $B \in \mathbb{R}^{n\times m}$, and B partitioned as $B = [\, B_1 \;\; B_2 \,]$ with $B_1 \in \mathbb{R}^{n\times m_1}$, $B_2 \in \mathbb{R}^{n\times m_2}$, determine the matrices F and G (either with $G = 0$ or $G \neq 0$) such that the descriptor system pair $(A + B_2F - \lambda E, B_1 + B_2G)$ has the maximal number of uncontrollable eigenvalues. Computational procedures to solve these problems have been proposed in [54] for descriptor systems and in [52] for standard systems (see also **Procedures GRMCOVER1** and **GRMCOVER2**, described in [59]).

2.12 Hankel Norm

The *Hankel norm* of a stable TFM $G(\lambda)$ is a measure of the influence of past inputs on future outputs and is denoted by $\|G(\lambda)\|_H$. The precise mathematical definition of the Hankel norm involves some advanced concepts from functional analysis (i.e., it is the norm of the Hankel operator Γ_G associated to $G(\lambda)$). For further details, see [69].

For the evaluation of the Hankel norm of a stable TFM $G(\lambda)$, the state-space representation allows to use explicit formulas.

Lemma 12. Let $G(s)$ be a proper and stable TFM of a continuous-time system and let $(A - sE, B, C, D)$ be an irreducible descriptor realization with E invertible. Then, the Hankel norm of $G(s)$ can be evaluated as

$$\|G(s)\|_H = \overline{\sigma}(RES),$$

where $P = SS^T$ is the positive semi-definite controllability Gramian and $Q = R^TR$ is the positive semi-definite observability Gramian, which satisfy the following generalized Lyapunov equations

$$\begin{array}{rcl} APE^T + EPA^T + BB^T & = & 0\,, \\ A^TQE + E^TQA + C^TC & = & 0\,. \end{array} \tag{73}$$

Lemma 13. Let $G(z)$ be a proper and stable TFM of a continuous-time system and let $(A - zE, B, C, D)$ be an irreducible descriptor realization with E invertible. Then, the Hankel norm of $G(z)$ can be evaluated as

$$\|G(z)\|_H = \overline{\sigma}(RES),$$

where $P = SS^T$ is the positive semi-definite controllability Gramian and $Q = R^TR$ is the positive semi-definite observability Gramian, which satisfy the following generalized Stein equations

$$\begin{array}{rcl} APA^T - EPE^T + BB^T & = & 0\,, \\ A^TQA - E^TQE + C^TC & = & 0\,. \end{array} \tag{74}$$

For the solution of the generalized Lyapunov equations (73) and the generalized Stein equations (74), computational procedures have been proposed in [33], which allow to directly determine the Cholesky factors S and R of the Gramians. These methods extend the algorithm of [18] proposed for standard systems with $E = I$.

2.13 The Gap Metric

The ν-gap metric $\delta_\nu(G_1, G_2)$ has been introduced in [65] for standard systems, as a natural measure of the "distance" between two proper transfer function matrices $G_1(\lambda)$ and $G_2(\lambda)$ of the same dimensions. The definition of $\delta_\nu(G_1, G_2)$ given in [65] can be extended to arbitrary transfer function matrices, because this definition involves only the normalized left coprime factorization $G_1(\lambda) = \widetilde{M}_1^{-1}(\lambda)\widetilde{N}_1(\lambda)$ and the right coprime factorizations $G_1(\lambda) = N_1(\lambda)M_1^{-1}(\lambda)$ and $G_2(\lambda) = N_2(\lambda)M_2^{-1}(\lambda)$. This definition can also serve as basis for a computational procedure. With $\widetilde{L}_2(\lambda) := [\,-\widetilde{M}_2(\lambda)\ \widetilde{N}_2(\lambda)\,]$, $R_i(\lambda) := \left[\begin{smallmatrix} N_i(\lambda) \\ M_i(\lambda) \end{smallmatrix}\right]$ for $i = 1, 2$, and $g(\lambda) := \det\left(R_2^{\sim}(\lambda)R_1(\lambda)\right)$, we have the following definition:

$$\delta_\nu(G_1, G_2) := \begin{cases} \left\|\widetilde{L}_2(\lambda)R_1(\lambda)\right\|_\infty & \text{if } g(\lambda) \neq 0\ \forall \lambda \in \partial\mathbb{C}_s \text{ and } \operatorname{wno}(g) = 0, \\ 1 & \text{otherwise,} \end{cases} \tag{75}$$

where wno(g) denotes the *winding number* of $g(\lambda)$ about the appropriate critical point for λ following the corresponding standard Nyquist contour. The winding number of $g(\lambda)$ can be determined as the difference between the number of unstable zeros of $g(\lambda)$ and the number of unstable poles of $g(\lambda)$ [64]. Generally, for any $G_1(\lambda)$ and $G_2(\lambda)$, we have $0 \leq \delta_\nu(G_1, G_2) \leq 1$. If $\delta_\nu(G_1, G_2)$ is small, then we can say that $G_1(\lambda)$ and $G_2(\lambda)$ are close and it is likely that a controller or filter suited for $G_1(\lambda)$ will also work with $G_2(\lambda)$. On the other side, if $\delta_\nu(G_1, G_2)$ is nearly equal to 1, then $G_1(\lambda)$ and $G_2(\lambda)$ are sufficiently distinct, such that an easy discrimination between the two models is possible. A common criticism of the ν-gap metric is that there are many transfer function matrices $G_2(\lambda)$ at a distance $\delta_\nu(G_1, G_2) = 1$ to a given $G_1(\lambda)$, but the metric fails to differentiate between them.

In [66], the point-wise ν-gap metric is also defined to evaluate the distance between two models in a single frequency point. If λ_s is a fixed complex frequency, then the point-wise ν-gap metric between two transfer-function matrices $G_1(\lambda)$ and $G_2(\lambda)$ at λ_s is:

$$\delta_\nu(G_1(\lambda_s), G_2(\lambda_s)) := \begin{cases} \left\| \widetilde{L}_2(\lambda_s) R_1(\lambda_s) \right\|_2 & \text{if } g(\lambda) \neq 0 \ \forall \lambda \in \partial \mathbb{C}_s \text{ and } \text{wno}(g) = 0, \\ 1 & \text{otherwise,} \end{cases} \tag{76}$$

2.14 Balancing-Related Order Reduction

Consider a stable TFM $G(\lambda)$ with a state-space descriptor system realization $(A - \lambda E, B, C, D)$ with E invertible and $\Lambda(A, E) \subset \mathbb{C}_s$. The controllability and observability properties of the state-space realization can be characterized by the controllability and observability gramians. The controllability gramian P and observability gramian Q satisfy appropriate generalized Lyapunov and Stein equations. In the continuous-time case, P and Q satisfy the generalized Lyapunov equations (73), while in the discrete-time case, P and Q satisfy the generalized Stein equations (74). For a stable system, both gramians P and Q are positive semi-definite matrices and fully characterize the controllability and observability properties. The controllability gramian $P > 0$ if and only if the pair $(A - \lambda E, B)$ is controllable, and the observability gramian $Q > 0$ if and only if the pair $(A - \lambda E, C)$ is observable. Besides these qualitative characterizations, the eigenvalues of P and Q provides quantitative measures of these properties. Thus, the degree of controllability can be defined as the least eigenvalue of P and represents a measure of the nearness of the state-space realization to an uncontrollable one. Analogously, the degree of observability, defined as the least eigenvalue of Q, represents a measure of the nearness of the state-space realization to an unobservable one. These measures are *not* invariant to coordinate transformations and, therefore, it is of interest to have state-space representations for which the controllability and observability properties are balanced. In fact, for a controllable and observable system, it is possible to find a particular state-space representation, called a *balanced realization*, for which the two gramians are equal and even diagonal.

The eigenvalues of the gramians of a balanced system are called the *Hankel singular values*. The largest singular value represents the *Hankel norm* $\|G(\lambda)\|_H$ of the corresponding TFM $G(\lambda)$, while the smallest one can be interpreted as a measure of the nearness of the system to a non-minimal one. Important applications of balanced realizations are to ensure minimum sensitivity to roundoff errors of real-time filter models or to perform model order reduction, by reducing large order models to lower order approximations. The order reduction can be performed by simply truncating the system state to a part corresponding to the "large" singular values, which significantly exceed the rest of "small" singular values.

A procedure to compute minimal balanced realizations of stable descriptor systems is **Procedure GBALMR** described in [59, Section 10.4.4]. This procedure is instrumental in solving the Nehari approximation problem for descriptor systems (see Section 2.15). The reduction of a linear state-space model to a balanced minimal realization may involve the usage of ill-conditioned coordinate transformations (or projections) for systems which are nearly non-minimal or nearly unstable. This is why, for the computation of minimal realizations or of lower order approximations, the so-called *balancing-free* approaches, as proposed in [46] for standard systems and in [35] for descriptor systems, are generally more accurate.

2.15 Solution of the Optimal Nehari Problems

In this section we consider the solution of the following optimal Nehari problem: Given an antistable $G(\lambda)$ (i.e., such that $G^{\sim}(\lambda)$ is stable), find a stable $X(\lambda)$ which is the closest to $G(\lambda)$ and satisfies

$$\|G(\lambda) - X(\lambda)\|_\infty = \|G^{\sim}(\lambda)\|_H \,. \tag{77}$$

This computation is encountered in the solution of the least-distance problem formulated in Section 2.16. As shown in [15], to solve the optimal Nehari approximation problem (77), we can solve instead for $X^{\sim}(\lambda)$ the optimal zeroth-order Hankel-norm approximation problem

$$\|G^{\sim}(\lambda) - X^{\sim}(\lambda)\|_\infty = \|G^{\sim}(\lambda)\|_H \,. \tag{78}$$

A computational method for continuous-time systems, can be devised using the method proposed in [15], which relies on computing first a balanced minimal order state-space realization of $G^{\sim}(\lambda)$. The corresponding procedure for discrete-time systems is much more involved (see [17]) and therefore, a preferred alternative, suggested in [15], is to use the procedure for continuous-time systems in conjunction with bilinear transformation techniques. This approach underlies **Procedure GNEHARI** described in [59, Section 10.4.5], which can be employed even for improper discrete-time systems.

If ρ is a given value satisfying $\rho > \|G^{\sim}(\lambda)\|_H$, then the suboptimal Nehari approximation problem is to determine a stable $X(\lambda)$ such that

$$\|G(\lambda) - X(\lambda)\|_\infty < \rho \,. \tag{79}$$

To solve this problem, a suboptimal Hankel-norm approximation $X^{\sim}(\lambda)$ of $G^{\sim}(\lambda)$ can be computed using the method of [15].

2.16 Solution of Least-Distance Problems

A possible solution method of the $\mathcal{H}_\infty$-model-matching problem [12] (see also Section 2.17) is to reduce this problem to a *least-distance problem* (LDP), which can be solved using Nehari-approximation techniques. The optimal LDP is the problem of computing a stable solution $X(\lambda)$ such that

$$\|[\, G_1(\lambda) - X(\lambda) \;\; G_2(\lambda) \,]\|_\infty = \min \,, \tag{80}$$

where $G_1(\lambda)$ and $G_2(\lambda)$ are TFMs without poles in $\partial\mathbb{C}_s$. Therefore, the use of the $\mathcal{L}_\infty$-norm in (80) is assumed. The suboptimal LDP is to find a stable $X(\lambda)$ such that

$$\|[\, G_1(\lambda) - X(\lambda) \;\; G_2(\lambda) \,]\|_\infty < \gamma \,, \tag{81}$$

where $\gamma > \|G_2(\lambda)\|_\infty$. If $G_2(\lambda)$ is present, we have a *2-block* LDP, while if $G_2(\lambda)$ is not present we have an *1-block* LDP. In what follows, we discuss shortly the solution approaches for the optimal 1- and 2-block problems.

Solution of the 1-block $\mathcal{H}_\infty$-LDP. In the case of the $\mathcal{L}_\infty$-norm, the stable optimal solution $X(\lambda)$ of the 1-block problem can be computed by solving an optimal Nehari problem. Let $L_s(\lambda)$ be the stable part and let $L_u(\lambda)$ be the unstable part in the additive decomposition

$$G_1(\lambda) = L_s(\lambda) + L_u(\lambda)\,. \tag{82}$$

Then, for the optimal solution we have successively

$$\|G_1(\lambda) - X(\lambda)\|_\infty = \|L_u(\lambda) - X_s(\lambda)\|_\infty = \|L_u^\sim(\lambda)\|_H\,,$$

where $X_s(\lambda)$ is the stable optimal Nehari solution and

$$X(\lambda) = X_s(\lambda) + L_s(\lambda)\,.$$

Solution of the 2-block $\mathcal{H}_\infty$-LDP. A stable optimal solution $X(\lambda)$ of the 2-block LDP can be approximately determined as the solution of the suboptimal 2-block LDP

$$\|[\,G_1(\lambda) - X(\lambda)\;\; G_2(\lambda)\,]\|_\infty < \gamma, \tag{83}$$

where $\gamma_{opt} < \gamma \leq \gamma_{opt} + \varepsilon$, with ε an arbitrary user specified (accuracy) tolerance for the least achievable value γ_{opt} of γ. The standard solution approach is a bisection-based γ-iteration method, where the solution of the 2-block problem is approximated by successively computed γ-suboptimal solutions of appropriately defined 1-block problems [12].

Let γ_l and γ_u be lower and upper bounds for γ_{opt}, respectively. Such bounds can be computed, for example, as

$$\gamma_l = \|G_2(\lambda)\|_\infty, \quad \gamma_u = \|[\,G_1(\lambda)\; G_2(\lambda)\,]\|_\infty\,. \tag{84}$$

For a given $\gamma > \gamma_l$, we solve first the stable minimum-phase left spectral factorization problem

$$\gamma^2 I - G_2(\lambda)G_2^\sim(\lambda) = V(\lambda)V^\sim(\lambda), \tag{85}$$

where the spectral factor $V(\lambda)$ is biproper, stable and minimum-phase. Further, we compute the additive decomposition

$$V^{-1}(\lambda)G_1(\lambda) = L_s(\lambda) + L_u(\lambda), \tag{86}$$

where $L_s(\lambda)$ is the stable part and $L_u(\lambda)$ is the unstable part. If $\gamma > \gamma_{opt}$, the suboptimal 2-block problem (83) is equivalent to the suboptimal 1-block problem

$$\left\|V^{-1}(\lambda)\big(G_1(\lambda) - X(\lambda)\big)\right\|_\infty \leq 1 \tag{87}$$

and $\gamma_H := \|L_u^\sim(\lambda)\|_H < 1$. In this case we readjust the upper bound to $\gamma_u = \gamma$. If $\gamma \leq \gamma_{opt}$, then $\gamma_H \geq 1$ and we readjust the lower bound to $\gamma_l = \gamma$. For the bisection value $\gamma = (\gamma_l + \gamma_u)/2$ we redo the factorization (85) and decomposition (86). This process is repeated until $\gamma_u - \gamma_l \leq \varepsilon$.

At the end of iterations, we have either $\gamma_{opt} < \gamma \leq \gamma_u$ if $\gamma_H < 1$ or $\gamma_l < \gamma \leq \gamma_{opt}$ if $\gamma_H \geq 1$, in which case we set $\gamma = \gamma_u$. We compute the stable solution of (87) as

$$X(\lambda) = V(\lambda)(L_s(\lambda) + X_s(\lambda)), \tag{88}$$

where, for any γ_1 satisfying $1 \geq \gamma_1 > \gamma_H$, $X_s(\lambda)$ is the stable solution of the optimal Nehari problem

$$\|L_u(\lambda) - X_s(\lambda)\|_\infty = \|L_u^\sim(\lambda)\|_H. \tag{89}$$

2.17 Approximate Model Matching

The formulation of the approximate model-matching problems is frequently done by formulating suitable error minimization problems in terms of $\mathcal{L}_2$- or $\mathcal{L}_\infty$-norms. For example, the approximate solution of the (left) rational equation $X(\lambda)G(\lambda) = F(\lambda)$ involves the minimization of the norm of the error $\mathcal{E}(\lambda) := F(\lambda) - X(\lambda)G(\lambda)$, while the approximate solution of the (right) rational equation $G(\lambda)X(\lambda) = F(\lambda)$ involves the minimization of the norm of the error $\mathcal{E}(\lambda) := F(\lambda) - G(\lambda)X(\lambda)$. In what follows, we consider either the $\mathcal{L}_\infty$- or $\mathcal{L}_2$-norms of the error, or the $\mathcal{H}_\infty$- or $\mathcal{H}_2$-norms of the error if $\mathcal{E}(\lambda)$ is stable. The corresponding problems are called $\mathcal{L}_\infty$ *model-matching problem* ($\mathcal{L}_\infty$-MMP), $\mathcal{L}_2$-MMP, $\mathcal{H}_\infty$-MMP or $\mathcal{H}_2$-MMP. We will use $\|\cdot\|_{\infty/2}$ to denote either the $\mathcal{L}_\infty$- or $\mathcal{L}_2$-norm, or the $\mathcal{H}_\infty$- or $\mathcal{H}_2$-norm, depending on the context. In practical applications, further conditions may be imposed on the desired solution $X(\lambda)$, as for example stability or properness, in conjunction with a stable $F(\lambda)$.

For the existence of a solution the error norm must be finite. Surprisingly, necessary and sufficient condition for the existence of an optimal solution of the $\mathcal{L}_\infty$-MMP are unknown (at least to the author), but several practice relevant sufficient conditions can be easily established. For example, an optimal solution of the $\mathcal{L}_\infty$-MMP exists if $F(\lambda)$ has no poles in $\partial\mathbb{C}_s$ (e.g., it is stable). Also, an optimal solution of the $\mathcal{L}_2$-MMP exists if $F(\lambda)$ has no poles in $\partial\mathbb{C}_s$ (e.g., it is stable) and additionally it is strictly proper. This result holds for arbitrary $G(\lambda)$ (e.g., even a improper one).

For filter and feedforward controller design applications, a standard formulation of the $\mathcal{H}_\infty$ *model-matching problem* ($\mathcal{H}_\infty$-MMP) is: given $G(\lambda), F(\lambda) \in \mathcal{H}_\infty$, find $X(\lambda) \in \mathcal{H}_\infty$ which minimizes $\|\mathcal{E}(\lambda)\|_\infty$. The $\mathcal{H}_2$ *model-matching problem* ($\mathcal{H}_2$-MMP) has a similar formulation. The following results, taken from [12], provide sufficient conditions for the solvability of the $\mathcal{H}_\infty$-MMP and $\mathcal{H}_2$-MMP in the standard case.

Lemma 14. An optimal solution $X(\lambda)$ of the $\mathcal{H}_\infty$-MMP exists if $G(\lambda)$ has no zeros in $\partial\mathbb{C}_s$.

Lemma 15. An optimal solution $X(\lambda)$ of the $\mathcal{H}_2$-MMP exists if $G(\lambda)$ has no zeros in $\partial\mathbb{C}_s$ and, additionally, in the continuous-time

$$\begin{aligned} &\operatorname{rank} G(\infty) = \operatorname{rank} \begin{bmatrix} G(\infty) \\ F(\infty) \end{bmatrix}, && \text{for a left equation}, \\ &\operatorname{rank} G(\infty) = \operatorname{rank}[\, G(\infty)\ F(\infty)\,], && \text{for a right equation}. \end{aligned} \tag{90}$$

The conditions of Lemma 14 and Lemma 15 are clearly not necessary. For example, in the case of solving the equations $X(\lambda)G(\lambda) = G(\lambda)$ or $G(\lambda)X(\lambda) = G(\lambda)$, the trivial optimal (exact) solution $X(\lambda) = I$ exists for any stable $G(\lambda)$. The rank conditions (90) merely ensures that the optimal solution $X(\lambda)$ of the $\mathcal{H}_2$-MMP can be chosen such that the resulting $\mathcal{E}(\lambda)$ is strictly proper and therefore $\|\mathcal{E}(\lambda)\|_2$ is finite. For the equations $X(\lambda)G(\lambda) = G(\lambda)$ or $G(\lambda)X(\lambda) = G(\lambda)$, they are fulfilled for arbitrary $G(\lambda)$, and the trivial optimal (exact) solution $X(\lambda) = I$ exists for arbitrary stable $G(\lambda)$.

In what follows, we will sketch a general approach for solving approximate MMPs based on transforming the error minimization problems into appropriate *least distance problems* (LDPs), by using a special factorization of $G(\lambda)$. For convenience, we will further assume $F(\lambda)$ stable, but will relax the condition on $G(\lambda)$, where for the stability of the solution, it is only assumed that $G(\lambda)$ has no zeros in $\partial\mathbb{C}_s$ (i.e., it can be unstable and even improper). This condition is

sufficient for the existence of a stable optimal solution of the $\mathcal{L}_\infty$-MMP, as it will be apparent from the constructive solution provided below. In what follows, this will be called the *standard case*, while the *non-standard case* is when $G(\lambda)$ has zeros in $\partial\mathbb{C}_s$. In this case, the resulting optimal solution $X(\lambda)$ may have poles in $\partial\mathbb{C}_s$, which represents a subset of the zeros of $G(\lambda)$ lying in $\partial\mathbb{C}_s$.

We only present a (constructive) computational approach to determine the optimal approximate solution of the left rational equation $X(\lambda)G(\lambda) = F(\lambda)$. However, the same procedure is also applicable to compute the approximate solution of the right rational equation $G(\lambda)X(\lambda) = F(\lambda)$, by applying it to the transposed equation $\widetilde{X}(\lambda)G^T(\lambda) = F^T(\lambda)$ to compute $\widetilde{X}(\lambda)$ and obtain $X(\lambda) = \widetilde{X}^T(\lambda)$.

The first step of the proposed constructive solution of both $\mathcal{L}_\infty$- and $\mathcal{L}_2$-MMPs is the computation of the (extended) quasi-co-outer–coinner factorization of $G(\lambda)$ to reduce these problems to $\mathcal{L}_\infty$- or $\mathcal{L}_2$-LDPs, respectively. Consider the factorization

$$G(\lambda) = \left[\begin{array}{cc} G_o(\lambda) & 0 \end{array}\right] G_i(\lambda) = \left[\begin{array}{cc} G_o(\lambda) & 0 \end{array}\right] \left[\begin{array}{c} G_{i,1}(\lambda) \\ G_{i,2}(\lambda) \end{array}\right] = G_o(\lambda)G_{i,1}(\lambda), \tag{91}$$

where $G_i(\lambda) := \left[\begin{smallmatrix} G_{i,1}(\lambda) \\ G_{i,2}(\lambda) \end{smallmatrix}\right]$ is square and inner and $G_o(\lambda)$ is full column rank (i.e., is left invertible). In the standard case (i.e., when $G(\lambda)$ has no zeros in $\partial\mathbb{C}_s$), $G_o(\lambda)$ has only stable zeros and $G_o(\lambda)$ has a stable left inverse $G_o^{-L}(\lambda)$. In the non-standard case (i.e., when $G(\lambda)$ has zeros in $\partial\mathbb{C}_s$), $G_o(\lambda)$ has unstable zeros (which are the zeros of $G(\lambda)$ in $\partial\mathbb{C}_s$) and any left inverse $G_o^{-L}(\lambda)$ has these zeros as poles. In the standard case, if $G(\lambda)$ is additionally stable, then the resulting $G_o(\lambda)$ is outer. The factorization (91) can be computed using algorithms proposed in [32] in the continuous-time case and [29] for the discrete-time case.

The factorization (91) allows to write successively

$$\begin{aligned} \|\mathcal{E}(\lambda)\|_{\infty/2} &= \|F(\lambda) - X(\lambda)G(\lambda)\|_{\infty/2} \\ &= \left\| \left(F(\lambda)G_i^\sim(\lambda) - X(\lambda)\left[\begin{array}{cc} G_o(\lambda) & 0 \end{array}\right]\right) G_i(\lambda) \right\|_{\infty/2} \\ &= \left\| \left[\begin{array}{cc} \widetilde{F}_1(\lambda) - Y(\lambda) & \widetilde{F}_2(\lambda) \end{array}\right] \right\|_{\infty/2}, \end{aligned}$$

where $Y(\lambda) := X(\lambda)G_o(\lambda) \in \mathcal{H}_\infty$ and

$$F(\lambda)G_i^\sim(\lambda) = \left[\begin{array}{c|c} F(\lambda)G_{i,1}^\sim(\lambda) & F(\lambda)G_{i,2}^\sim(\lambda) \end{array}\right] := \left[\begin{array}{c|c} \widetilde{F}_1(\lambda) & \widetilde{F}_2(\lambda) \end{array}\right].$$

Thus, the problem of computing an approximate solution $X(\lambda)$ which minimizes the error norm $\|\mathcal{E}(\lambda)\|_{\infty/2}$ has been reduced to a LDP to compute the stable solution $Y(\lambda)$ which minimizes the norm $\left\| \left[\begin{smallmatrix} \widetilde{F}_1(\lambda) - Y(\lambda) & \widetilde{F}_2(\lambda) \end{smallmatrix}\right] \right\|_{\infty/2}$. The solution of the original MMP is given by

$$X(\lambda) = Y(\lambda)G_o^{-L}(\lambda),$$

where $G_o^{-L}(\lambda)$ is particular left inverse of $G_o(\lambda)$, determined such that its only unstable poles are the unstable zeros of $G_o(\lambda)$ lying in $\partial\mathbb{C}_s$. It follows, that in the standard case $X(\lambda)$ results always stable, while in the non-standard case $X(\lambda)$ may result unstable, having possibly some poles lying in $\partial\mathbb{C}_s$. However, $X(\lambda)$ may result stable even in the non-standard case if unstable pole-zero cancellations take place when computing $X(\lambda)$ as a consequence of the presence of such zeros in the transfer function matrix $F(\lambda)$.

In general, we have that $\left[\,\widetilde{F}_1(\lambda)\ \widetilde{F}_2(\lambda)\,\right] \notin \mathcal{H}_\infty$. If $\widetilde{F}_2(\lambda)$ is present (i.e., $G(\lambda)$ has no full column rank), we have a *2-block* LDP, while if $G(\lambda)$ has full column rank, then $\widetilde{F}_2(\lambda)$ is not present and we have an *1-block* LDP. The solution of the $\mathcal{H}_\infty$-LDP has been discussed in Section 2.16. Therefore, in what follows, we only shortly discuss the solution approach of the $\mathcal{H}_2$-LDP.

Let $L_s(\lambda)$ be the stable part and let $L_u(\lambda)$ be the unstable part in the additive decomposition

$$\widetilde{F}_1(\lambda) = L_s(\lambda) + L_u(\lambda)\,. \tag{92}$$

In the case of $\mathcal{H}_2$-norm, the solution of the LDP is

$$Y(\lambda) = L_s(\lambda),$$

where $L_s(\lambda)$ is the stable projection in (92). In the continuous-time case, we take the unstable projection $L_u(s)$ strictly proper. With the above choice, the achieved minimum $\mathcal{H}_2$-norm of $\mathcal{E}(\lambda)$ is

$$\|\mathcal{E}(\lambda)\|_2 = \|[\,L_u(\lambda)\ \ \widetilde{F}_2(\lambda)\,]\|_2\,.$$

Since the underlying TFMs are unstable, the $\mathcal{L}_2$-norm is used in the last equation. In the continuous-time case, the error norm $\|\mathcal{E}(s)\|_2$ is finite only if $\widetilde{F}_2(s)$ is strictly proper.

In some applications, it is sufficient to determine a so-called γ-suboptimal solution $X(\lambda)$, which, for a given sub-optimality degree γ, satisfies

$$\|\mathcal{E}(\lambda)\|_\infty \leq \gamma.$$

The choice of γ must satisfy $\gamma > \gamma_{opt}$, where γ_{opt} is the optimal (minimal) value of $\|\mathcal{E}(\lambda)\|_\infty$. The solution approach is similar as in the optimal case, but instead of solving an optimal $\mathcal{H}_\infty$-LDP, a γ-suboptimal $\mathcal{H}_\infty$-LDP is solved, by determining a stable $Y(\lambda)$ which satisfies

$$\left\|\left[\,\widetilde{F}_1(\lambda) - Y(\lambda)\ \widetilde{F}_2(\lambda)\,\right]\right\|_\infty < \gamma.$$

Since no γ-iteration is needed to be performed, the solution of the sub-optimal problem is significantly easier than the solution of the optimal problem.

3 Description of DSTOOLS

This user's guide is intended to provide basic information on the **DSTOOLS** collection for the operation on and manipulation of rational transfer function matrices via their descriptor system realizations as described in Section 2. The notations and terminology used throughout this guide have been introduced and extensively discussed in Chapter 9 of the accompanying book [59], while Chapter 10 also represents the main reference for the implemented computational methods in **DSTOOLS**. Information on the requirements for installing **DSTOOLS** are given in Appendix A.

In this section, we present first a short overview of the existing functions of **DSTOOLS** and then, we give in-depth information on the command syntax of the functions of the **DSTOOLS** collection. To execute the examples presented in this guide, simply copy and paste the presented code sequences into the MATLAB command window.

3.1 Quick Reference Tables

The current release of **DSTOOLS** is version V0.74, dated July 30, 2019. The corresponding `Contents.m` file is listed in Appendix B. This section contains quick reference tables for the functions of the **DSTOOLS** collection. The main M- and MEX-files available in the current version of **DSTOOLS** are listed below by category, with short descriptions.

Demonstration	
`DSToolsdemo`	Demonstration of Descriptor System Tools

System analysis	
`gpole`	Poles of a LTI descriptor system.
`gzero`	Zeros of a LTI descriptor system.
`gnrank`	Normal rank of a transfer function matrix of a LTI system.
`ghanorm`	Hankel norm of a proper and stable LTI descriptor system.
`gnugap`	ν-gap distance between two LTI systems.

Order reduction	
`gir`	Reduced order realizations of a LTI descriptor system.
`gminreal`	Minimal realization of a LTI descriptor system.
`gbalmr`	Balancing-based model reduction of a stable LTI descriptor system.
`gss2ss`	Conversions to SVD-like coordinate forms without non-dynamic modes.

Operations on transfer function matrices	
`grnull`	Right nullspace basis of a transfer function matrix.
`glnull`	Left nullspace basis of a transfer function matrix.
`grange`	Range space basis of a transfer function matrix.
`gcrange`	Coimage space basis of a transfer function matrix.
`grsol`	Solution of the linear rational matrix equation $G(\lambda)X(\lambda) = F(\lambda)$.
`glsol`	Solution of the linear rational matrix equation $X(\lambda)G(\lambda) = F(\lambda)$.
`gsdec`	Generalized additive spectral decompositions.

grmcover1	Right minimum dynamic cover of Type 1 based order reduction.
glmcover1	Left minimum dynamic cover of Type 1 based order reduction.
grmcover2	Right minimum dynamic cover of Type 2 based order reduction.
glmcover2	Left minimum dynamic cover of Type 2 based order reduction.
gbilin	Generalized bilinear transformation.
gbilin1	Transfer functions of commonly used bilinear transformations.

Factorizations of transfer function matrices	
grcf	Right coprime factorization with proper and stable factors.
glcf	Left coprime factorization with proper and stable factors.
grcfid	Right coprime factorization with inner denominator.
glcfid	Left coprime factorization with inner denominator.
gnrcf	Normalized right coprime factorization.
gnlcf	Normalized left coprime factorization.
giofac	Inner-outer and QR-like factorizations of a transfer function matrix.
goifac	Co-outer–coinner and RQ-like factorizations of a transfer function matrix.
grsfg	Right spectral factorization of $\gamma^2 I - G^{\sim}(\lambda)G(\lambda)$.
glsfg	Left spectral factorization of $\gamma^2 I - G(\lambda)G^{\sim}(\lambda)$.

Model-matching problems	
grasol	Approximate solution of the linear rational matrix equation $G(\lambda)X(\lambda) = F(\lambda)$.
glasol	Approximate solution of the linear rational matrix equation $X(\lambda)G(\lambda) = F(\lambda)$.
glinfldp	Solution of the least distance problem $\min_{X(\lambda)} \|[G_1(\lambda) - X(\lambda)\ G_2(\lambda)]\|_\infty$.
gnehari	Generalized Nehari approximation.

Matrix pencils and stabilization	
gklf	Kronecker-like staircase forms of a linear matrix pencil.
gsklf	Special Kronecker-like form of a system matrix pencil.
gsorsf	Specially ordered generalized real Schur form.
gsfstab	Generalized state-feedback stabilization.

SLICOT-based MEX-files	
sl_gstra	Descriptor system coordinate transformations.
sl_gminr	Minimal realization of descriptor systems.
sl_gsep	Descriptor system spectral separations.
sl_gzero	Computation of system zeros and Kronecker structure.
sl_klf	Pencil reduction to Kronecker-like forms.
sl_glme	Solution of generalized linear matrix equations.

3.2 Getting Started

In this section we shortly recall how to construct generalized LTI system objects using commands of the Control System Toolbox (CST) of MATLAB [25] and discuss some basic model conversion

techniques and operations with rational matrices via their descriptor system representations. We also illustrate and compare the functionality of some functions available in the CST and **DSTOOLS** to perform model conversions and manipulations.

3.2.1 Building Generalized LTI Models

To describe generalized LTI systems via their TFM representations, the CST supports two model objects (or classes) called `tf` and `zpk`. The model class `tf` represents the elements of the TFM as ratios of two polynomials, in the form (1). The corresponding constructor command `tf` can be used to build transfer functions for SISO or TFMs for MIMO LTI systems. The model class `zpk` represents the elements of the TFM in a zero-pole-gain (factorized) form as in (30) and the corresponding constructor command is `zpk`.

A straightforward method to enter TFM based models relies on a powerful feature of the CST to connect subsystems. For example, to enter the continuous-time improper TFM

$$G_c(s) = \begin{bmatrix} s^2 & \dfrac{s}{s+1} \\ 0 & \dfrac{1}{s} \end{bmatrix},$$

the following commands can be used

```
s = tf('s');                    % define the complex variable s
Gc = [s^2 s/(s+1); 0 1/s]       % define the 2-by-2 improper Gc(s)
```

Similarly, the discrete-time improper TFM

$$G_d(z) = \begin{bmatrix} z^2 & \dfrac{z}{z-2} \\ 0 & \dfrac{1}{z} \end{bmatrix},$$

with sampling time equal to 0.5, can be entered using the commands

```
z = tf('z',0.5);                % define the complex variable z
Gd = [z^2 z/(z-2); 0 1/z]       % define the 2-by-2 improper Gd(z)
```

To describe LTI systems in state-space form, the model class `ss` is provided jointly with the class constructor commands `dss`, for descriptor systems and `ss` for standard state-space systems (with $E = I$). For example, a descriptor system model of the form (2) can be constructed by entering the system matrices E, A, B, C and D and using the command `dss`. The descriptor system with the state-space realization

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

can be entered using the following commands:

```
A = [ 1 0 0 0 0; 0 1 0 0 0; 0 0 1 0 0; 0 0 0 -1 0; 0 0 0  0 0];
B = [ 0 0 -1 0 0; 0 0 0 1 1]';
C = [ 1 0 0 -1 0; 0 0 0 0 1];
D = [ 0 1; 0 0 ];
E = [ 0 1 0 0 0; 0 0 1 0 0; 0 0 0 0 0; 0 0 0 1 0; 0 0 0 0 1];
sys = dss(A,B,C,D,E);
```

If $E = I$, the system model is a standard state-space model, which can be constructed using the class constructor command `ss`.

A less known aspect of building descriptor system models is the available "freedom" in the CST to allow the construction of descriptor system models with a singular pole pencil $A - \lambda E$. This "freedom" is very questionable and may lead to potential conceptual and computational difficulties. This can be easily illustrated with the following trivial non-regular descriptor system, for which many of the functions of the CST produce misleading warnings or questionable results, as shown by the following sequence of commands:

```
A = 0; E = 0; B = 1; C = 1; D = 0;
syst = dss(A,B,C,D,E);
pole(syst)       % the pole must be NaN, similar to eig(A,E), and not empty!
tf(syst)         % the transfer function is infinite and not NaN!
evalfr(syst,1)   % this evaluation of an infinite frequency response is correct
isproper(syst)   % this test is wrong, because the system is not proper
```

The function `gpole` of **DSTOOLS** can be used to check the regularity of the pole pencil. Both commands `gpole` and `eig` below

```
gpole(syst)
eig(A,E)
```

compute the "correct" value of the pole, which, in this case, is `NaN`.

The **DSTOOLS** collection exclusively deals with regular descriptor models, for which the pole pencil $A - \lambda E$ is regular. All functions of **DSTOOLS** guarantee that the computed results are regular descriptor systems, provided the input systems are regular. We have to stress, that for efficiency reasons, in most of functions of **DSTOOLS**, the regularity condition for the input system data is only *assumed*, but it is not explicitly checked. Therefore, it is likely that some functions, even if they perform without issuing error messages, may still deliver nonsense results if the regularity assumption is not fulfilled by the input system descriptions.

3.2.2 Conversions between LTI Model Representations

Most of functions of **DSTOOLS** accept only state-space system objects as inputs and therefore the input-output models must be converted to a standard or descriptor state-space form. For the above defined TFMs $G_c(s)$ and $G_d(z)$, this conversion can be done simply using

```
sysc = ss(Gc)
sysd = ss(Gd)
```

The resulting state-space realization are usually non-minimal. Incidentally, both of the resulting state-space realizations `sysc` and `sysd` have order 5 and are minimal.

For visualization purposes, often the more compact TFM models are better suited than state-space models. Both class constructor commands `tf` and `zpk` can also serve to explicitly convert a LTI model to `tf` or `zpk` forms, respectively. The resulting transfer-function models computed from state-space models, usually contains uncancelled common factors in the numerator and denominator polynomials of the rational matrix elements, and, therefore, are non-minimal.

To compute minimal realizations, the function `minreal` is available in the CST. This function, applied to transfer-function models, enforces the cancellation of common factors in the numerator and denominator polynomials of each element. However, this function is *only* applicable to proper descriptor systems (also including the case of systems with singular E). When applied to an improper descriptor system, an error message is issued:

```
order(minreal(sysc))
Error using DynamicSystem/minreal (line 53)
The "minreal" command cannot be used for models with more zeros than poles.
```

Unfortunately, the above error message is misleading, since for the system `sysc` with an invertible TFM $G_c(s)$, the number of poles and zeros (counting also the infinite poles and zeros), must coincide in accordance with (33). This can be checked with the functions `gpole` and `gzero` of **DSTOOLS**:

```
POLES = gpole(sysc)
ZEROS = gzero(sysc)
```

which produce the following results:

```
POLES =
   0.0000 + 0.0000i
  -1.0000 + 0.0000i
       Inf + 0.0000i
       Inf + 0.0000i

ZEROS =
  -1.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 0.0000i
       Inf + 0.0000i
```

These results also indicate that the McMillan degree of the system is 4 (recall that the order of the minimal descriptor state-space realization is 5). In contrast, the functions `pole` and `tzero` of the CST, compute only the finite poles and finite zeros, respectively, and provide no hint on the actual McMillan degree.

Alternative functions to compute irreducible and minimal realizations, are, respectively, the functions `gir` and `gminreal` available in **DSTOOLS**. For example, the minimality of the above computed realizations can be checked with the function `gminreal` of **DSTOOLS**:

```
order(gminreal(sysc))  % computes the order of the minimal realization
```

3.2.3 Conversion to Standard State-Space Form

A useful conversion which we discuss separately is the conversion of a descriptor system model of a proper system into a standard state-space model. Specifically, we discuss the conversion of a proper descriptor system of the form

$$\begin{array}{rcl} E\lambda x(t) & = & Ax(t) + Bu(t)\,, \\ y(t) & = & Cx(t) + Du(t)\,, \end{array} \tag{93}$$

with $x(t) \in \mathbb{R}^n$, to a standard state-space system of the form

$$\begin{array}{rcl} \lambda\widetilde{x}(t) & = & \widetilde{A}\widetilde{x}(t) + \widetilde{B}u(t)\,, \\ y(t) & = & \widetilde{C}\widetilde{x}(t) + \widetilde{D}u(t)\,, \end{array} \tag{94}$$

with $\widetilde{x}(t) \in \mathbb{R}^{\tilde{n}}$ and $\tilde{n} \leq n$, and such that the two systems have the same transfer function matrices, i.e.

$$C(\lambda E - A)^{-1}B + D = \widetilde{C}(\lambda I_{\tilde{n}} - \widetilde{A})^{-1}\widetilde{B} + \widetilde{D}.$$

This conversion is usually necessary, to obtain for the designed controllers and filters simpler representations, which are better suited for real-time processing. However, we cautiously recommend to avoid such conversions at early steps of the synthesis procedures, unless it is possible to guarantee that no significant loss of accuracy takes place due to ill-conditioned transformations.

For simplicity, we consider only the case when the descriptor realization $(A - \lambda E, B, C, D)$ is already irreducible. Such realizations can be obtained, for example, using the **DSTOOLS** function `gir`. We further assume that the regular pencil $A - \lambda E$ has r finite eigenvalues and $n - r$ simple eigenvalues at infinity, where r is the rank of E. When E is nonsingular, we can simply choose $\widetilde{x}(t) = x(t)$ and

$$\widetilde{A} = E^{-1}A, \quad \widetilde{B} = E^{-1}B, \quad \widetilde{C} = C, \quad \widetilde{D} = D,$$

or alternatively choose $\widetilde{x}(t) = Ex(t)$ and

$$\widetilde{A} = AE^{-1}, \quad \widetilde{B} = B, \quad \widetilde{C} = CE^{-1}, \quad \widetilde{D} = D\,.$$

In these conversion formulas, the inverse of E is explicitly involved and, therefore, severe loss of accuracy can occur if the condition number $\kappa(E) := \|E\|_2\|E^{-1}\|_2$ is large.

A numerically better conversion approach is to use the *singular value decomposition* (SVD) $E = U\Sigma V^T$, with U and V orthogonal matrices and Σ a diagonal matrix whose diagonal elements are the decreasingly ordered singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$. We can choose $\widetilde{x}(t) = \Sigma^{\frac{1}{2}}V^T x(t)$ and

$$\widetilde{A} = \Sigma^{-\frac{1}{2}}U^T AV\Sigma^{-\frac{1}{2}}, \quad \widetilde{B} = \Sigma^{-\frac{1}{2}}U^T B, \quad \widetilde{C} = CV\Sigma^{-\frac{1}{2}}, \quad \widetilde{D} = D\,. \tag{95}$$

From the SVD of E, we can easily compute the condition number $\kappa(E) = \sigma_1/\sigma_n$, and thus have a rough estimation of potential loss of accuracy induced by using the above transformation.

It is also possible to use the QR-decomposition $E = QR$, with Q orthogonal and R upper-triangular and with positive diagonal elements (this can always be arranged using an orthogonal diagonal scaling matrix). Let $R^{\frac{1}{2}}$ be the upper-triangular square root of R, which can be

computed using the method described in [19, Algorithm 6.7], which is also implemented in the MATLAB function `sqrtm`. We can choose $\widetilde{x}(t) = R^{\frac{1}{2}}x(t)$ and

$$\widetilde{A} = R^{-\frac{1}{2}}Q^TAR^{-\frac{1}{2}}, \quad \widetilde{B} = R^{-\frac{1}{2}}Q^TB, \quad \widetilde{C} = CR^{-\frac{1}{2}}, \quad \widetilde{D} = D\,. \tag{96}$$

From the QR-decomposition of E, we can easily compute the condition number $\kappa(E) = \kappa(R)$ (e.g., by using `rcond(R)` to estimate the reverse condition number $1/\kappa(R)$). In this way, we can have a rough estimation of potential loss of accuracy induced by using the above transformation. Although this method can be generally used, its primary use is when the original pair (A, E) is already in a GRSF, with A upper quasi-triangular and E upper triangular. In this case, the resulting $\widetilde{A}$ is in a RSF.

More involved transformation is necessary when E is singular, with rank $E = r < n$. In this case, we can employ the singular value decomposition of E in the form

$$E = U\Sigma V^T := \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \widetilde{\Sigma} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^T, \tag{97}$$

where $\widetilde{\Sigma}$ is a nonsingular diagonal matrix of order $\tilde{n} := r$ with the nonzero singular values of E on the diagonal, and U and V are compatibly partitioned orthogonal matrices. If we apply a system similarity transformation with the transformation matrices

$$\widetilde{U} = \operatorname{diag}\left(\widetilde{\Sigma}^{-\frac{1}{2}}, I_{n-r}\right)U^T, \quad \widetilde{V} = V\operatorname{diag}\left(\widetilde{\Sigma}^{-\frac{1}{2}}, I_{n-r}\right)$$

we obtain

$$\widetilde{U}(A - \lambda E)\widetilde{V} = \begin{bmatrix} A_{11} - \lambda I_r & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad \widetilde{U}B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \quad C\widetilde{V} = \begin{bmatrix} C_1 & C_2 \end{bmatrix}, \tag{98}$$

where A_{22} is nonsingular, due to the assumption of only simple infinite eigenvalues of the regular pencil $A - \lambda E$. The above transformed matrices correspond to the coordinate transformation $\overline{x} = \widetilde{V}^{-1}x(t)$ and lead to the partitioned system representation

$$\begin{array}{rcl} \lambda\overline{x}_1(t) & = & A_{11}\overline{x}_1(t) + A_{12}\overline{x}_2(t) + B_1u(t)\,, \\ 0 & = & A_{21}\overline{x}_1(t) + A_{22}\overline{x}_2(t) + B_2u(t)\,, \\ y(t) & = & C_1\overline{x}_1(t) + C_2\overline{x}_2(t) + Du(t)\,, \end{array}$$

where $\overline{x}(t) = \begin{bmatrix} \overline{x}_1(t) \\ \overline{x}_2(t) \end{bmatrix}$ is partitioned such that $\overline{x}_1(t) \in \mathbb{R}^r$ and $\overline{x}_2(t) \in \mathbb{R}^{n-r}$. We can solve the second (algebraic) equation for $\overline{x}_2(t)$ to obtain

$$\overline{x}_2(t) = -A_{22}^{-1}A_{21}\overline{x}_1(t) - A_{22}^{-1}B_2u(t)$$

and arrive to a standard system representation with $\widetilde{x}(t) = \overline{x}_1(t)$ and the corresponding matrices

$$\begin{array}{ll} \widetilde{A} = A_{11} - A_{12}A_{22}^{-1}A_{21}, & \widetilde{B} = B_1 - A_{12}A_{22}^{-1}B_2, \\ \widetilde{C} = C_1 - C_2A_{22}^{-1}A_{21}, & \widetilde{D} = D - C_2A_{22}^{-1}B_2. \end{array} \tag{99}$$

In this case, if any of the condition numbers $\kappa(\widetilde{\Sigma})$ or $\kappa(A_{22})$ is large, potential accuracy losses can be induced by the conversion to a standard state-space form.

The conversion formulas (95) and (97)-(99) underly the implementation of the function `gss2ss` of **DSTOOLS** (used as default options). The formulas (96) are used, by default, if the input pair (A, E) is in a GRSF, with E invertible. The CST function `isproper` has a hidden input flag (`'explicit'`), which allows to convert proper descriptor models to a standard state-space form.

3.2.4 Sensitivity Issues for Polynomial-Based Representations

The ill-conditioning of polynomial-based representations was a constant discussion subject in the control literature to justify the advantage of using state-space realization-based models for numerical computations. For an authoritative discussion see [39]. More details on these issues are provided in Chapter 6 of [41].

The extreme sensitivity of roots of polynomials with respect to small variations in the coefficients, illustrated in the following example, is well known in the literature and is inherent for polynomial-based representations above a certain degree (say $n > 10$). Therefore, all algorithms which involve rounding errors are doomed to fail by giving results of extremely poor accuracy when dealing with an ill-conditioned polynomial. This potential loss of accuracy is one of the main reasons why polynomial-based system representations with rational or polynomial matrices are generally not suited for numerical computations.

It is well known that polynomials with multiple roots are very sensitive to small variations in the coefficients. However, it is less known that this large sensitivity may be present even in the case of polynomials with well separated roots, if the order of the polynomial is sufficiently large. This will be illustrated by the following example.

Example 1. The simple transfer function

$$g(s) = \frac{1}{(s+1)(s+2)\cdots(s+25)} = \frac{1}{s^{25} + 325s^{24} + \cdots + 25!}$$

has the exact poles $\{-1, -2, \ldots, -25\}$. The denominator is a modification coined by Daniel Kressner (private communication) of the famous Wilkinson polynomial analyzed in [68] (originally of order 20 and with positive roots). This polynomial has been used in many works to illustrate the pitfalls of algorithms for computing eigenvalues of a matrix by computing the roots of its characteristic polynomial.

If we explicitly construct the transfer function $g(s)$ and compute its poles using the MATLAB commands

```
g = tf(1,poly(-25:1:-1));
sys = ss(g);
pole(sys)
```

inaccurate poles with significant imaginary parts result, as can be observed from Fig. 1. For example, instead the poles at -19 and -20, two complex conjugate poles at $-19.8511 \pm 3.2657\mathrm{i}$ result.[2]

The main reason for these inaccurately computed poles is the high sensitivity of the polynomial roots to small variations in the coefficients. In this case, inherent truncations take place in representing the large integer coefficients due to the finite representation with 16 accurate digits

[2]Computed with MATLAB R2015b Version, running under 64-Bit Microsoft Windows 10

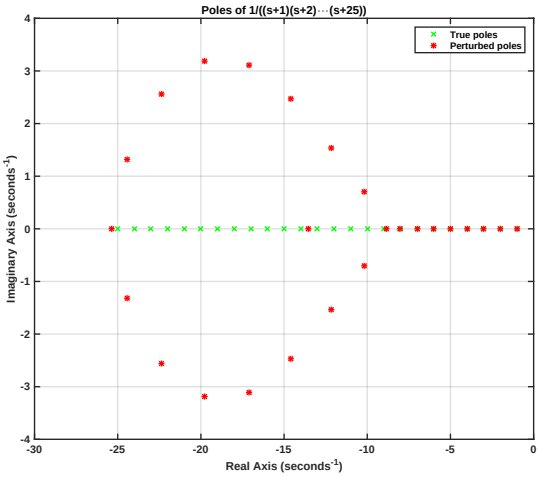

Figure 1: Example for high sensitivity of polynomial poles

of double-precision floating-point numbers. For example, the constant term in the denominator $25! \approx 1.55 \cdot 10^{25}$ has 25 decimal digits, thus can not be exactly represented with 16 digits precision. While the relative error in representing 25! is of the order of the machine-precision $\varepsilon_M \approx 10^{-16}$, the absolute error is of the order of 10^9!

For this particular example, the zero-pole-gain based representation is possibly better suited as starting point for building a state-space realization. For example, a state-space realization of $g(s)$ can be constructed, which still preserves the full accuracy of poles, as shown in the following example:

```
s = zpk('s'); g = 1; for i=1:20, g = g/(s+i); end
sys = ss(g);
pole(sys)
```

Interestingly, the following command sequence builds a 60-th order descriptor system realization of $g(s)$, which contains 40 non-dynamics modes. After eliminating the non-dynamic modes and converting the realization to a standard state-space model, the full accuracy of poles is still preserved to machine precision.

```
s = ss(tf('s')); g = 1; for i=1:20, g = g/(s+i); end
sys = gss2ss(g);
pole(sys)
```

$\Diamond$

This example illustrates that using descriptor system models for model building may occasionally alleviate the numerical difficulties which are inherent when using polynomial based models.

3.2.5 Operations with Rational Matrices

In this section we succinctly discuss the operations with rational matrices via their descriptor system realizations. These operations play an important role in implementing many of the functions available in **DSTOOLS** and therefore, we present some illustrations of the usage of these operations. Let $G(\lambda)$ be a rational TFM having the descriptor system realization $(A-\lambda E, B, C, D)$. We assume that two system objects have been defined to represent $G(\lambda)$: the transfer-function model `g` (e.g., defined with either the `tf` or `zpk` constructors) and the state-space model `sys` (e.g., defined either with the `ss` or the `dss` constructors).

The *transpose* $G^T(\lambda)$ and the corresponding *dual* descriptor system realization $G^T(\lambda) = (A^T-\lambda E^T, C^T, B^T, D^T)$ can be simply computed as `g.'` and `sys.'`, respectively. This operation can be useful to implement, for example, a left oriented function by using an already available right oriented function. For example, to compute a left coprime factorization of $G(\lambda)$ as $G(\lambda) = M^{-1}(\lambda)N(\lambda)$, the function `glcf` calls the function `grcf` to compute a right coprime factorization of the transpose as $G^T(\lambda) = \widetilde{N}(\lambda)\widetilde{M}^{-1}(\lambda)$ and sets $N(\lambda) = \widetilde{N}^T(\lambda)$ and $M(\lambda) = \widetilde{M}^T(\lambda)$. An interesting aspect of computing right coprime factorizations with the function `grcf` is that both resulting descriptor realizations of $\widetilde{N}(\lambda) = (\widetilde{A}_N - \lambda\widetilde{E}_N, \widetilde{B}_N, \widetilde{C}_N, \widetilde{D}_N)$ and $\widetilde{M}(\lambda) = (\widetilde{A}_M - \lambda\widetilde{E}_M, \widetilde{B}_M, \widetilde{C}_M, \widetilde{D}_M)$ have the resulting pairs $(\widetilde{A}_N, \widetilde{E}_N)$ and $(\widetilde{A}_M, \widetilde{E}_M)$ in a GRSF (i.e., with the pole pencils upper quasi-triangular). This condensed form of the pole pencils may be useful for further computations, where this structure can be efficiently exploited (e.g., the eigenvalues can be computed at no cost using the function `ordeig` or the function `grcf` can be applied a second time, with a significantly less computational burden). Therefore, it is highly desirable that the left oriented function `glcf` preserves the upper quasi-triangular shape of the pole pencils of the factors. However, by simply forming the transposed matrices of the dual descriptor systems, this useful property is lost. To preserve the upper quasi-triangular form of the pole pencil of the dual system, an alternative realization of the dual system can be constructed as $G^T(\lambda) = (PA^TP - \lambda PE^TP, PC^T, B^TP, D^T)$, where P is the (orthogonal) permutation matrix with ones down on the secondary diagonal

$$P = \begin{bmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{bmatrix}.$$

The following sequence of commands is used in the function `glcf` to efficiently build permuted dual realizations using the function `xperm` of the CST:

```
% apply grcf to the dual system, by trying to preserve upper-triangular shapes
[sysn,sysm] = grcf(xperm(sys,order(sys):-1:1).',options);
% build dual factors, by preserving upper-triangular shapes
sysn = xperm(sysn,order(sysn):-1:1).';
sysm = xperm(sysm,order(sysm):-1:1).';
```

If $G(\lambda)$ is invertible, then an inversion free realization of the *inverse* TFM $G^{-1}(\lambda)$ is given by

$$G^{-1}(\lambda) = \left[\begin{array}{cc|c} A-\lambda E & B & 0 \\ C & D & I \\ \hline 0 & -I & 0 \end{array}\right]. \tag{100}$$

This realization is not minimal, even if the original realization is minimal. If D is invertible, then an alternative realization of the inverse is

$$G^{-1}(\lambda) = \left[\begin{array}{c|c} A - BD^{-1}C - \lambda E & -BD^{-1} \\ \hline D^{-1}C & D^{-1} \end{array}\right], \tag{101}$$

which is minimal if the original realization is minimal.

To compute inverse systems, the overloaded function `inv` can be used to compute the inverse systems as `inv(g)` or `inv(sys)`. The function `inv` applied to a descriptor state-space model always builds the realization (100) of the inverse (even if $E = I$), while for standard state-space systems (with $E = I$), the realization (101) is used, provided D is reasonably well-conditioned with respect to the inversion. For transfer-function models, an automatic conversion to state-space form is performed, and the computed inverse is converted back to the transfer-function form. These conversions often lead to non-minimal representations containing uncancelled factors between the numerator and denominators of the matrix elements.

Operations involving inverses can often be performed using the overloaded matrix operators `\` (left divide) or `/` (right divide), as for example, to compute `sys1\sys2` or `sys1/sys2` for two systems `sys1` and `sys2`. However, these operation do not avoid the explicit building of the inverses, as can be seen by performing `sys1/sys1` or `sys2\sys2`, which, instead of producing a non-dynamic systems with the direct feedthrough gain equal to the identity matrix, determine realizations of orders (at least) double of the orders of `sys1` or `sys2`. Alternative computation of `sys1\sys2` can be done using the function `glsol` of **DSTOOLS**, while for the evaluation of `sys1/sys2`, the function `grsol` of **DSTOOLS** can be used. These functions are generally applicable to solve compatible systems of linear rational matrix equations and always determine minimal order realizations of the solutions.

The *conjugate* (or *adjoint*) TFM $G^{\sim}(\lambda)$ is defined in the continuous-time case as $G^{\sim}(s) = G^T(-s)$ and has the realization

$$G^{\sim}(s) = \left[\begin{array}{c|c} -A^T - sE^T & C^T \\ \hline -B^T & D^T \end{array}\right],$$

while in the discrete-time case $G^{\sim}(z) = G^T(1/z)$ and has the realization

$$G^{\sim}(z) = \left[\begin{array}{cc|c} E^T - zA^T & 0 & -C^T \\ zB^T & I & D^T \\ \hline 0 & I & 0 \end{array}\right].$$

If $G(z)$ has a standard state-space realization (A, B, C, D) with A invertible, then an alternative realization of $G^{\sim}(z)$ is

$$G^{\sim}(z) = \left[\begin{array}{c|c} A^{-T} - zI & -A^{-T}C^T \\ \hline B^TA^{-T} & D^T - B^TA^{-T}C^T \end{array}\right].$$

This realization is only recommended if A is well-conditioned with respect to the inversion.

The conjugate of a transfer function model `g` can be simply computed as `g'`, while for a state-space model `sys` with `sys'`. For a discrete-time state-space model, the conjugate system is always a descriptor system, which is usually non-minimal (frequently contains non-dynamic modes).

3.3 Functions for System Analysis

The system analysis functions cover the computation of poles and zeros, of normal rank and Hankel norm of the transfer function matrix of a LTI descriptor system.

3.3.1 `gpole`

Syntax

```
[POLES,INFO] = gpole(SYS)
[POLES,INFO] = gpole(SYS,TOL)
[POLES,INFO] = gpole(SYS,TOL,OFFSET)
```

Description

`gpole` computes for a LTI descriptor system, the finite and infinite zeros of the pole pencil, and provides information related to the pole pencil structure.

Input data

`SYS` is a LTI system in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{102}$$

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

`OFFSET` is the stability boundary offset β, to be used to assess the finite eigenvalues which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite eigenvalues having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite eigenvalues having moduli in the interval $[1-\beta, 1+\beta]$. (Default: $\beta = 1.4901 \cdot 10^{-08}$).

Output data

`POLES` is a complex column vector which contains the zeros (finite and infinite) of the *pole pencil* $A - \lambda E$. These are the poles of the TFM of `SYS` if the pencil $A - \lambda E$ is regular and the descriptor realization $(A - \lambda E, B, C, D)$ is irreducible. If the pencil $A - \lambda E$ is not regular, a number of components of `POLES` equal to the rank deficiency of $A - \lambda E$ are set to `NaN`.

`INFO` is a MATLAB structure containing additional structural information related to the pole pencil $A - \lambda E$, as follows:

INFO fields	Description
nfev	number of finite eigenvalues of the pencil $A - \lambda E$, and also the number of finite poles of SYS if SYS is irreducible;
niev	number of infinite eigenvalues of the pencil $A - \lambda E$;
nisev	number of simple infinite eigenvalues of the pencil $A - \lambda E$ (also the number of non-dynamic modes);
nip	number of infinite zeros of the pencil $A - \lambda E$, and also the number of infinite poles of the system SYS if SYS is irreducible;
nfsev	number of (stable) finite eigenvalues in $\mathbb{C}_s$ (the stability domain);
nfsbev	number of finite eigenvalues in $\partial\mathbb{C}_s$ (the boundary of the stability domain);
nfuev	number of (unstable) finite eigenvalues in $\mathbb{C}_u$ (the open instability domain without including infinity);
nhev	number of hiden eigenvalues of the pencil $A - \lambda E$ (can be nonzero only if the pencil $A - \lambda E$ is singular, see **Method**);
nrank	normal rank of the pencil $A - \lambda E$;
miev	integer row vector, which contains the multiplicities of the infinite eigenvalues of the pencil $A - \lambda E$ (also the dimensions of the elementary infinite blocks in the Kronecker form of $A - \lambda E$);
mip	integer row vector, which contains the information on the multiplicities of the infinite zeros of the pencil $A - \lambda E$ as follows: $A - \lambda E$ has INFO.mip(i) infinite zeros of multiplicity i. INFO.mip is empty if $A - \lambda E$ has no infinite zeros.
kr	integer row vector, which contains the right Kronecker indices of the pencil $A - \lambda E$. For a regular pencil, this vector is empty.
kl	integer row vector, which contains the left Kronecker indices of the pencil $A - \lambda E$. For a regular pencil, this vector is empty.
regular	logical value, which is set to true if the pencil $A - \lambda E$ is regular, or to false, if the pencil $A - \lambda E$ is singular.
proper	logical value, which is set to true if the pencil $A - \lambda E$ is regular and all its infinite eigenvalues are simple (has only non-dynamic modes), or to false, if the pencil $A - \lambda E$ is singular or it is regular, but has non-simple infinite eigenvalues.
stable	logical value, which is set to true if the pencil $A - \lambda E$ is regular and has only stable finite eigenvalues and all its infinite eigenvalues are simple (has only non-dynamic modes), or to false, if the pencil $A - \lambda E$ is singular or it is regular, but has unstable finite eigenvalues or non-simple infinite eigenvalues.

Method

Let $G(\lambda)$ be the TFM $G(\lambda) = C(\lambda E - A)^{-1}B + D$ of the LTI system SYS. For the definition of the poles of $G\lambda$) in terms of the descriptor realization (102), see Section 2.5. If the descriptor system realization $(A - \lambda E, B, C, D)$ of SYS has a regular pole pencil $A - \lambda E$ and is irreducible (i.e., controllable and observable), then the computed finite poles in POLES are simply the finite

generalized eigenvalues of the pair (A, E), and the multiplicities of the infinite generalized eigenvalues of (A, E) are in excess with one with respect to the multiplicities of the infinite poles. For the computation of the eigenvalues of $A - \lambda E$ and the information related to its Kronecker structure, the zeros computation algorithm of [26] (see also [36]) is applied to the particular system matrix pencil $S(\lambda) := A - \lambda E$ (i.e., of a system without inputs and outputs), by calling the MEX-function `sl_gzero`. This algorithm determines the following structural information related to the pencil $A - \lambda E$:

- n_f finite zeros of $A - \lambda E$, λ_i, $i = 1, \ldots, n_f$ (n_f is provided in `INFO.nfev`);
- the dimensions s_i^∞, for $i = 1, \ldots, h$, of the elementary infinite Jordan blocks in (10) in the Weierstrass canonical form of the regular part (15), representing the multiplicities of infinite eigenvalues; the number of infinite eigenvalues is $n_\infty = \sum_{i=1}^{h} s_i^\infty$, of which n_∞^0 are simple infinite eigenvalues for which $s_i^\infty = 1$ (n_∞ is provided in `INFO.niev`, n_∞^0 is provided in `INFO.nsiev`, and the multiplicities of the infinite eigenvalues are provided in `INFO.miev`);
- the numbers m_i^∞, for $i = 1, \ldots, k$, where m_i^∞ is the number of the elementary infinite Jordan blocks of order $i + 1$ in (10) in the Weierstrass canonical form of the regular part (15); the number of infinite zeros is $n_{p,\infty} = \sum_{i=1}^{k} i m_i^\infty$ (also equal to $n_\infty - n_\infty^0$) ($n_{p,\infty}$ is provided in `INFO.nip` and the numbers m_i^∞, for $i = 1, \ldots, k$, are provided in `INFO.mip`);
- ν_r right Kronecker indices $\epsilon_i = 0$, for $i = 1, \ldots, \nu_r$ of the pencil $A - \lambda E$, corresponding to the ν_r Kronecker blocks $L_{\epsilon_i}(\lambda)$ of the form $\epsilon_i \times (\epsilon_i + 1)$, which are part of the Kronecker-form of the pencil $A - \lambda E$, as in (13) (the right Kronecker indices are provided in `INFO.kr`);
- ν_l left Kronecker indices $\eta_i = 0$, for $i = 1, \ldots, \nu_l$ of the pencil $A - \lambda E$, corresponding to the ν_l Kronecker blocks $L_{\eta_i}^T(\lambda)$ of the form $(\eta_i + 1) \times \eta_i$, which are part of the Kronecker-form of the pencil $A - \lambda E$, as in (16) (the left Kronecker indices are provided in `INFO.kl`);
- $\rho := n_f + n_\infty + \sum_{i=1}^{\nu_r} \epsilon_i + \sum_{i=1}^{\nu_l} \eta_i$, the normal rank of the n-th order pencil $A - \lambda E$, where $n - \rho$ is the rank deficiency of $A - \lambda E$ (ρ is provided in `INFO.nrank`).

The multiplicities of the infinite zeros are related to the orders of the elementary infinite blocks, as follows: to each elementary infinite block of order $s_i^\infty > 1$ corresponds an infinite zero of multiplicity $s_i^\infty - 1$.

The presence of the right or left Kronecker indices indicates that the pencil $A - \lambda E$ is singular. In this case, a number of $\nu_r + \nu_l$ generalized eigenvalues of the pair (A, E) are *hidden* (their values depend on the employed transformation matrices) and a number of eigenvalues corresponding to the rank deficiency of $A - \lambda E$ are not defined, because would involve forming the fractions $\frac{0}{0}$. Therefore, the components of the vector `POLES` consist in general of: n_f finite values λ_i, $i = 1, \ldots, n_f$; n_∞ infinite values; and $n - \rho$ values set to `NaN`. The hidden eigenvalues are not included in `POLES`. *Note: The term hidden eigenvalue is not standard in the literature and has been introduced only for convenience. Some hidden eigenvalues may explicitly appear when directly using the function* `eig` *as* `eig(SYS.a,SYS.e)`. *Interestingly,* `eig(SYS.a.',SYS.e.')` *may produce different hidden eigenvalues!*

The number of finite stable eigenvalues of $A - \lambda E$ lying in $\mathbb{C}_s$ (the appropriate open stability domain) is provided in `INFO.nfsev`. The number of finite unstable eigenvalues of $A - \lambda E$ lying in $\mathbb{C}_u$ (the appropriate open instability domain without including infinity) is provided in

`INFO.nfuev`. The number of finite eigenvalues of $A - \lambda E$ lying in $\partial\mathbb{C}_s$ (the boundary of the appropriate stability domain) is provided in `INFO.nfsbev`. If these eigenvalues have multiplicity one, they are also called the *marginally stable poles* of the system `SYS`. The stability boundary offset β specified in `OFFSET` is used to numerically assess if an eigenvalue belongs or not to $\mathbb{C}_s$, $\partial\mathbb{C}_s$ or $\mathbb{C}_u$.

Additional qualitative information is provided in the `INFO` structure, to characterize some properties related to system poles:

- `INFO.regular` is set to `true` if $A-\lambda E$ is a regular pencil, and is set to `false` for a singular $A - \lambda E$ (in which case $\rho < n$ and $\det(A - \lambda E) \equiv 0$).
- `INFO.proper` is set to `true` if $A - \lambda E$ is a regular pencil and all its infinite eigenvalues are simple. This property characterizes a *proper* system `SYS`, provided the descriptor realization $(A - \lambda E, B, C, D)$ is irreducible. Otherwise, `INFO.proper` is set to `false` (i.e., for a singular $A - \lambda E$ or if non-simple infinite eigenvalues are present).
- `INFO.stable` is set to `true` for an exponentially stable system, for which $A - \lambda E$ is a regular pencil, with all its finite eigenvalues belonging to the stability domain $\mathbb{C}_s$, and with only simple infinite eigenvalues. Otherwise, `INFO.stable` is set to `false`.

3.3.2 `gzero`

Syntax

```
[Z,INFO] = gzero(SYS)
[Z,INFO] = gzero(SYS,TOL)
[Z,INFO] = gzero(SYS,TOL,OFFSET)
```

Description

`gzero` computes for a LTI descriptor system, the finite and infinite zeros of the system matrix pencil, and provides information related to the system matrix pencil structure.

Input data

`SYS` is a LTI system in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{103}$$

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

`OFFSET` is the stability boundary offset β, to be used to assess the finite zeros which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite zeros having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite zeros having moduli in the interval $[1 - \beta, 1 + \beta]$.
(Default: $\beta = 1.4901 \cdot 10^{-08}$).

Output data

`Z` is a complex column vector which contains the invariant zeros (finite and infinite) of the system matrix pencil

$$S(\lambda) = \begin{bmatrix} A - \lambda E & B \\ C & D \end{bmatrix}. \tag{104}$$

These are also called the transmission zeros of the TFM $G(\lambda)$ of `SYS` if the descriptor realization $(A - \lambda E, B, C, D)$ is irreducible.

`INFO` is a MATLAB structure containing additional structural information related to the system matrix pencil $S(\lambda)$, as follows:

`INFO` fields	Description
`nfz`	number of finite zeros of the pencil $S(\lambda)$;
`niev`	number of infinite eigenvalues of the pencil $S(\lambda)$;
`nisev`	number of simple infinite eigenvalues of the pencil $S(\lambda)$;
`niz`	number of infinite zeros of the pencil $S(\lambda)$;
`nfsz`	number of finite stable zeros in $\mathbb{C}_s$ (the stability domain);
`nfsbz`	number of finite zeros in $\partial\mathbb{C}_s$ (the boundary of the stability domain);
`nfuz`	number of finite unstable zeros in $\mathbb{C}_u$ (the open instability domain without including infinity);
`nrank`	normal rank of the pencil $S(\lambda)$;
`miev`	integer row vector, which contains the multiplicities of the infinite eigenvalues of the pencil $S(\lambda)$ (also the dimensions of the elementary infinite blocks in the Kronecker form of $S(\lambda)$);
`miz`	integer row vector, which contains the information on the multiplicities of the infinite zeros of the pencil $S(\lambda)$ as follows: $S(\lambda)$ has `INFO.miz`(i) infinite zeros of multiplicity i. `INFO.miz` is empty if $S(\lambda)$ has no infinite zeros.
`kr`	integer row vector, which contains the right Kronecker indices of the pencil $S(\lambda)$.
`kl`	integer row vector, which contains the left Kronecker indices of the pencil $S(\lambda)$.
`minphase`	logical value, which is set to `true` if the pencil $S(\lambda)$ has only stable finite eigenvalues and all its infinite eigenvalues are simple, or to `false`, if the pencil $S(\lambda)$ has unstable finite eigenvalues or infinite zeros.

Method

Let $G(\lambda)$ be the TFM $G(\lambda) = C(\lambda E - A)^{-1}B + D$ of the LTI system `SYS`. For the definition of the zeros of $G\lambda)$ in terms of the descriptor realization (103), see Section 2.5. If the descriptor system realization $(A - \lambda E, B, C, D)$ of `SYS` is irreducible (i.e., controllable and observable), then the computed finite zeros in `Z` are simply the finite generalized eigenvalues of the system matrix pencil $S(\lambda)$, and the multiplicities of the infinite generalized eigenvalues of $S(\lambda)$ are in excess with one with respect to the multiplicities of the infinite zeros. For the computation of

the eigenvalues of $S(\lambda)$ and its Kronecker structure, the zeros computation algorithm of [26] is applied to the system matrix pencil $S(\lambda)$ in (35), by calling the MEX-function `sl_gzero`. In the case of a standard state-space model with $E = I$, `sl_gzero` uses the algorithm of [10] in conjunctions with the extension proposed in [36]. These algorithms determine:

- the n_f finite zeros of $S(\lambda)$, λ_i, $i = 1, \ldots, n_f$ (n_f is provided in `INFO.nfz`);
- the dimensions s_i^∞, for $i = 1, \ldots, h$, of the elementary infinite Jordan blocks in (10) in the Weierstrass canonical form of the regular part (15), representing the multiplicities of infinite eigenvalues; the number of infinite eigenvalues is $n_\infty = \sum_{i=1}^{h} s_i^\infty$, of which n_∞^0 are simple infinite eigenvalues for which $s_i^\infty = 1$ (n_∞ is provided in `INFO.niev`, n_∞^0 is provided in `INFO.nsiev`, and the multiplicities of the infinite eigenvalues are provided in `INFO.miev`);
- the numbers m_i^∞, for $i = 1, \ldots, k$, where m_i^∞ is the number of the elementary infinite Jordan blocks of order $i + 1$ in (10) in the Weierstrass canonical form of the regular part (15); the number of infinite zeros is $n_{z,\infty} = \sum_{i=1}^{k} i m_i^\infty$ (also equal to $n_\infty - n_\infty^0$) ($n_{z,\infty}$ is provided in `INFO.niz` and the numbers m_i^∞, for $i = 1, \ldots, k$, are provided in `INFO.miz`);
- the ν_r right Kronecker indices ϵ_i, for $i = 1, \ldots, \nu_r$, of the pencil $S(\lambda)$ (corresponding to the ν_r Kronecker blocks $L_{\epsilon_i}(\lambda)$ of the form $\epsilon_i \times (\epsilon_i + 1)$ which are part of the Kronecker canonical form of the pencil $S(\lambda)$, as in (13)) (the right Kronecker indices are provided in `INFO.kr`);
- the ν_l left Kronecker indices η_i, for $i = 1, \ldots, \nu_l$, of the pencil $S(\lambda)$ (corresponding to the ν_l Kronecker blocks $L_{\eta_i}(\lambda)$ of the form $\eta_i \times (\eta_i + 1)$ which are part of the Kronecker canonical form of the pencil $S(\lambda)$, as in (16)) (the left Kronecker indices are provided in `INFO.kl`);
- $\rho := n_f + n_\infty + \sum_{i=1}^{\nu_r} \epsilon_i + \sum_{i=1}^{\nu_l} \eta_i$, the normal rank of the pencil $S(\lambda)$ (ρ is provided in `INFO.nrank`).

The multiplicities of the infinite zeros are related to the orders of the elementary infinite blocks, as follows: to each elementary infinite block of order $s_i^\infty > 1$ corresponds an infinite zero of multiplicity $s_i^\infty - 1$.

The number of finite stable zeros of $S(\lambda)$ lying in $\mathbb{C}_s$ (the appropriate open stability domain) is provided in `INFO.nfsz`. The number of finite unstable zeros of $S(\lambda)$ lying in $\mathbb{C}_u$ (the appropriate open instability domain without including infinity) is provided in `INFO.nfuz`. The number of finite zeros of $S(\lambda)$ lying in $\partial\mathbb{C}_s$ (the boundary of the appropriate stability domain) is provided in `INFO.nfsbz`. The stability boundary offset β specified in `OFFSET` is used to numerically assess if an eigenvalue belongs or not to $\mathbb{C}_s$, $\partial\mathbb{C}_s$ or $\mathbb{C}_u$.

The components of the vector `Z` are the n_f finite values λ_i, $i = 1, \ldots, n_f$ and the n_∞ infinite values. Additional qualitative information is provided in the `INFO` structure, to characterize the minimum-phase property: `INFO.minphase` is set to `true` if all finite eigenvalues of $S(\lambda)$ are stable and $S(\lambda)$ has only simple infinite eigenvalues. Otherwise, `INFO.minphase` is set to `false`.

Application examples

The *input decoupling zeros* are defined as the zeros of the particular system matrix pencil

$$S(\lambda) := [\, A - \lambda E \;\; B \,],$$

which corresponds to a system with empty outputs. The finite zeros of $S(\lambda)$ are also known as the uncontrollable finite eigenvalues of the pencil $A-\lambda E$, while the infinite zeros of $S(\lambda)$ corresponds to uncontrollable infinite eigenvalues of the pencil $A - \lambda E$ (their multiplicities exceeds with 1 the multiplicities of the infinite zeros). The function `gzero` can be used to compute the input decoupling zeros of a LTI state-space system `SYS` using

```
[Z,INFO] = gzero(SYS([],:),TOL)
```

For a controllable pair $(A - \lambda E, B)$, `Z` results empty.

Similarly, the *output decoupling zeros* are defined as the invariant zeros of the particular system matrix

$$S(\lambda) := \begin{bmatrix} A - \lambda E \\ C \end{bmatrix},$$

which corresponds to a system with empty inputs. The finite zeros of $S(\lambda)$ are also known as the unobservable finite eigenvalues of the pencil $A-\lambda E$, while the infinite zeros of $S(\lambda)$ corresponds to unobservable infinite eigenvalues of the pencil $A - \lambda E$ (their multiplicities exceeds with 1 the multiplicities of the infinite zeros). The function `gzero` can be used to compute the output decoupling zeros of a LTI state-space system `SYS` using

```
[Z,INFO] = gzero(SYS(:,[]),TOL)
```

For an observable pair $(A - \lambda E, C)$, `Z` results empty.

Finally, the zeros of the pole pencil can be computed as the invariant zeros of the particular system matrix

$$S(\lambda) := A - \lambda E,$$

which corresponds to a system with empty inputs and empty outputs. The function `gzero` can be used to compute the zeros of the pole pencil of a LTI state-space system `SYS` using

```
[Z,INFO] = gzero(SYS([],[]),TOL)
```

For an irreducible realization $(A - \lambda E, B, C, D)$, `Z` contains the poles (finite and infinite) of the system `SYS`.

3.3.3 `gnrank`

Syntax

```
NR = gnrank(SYS)
NR = gnrank(SYS,TOL)
NR = gnrank(SYS,TOL,FREQ)
```

Description

`gnrank` computes for the LTI system `SYS`, the normal rank of its transfer function matrix.

Input data

`SYS` is a LTI system, which can be specified in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{105}$$

or in an input-output form

$$\mathbf{y}(\lambda) = G(\lambda)\mathbf{u}(\lambda)\,, \tag{106}$$

where $G(\lambda)$ is the rational transfer function matrix of the system.

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

`FREQ` is a complex vector containing the frequency values to be used to estimate the normal rank of $G(\lambda)$ (it is recommended to use frequency values which are distinct from the poles and zeros of the system).

Output data

`NR` is the normal rank of the transfer function matrix $G(\lambda)$ of the LTI system `SYS`.

Method

If `SYS` is specified in the descriptor form (105), then its transfer-function matrix is $G(\lambda) = C(\lambda E - A)^{-1}B + D$. If `SYS` is specified in the input-output form (106), then a (possibly non-minimal) state-space realization of the form (105) is automatically constructed. For the definition of the normal rank r of a rational TFM $G(\lambda)$, see Section 2.4. For the calculation of the normal rank r of $G\lambda)$ in terms of the descriptor representation, we use the relation

$$r = \operatorname{rank} S(\lambda) - n,$$

where $\operatorname{rank} S(\lambda)$ is the normal rank of the system matrix pencil $S(\lambda)$ defined as

$$S(\lambda) = \begin{bmatrix} A - \lambda E & B \\ C & D \end{bmatrix} \tag{107}$$

and n is the order of the descriptor state-space realization. For the computation of the normal rank of $S(\lambda)$, different methods are employed, depending on the frequency values contained in `FREQ`.

If `FREQ` is empty or not specified, to determine the normal rank of $S(\lambda)$, the structural elements of its Kronecker structure are determined using the zeros computation algorithm of [26], by calling the MEX-function `sl_gzero`. In the case of a standard state-space model with $E = I$, `sl_gzero` uses the algorithm of [10] in conjunctions with the extension proposed in [36]. These algorithms determine:

- n_f, the number of finite eigenvalues of $S(\lambda)$;
- the ν_r right Kronecker indices ϵ_i, for $i = 1, \ldots, \nu_r$ of the pencil $S(\lambda)$ (corresponding to the ν_r Kronecker blocks $L_{\epsilon_i}(\lambda)$ of the form $\epsilon_i \times (\epsilon_i + 1)$ which are part of the Kronecker canonical form of the pencil $S(\lambda)$, as in (13));

- the ν_l left Kronecker indices η_i, for $i = 1, \ldots, \nu_l$ of the pencil $S(\lambda)$ (corresponding to the ν_l Kronecker blocks $L_{\eta_i}(\lambda)$ of the form $\eta_i \times (\eta_i + 1)$ which are part of the Kronecker canonical form of the pencil $S(\lambda)$, as in (16));
- the orders of the elementary infinite blocks s_i^∞ , for $i = 1, \ldots, h$, (i.e., the dimensions of the elementary infinite Jordan blocks in (10) in the Weierstrass canonical form of the regular part (15)).

The normal rank of $S(\lambda)$ is determined as

$$\operatorname{rank} S(\lambda) = n_f + \sum_{i=1}^{h} s_i^\infty + \sum_{i=1}^{\nu_r} \epsilon_i + \sum_{i=1}^{\nu_l} \eta_i.$$

If `FREQ` is a complex vector with n_f test frequency values, which form the set Ω, them the normal rank of $S(\lambda)$ is determined as

$$\operatorname{rank} S(\lambda) = \max_{\lambda_z \in \Omega} \operatorname{rank} S(\lambda_z). \tag{108}$$

This approach involves n_f singular value decompositions of the matrix $S(\lambda_z)$. To avoid complex computations, the elements of FREQ can be chosen real. Usually a few (one or two) random values are sufficient to obtain the correct value of the rank. Also, choosing a suitable threshold for rank decisions is alleviated by the fact that $S(\lambda_z)$ has usually nonzero rank, thus the use default tolerances based on largest singular value (i.e., the 2-norm $\|S(\lambda_z)\|_2$) is reliable.

Alternative computation of normal rank

The normal rank of a LTI system `SYS` can be alternatively computed by evaluating the rank of the TFM $G(\lambda)$ at a random frequency (or taking the maximum rank for a few random frequencies). The following command can be used for this purpose:

```
nr = rank(evalfr(SYS,rand),TOL)
```

3.3.4 `ghanorm`

Syntax

```
[HANORM,HS] = ghanorm(SYS)
```

Description

`ghanorm` computes for a proper and stable LTI state-space system `SYS` with the transfer function matrix $G(\lambda)$, the Hankel norm $\|G(\lambda)\|_H$ and the Hankel singular values of the system.

Input data

`SYS` is a LTI system, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{109}$$

Output data

`HANORM` is the Hankel norm $\|G(\lambda)\|_H$ of the transfer function matrix $G(\lambda)$ of the LTI system `SYS`.

`HS` is a column vector which contains the decreasingly ordered Hankel singular values of `SYS`.

Method

Let $G(\lambda)$ be the TFM of the LTI `SYS`. For the definition and computation of the Hankel norm of a proper and stable rational TFM $G(\lambda)$ see Section 2.12 and for the definition and computation of the Hankel singular values, see Section 2.14. If the original descriptor system is proper, but E is singular, then an automatic conversion is performed using the function `gss2ss`, to a reduced order descriptor state-space representation with E invertible and upper triangular. For the solution of the intervening generalized Lyapunov equation (73) or generalized Stein equation (74), the MEX-function `sl_glme` is called.

3.3.5 gnugap

Syntax

```
[NUGAP,FPEAK] = gnugap(SYS1,SYS2,TOL)
[NUGAP,FPEAK] = gnugap(SYS1,SYS2,TOL,FREQ)
[NUGAP,FPEAK] = gnugap(SYS1,SYS2,TOL,FREQ,OFFSET)
```

Description

`gnugap` computes for two LTI systems `SYS1` and `SYS2` with the transfer function matrices $G_1(\lambda)$ and $G_2(\lambda)$, respectively, the ν-gap distance $\delta_\nu\big(G_1(\lambda), G_2(\lambda)\big)$ between the two systems.

Input data

`SYS1` is a LTI system, which can be specified in a descriptor system state-space form

$$\begin{aligned} E_1\lambda x_1(t) &= A_1 x_1(t) + B_1 u(t), \\ y_1(t) &= C_1 x_1(t) + D_1 u(t) \end{aligned} \tag{110}$$

or in an input-output form

$$\mathbf{y}_1(\lambda) = G_1(\lambda)\mathbf{u}(\lambda)\,, \tag{111}$$

where $G_1(\lambda)$ is the rational transfer function matrix of the system `SYS1`.

`SYS2` is a LTI system, which can be specified in a descriptor system state-space form

$$\begin{aligned} E_2\lambda x_2(t) &= A_2 x_2(t) + B_2 u(t), \\ y_2(t) &= C_2 x_2(t) + D_2 u(t) \end{aligned} \tag{112}$$

or in an input-output form

$$\mathbf{y}_2(\lambda) = G_2(\lambda)\mathbf{u}(\lambda)\,, \tag{113}$$

where $G_2(\lambda)$ is the rational transfer function matrix of the system `SYS2`.

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

`FREQ` is a real vector with nonnegative elements, which contains a set of frequency values to be used for the point-wise evaluation of the distances. If `FREQ = []` or not specified, no point-wise evaluation is performed.

`OFFSET` is the stability boundary offset β, to be used to assess the finite zeros which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite zeros having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite zeros having moduli in the interval $[1-\beta, 1+\beta]$.
(Default: $\beta = 1.4901 \cdot 10^{-08}$).

Output data

`NUGAP`, if `FREQ = []` or not specified, is the ν-gap distance $\delta_\nu\big(G_1(\lambda), G_2(\lambda)\big)$ between the two systems. If `FREQ` is a non-empty vector with n_f elements, then `NUGAP` is an n_f-dimensional vector, where `NUGAP(`i`)` contains the point-wise distance at i-th frequency value `FREQ(`i`)`.

`FPEAK` is the frequency value where the peak value of $\delta_\nu\big(G_1(\lambda), G_2(\lambda)\big)$ is achieved. `FPEAK = []` if `NUGAP` $= 1$.

Method

The ν-gap metric is defined in Section 2.13 for arbitrary transfer function matrices. The critical aspect of the computation of $\delta_\nu\big(G_1(\lambda), G_2(\lambda)\big)$, using as computational basis the definition (75), is the reliable determination of the winding number of $g(\lambda) := \det\big(R_2^{\sim}(\lambda)R_1(\lambda)\big)$, using the computed poles and zeros of $R_2^{\sim}(\lambda)R_1(\lambda)$, and the checking of the associated conditions on its determinant. For the computation of the poles and zeros, an irreducible realization of $R_2^{\sim}(\lambda)R_1(\lambda)$ is first computed, and then the poles are computed using the function `gpole` and the zeros using the function `gzero`.

3.4 Functions for System Order Reduction

The system order reduction functions cover the computation of irreducible or minimal realizations, the balancing-related model reduction, and the conversion of descriptor systems representation to various SVD-like coordinate forms without non-dynamic modes, including the conversion to standard state-space forms.

3.4.1 `gir`

Syntax

```
SYSR = gir(SYS)
SYSR = gir(SYS,TOL)
SYSR = gir(SYS,TOL,JOBOPT)
```

Description

`gir` computes for a LTI descriptor state-space system $(A - \lambda E, B, C, D)$, a reduced order (e.g., controllable, observable, or irreducible) descriptor realization $(\widetilde{A} - \lambda\widetilde{E}, \widetilde{B}, \widetilde{C}, D)$, such that the corresponding transfer function matrices are equal.

Input data

`SYS` is a LTI system, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{114}$$

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

`JOBOPT` is a character option variable to specify various order reduction options, as follows:

`'irreducible'`	– compute an irreducible descriptor realization (default);
`'finite'`	– compute a finite controllable and finite observable realization
`'infinite'`	– compute an infinite controllable and infinite observable realization
`'contr'`	– compute a controllable realization
`'obs'`	– compute an observable realization
`'finite_contr'`	– compute a finite controllable realization
`'infinite_contr'`	– compute an infinite controllable realization
`'finite_obs'`	– compute a finite observable realization
`'infinite_obs'`	– compute an infinite observable realization

Output data

`SYSR` contains the resulting reduced order system in a descriptor system state-space form

$$\begin{aligned} \widetilde{E}\lambda\widetilde{x}(t) &= \widetilde{A}\widetilde{x}(t) + \widetilde{B}u(t), \\ y(t) &= \widetilde{C}x(t) + Du(t). \end{aligned} \tag{115}$$

`SYSR` has the same TFM as `SYS` and the resulting order of `SYSR` depends on the order reduction option selected via `JOBOPT`. If no order reduction takes place, then `SYSR` has the same realization as `SYS`.

Remark on input and output data

The function `gir` accepts an array of LTI systems `SYS` as input parameter. In this case, `SYSR` is also an array of LTI systems of the same size as `SYS`. To each component system in `SYS(:,:,i,j)` corresponds a component system `SYSR(:,:,i,j)` with a reduced order.

Method

Let $G(\lambda)$ be the TFM of the LTI `SYS` with the descriptor realization (114). The conditions for finite and infinite controllability and observability are given by Theorem 2, as conditions $(i)-(iv)$. The concepts of finite controllable and observable eigenvalues of the pencil $A-\lambda E$ are discussed in Section 2.5. The elimination of uncontrollable or unobservable eigenvalues is done by determining the so-called Kalman controllability or Kalman observability forms, which exhibit explicitly these eigenvalues. For the computation of the controllability Kalman form for a descriptor system representation, the **Procedure GCSF** described in [59] is employed, which employs the orthogonal reduction technique proposed in [44]. This algorithm computes for a pair $(A-\lambda E, B)$, orthogonal transformation matrices Q and Z such that the matrices of the transformed triple $(\widetilde{A}-\lambda\widetilde{E},\widetilde{B},\widetilde{C}) := (Q^TAZ-\lambda Q^TEZ, Q^TB, CZ)$ have the form

$$\widetilde{A}-\lambda\widetilde{E} = \begin{bmatrix} A_c-\lambda E_c & * \\ 0 & A_{\bar{c}}-\lambda E_{\bar{c}} \end{bmatrix}, \quad \widetilde{B} = \begin{bmatrix} B_c \\ 0 \end{bmatrix}, \quad \widetilde{C} = [\, C_c \; * \,],$$

where the pair $(A_c-\lambda E_c, B_c)$ is finite controllable, and $\Lambda(A_{\bar{c}}-\lambda E_{\bar{c}})$ contains the finite uncontrollable eigenvalues (as well as possible some infinite uncontrollable eigenvalues too). The reduced order system $(A_c-\lambda E_c, B_c, C_c, D)$ is finite controllable and has the same TFM $G(\lambda)$ as the original system. In this way, all uncontrollable finite eigenvalues contained in $A_{\bar{c}}-\lambda E_{\bar{c}}$ have been removed from the system model. By applying the same algorithm to the dual realization $(A^T-\lambda E^T, C^T, B^T, D^T)$, the finite unobservable eigenvalues can be removed (as the finite uncontrollable eigenvalues of the dual system). If the matrices A and E are interchanged, by forming a model with $(E-\lambda A, B, C, D)$, then the uncontrollable or unobservable null eigenvalues can be removed using the same algorithms. However, by removing the uncontrollable or unobservable null eigenvalues of this model, we remove in fact the infinite uncontrollable or unobservable eigenvalues of the original model. An irreducible (controllable and observable) realization can be thus computed in four steps, which form the **Procedure GIR** described in [59]. The computational method to compute an irreducible realization or to perform only a specific step of the **Procedure GIR** is implemented in the MEX-function `sl_gminr`, which is called, with appropriately set options, by the function `gir`.

Application example

The function `gir` displays messages indicating the number of uncontrollable and the number of unobservable eigenvalues removed from the model. This verbose output (especially when applied to array of systems) can be disabled as shown in the example below:

```
warning off
sysr = gir(sys);
warning on
```

3.4.2 `gminreal`

Syntax

```
[SYSMIN,INFO] = gminreal(SYS)
[SYSMIN,INFO] = gminreal(SYS,TOL)
[SYSMIN,INFO] = gminreal(SYS,TOL,NDMONLYFLAG)
```

Description

`gminreal` computes for a LTI descriptor state-space system $(A - \lambda E, B, C, D)$, a minimal order descriptor realization $(A_m - \lambda E_m, B_m, C_m, D_m)$ (i.e., controllable, observable, without non-dynamic modes), such that the corresponding transfer function matrices are equal.

Input data

`SYS` is a LTI system, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{116}$$

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

`NDMONLYFLAG` is a character option variable to be set to `'ndmonly'` to remove only the non-dynamic modes (i.e., simple infinite eigenvalues). By default, the non-dynamic modes are jointly removed with the uncontrollable and unobservable eigenvalues.

Output data

`SYSMIN` contains the resulting minimal order system in a descriptor system state-space form

$$\begin{aligned} E_m\lambda x_m(t) &= A_m x_m(t) + B_m u(t), \\ y_m(t) &= C_m x_m(t) + D_m u(t). \end{aligned} \tag{117}$$

`SYSMIN` has the same TFM as `SYS`. If `NDMONLYFLAG = 'ndmonly'` is specified, then the resulting `SYSMIN` contains a reduced order system without non-dynamic modes. If no order reduction takes place, then `SYSMIN` has the same realization as `SYS`.

`INFO(1:3)` contains information on the number of removed eigenvalues, as follows:
`INFO(1)` – the number of removed uncontrollable eigenvalues;
`INFO(2)` – the number of removed unobservable eigenvalues;
`INFO(3)` – the number of removed non-dynamic (infinite) eigenvalues.

Method

The conditions for minimality are given by Theorem 2, as conditions $(i) - (v)$. The concepts of finite controllable and observable eigenvalues of the pencil $A - \lambda E$ are discussed in Section 2.5. The elimination of uncontrollable and unobservable eigenvalues is done in several steps, by determining appropriate Kalman controllability and observability forms, which explicitly exhibit these eigenvalues. The basic computation is the reduction of the descriptor system matrices to the controllability Kalman form using the orthogonal transformation based reduction technique proposed in [44] (see also **Procedure GCSF** described in [59]). For more details on the computation of irreducible realizations, see the description of **Method** for the function `gir`.

To remove the non-dynamic infinite eigenvalues of a descriptor system $(A - \lambda E, B, C, D)$, the system matrices are reduced to a SVD-like coordinate form

$$(\widetilde{A} - \lambda\widetilde{E}, \widetilde{B}, \widetilde{C}, D) := (Q^T AZ - \lambda Q^T EZ, Q^T B, CZ, D),$$

where

$$\widetilde{A}-\lambda\widetilde{E} = \left[\begin{array}{ccc} A_{11}-\lambda E_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & 0 \\ A_{31} & 0 & 0 \end{array}\right], \quad \widetilde{B} = \left[\begin{array}{c} B_1 \\ B_2 \\ B_3 \end{array}\right],$$
$$\widetilde{C} = \left[\begin{array}{ccc} C_1 & C_2 & C_3 \end{array}\right],$$

with E_{11} and A_{22} upper triangular invertible matrices. Then, the reduced descriptor system without non-dynamic modes $(A_m - \lambda E_m, B_m, C_m, D_m)$ is computed as

$$\begin{aligned} A_m-\lambda E_m &= \left[\begin{array}{cc} A_{11}-A_{12}A_{22}^{-1}A_{21}-\lambda E_{11} & A_{13} \\ A_{31} & 0 \end{array}\right], \\ B_m &= \left[\begin{array}{c} B_1-A_{12}A_{22}^{-1}B_2 \\ B_3 \end{array}\right], \\ C_m &= \left[\begin{array}{cc} C_1-C_2A_{22}^{-1}A_{21} & C_3 \end{array}\right], \\ D_m &= D-C_2A_{22}^{-1}B_2\,. \end{aligned}$$

The computational method to determine a minimal order descriptor system realization or a reduced order realization without non-dynamic modes is implemented in the MEX-function `sl_gminr`, which is called, with appropriately set options, by the function `gminreal`.

Application example

The function `gminreal` displays messages indicating the number of eigenvalues (uncontrollable, unobservable, non-dynamic) removed from the model. This verbose output can be disabled as shown in the example below:

```
warning off
sysmin = gminreal(sys);
warning on
```

3.4.3 gbalmr

Syntax

```
[SYSR,HS] = gbalmr(SYS)
[SYSR,HS] = gbalmr(SYS,TOL)
[SYSR,HS] = gbalmr(SYS,TOL,BALANCE)
```

Description

`gbalmr` performs model reduction of a stable LTI state-space system using balancing-related methods.

Input data

`SYS` is a stable LTI system, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t)+Bu(t), \\ y(t) &= Cx(t)+Du(t). \end{aligned} \tag{118}$$

`TOL` is a relative tolerance used to determine the order of the reduced model. If `TOL` is not specified as input, or if `TOL` is empty, or if `TOL` = 0, then the value `TOL = sqrt(eps)` is internally used.

`BALANCE` is a character option variable to be set to `'balance'` to compute a balanced realization of the reduced order model. By default, the state-space realization of the computed reduced order model is not balanced.

Output data

`SYSR` contains the minimal realization of the resulting reduced order system in a descriptor system state-space form

$$\begin{aligned} E_r\lambda x_r(t) &= A_r x_r(t) + B_r u(t), \\ y_r(t) &= C_r x_r(t) + D_r u(t). \end{aligned} \tag{119}$$

The realization of `SYSR` is balanced (i.e., $E_r = I$ and the controllability and observability Gramians are equal and diagonal) if the balancing option `BALANCE = 'balance'` has been specified. The order of `SYSR` is the number of Hankel singular values of `SYS`, which are greater than `TOL` $\|G(\lambda)\|_H$, where $G(\lambda)$ is the TFM of the system (118).

`HS` is a column vector which contains the decreasingly ordered Hankel singular values of `SYS`.

Method

For the order reduction of a standard system (i.e., with $E = I$), the balancing-free method of [45] or the balancing-based method of [37] are used. For a descriptor system the balancing related order reduction methods of [35] are used. For the solution of the intervening Lyapunov and Stein equation for the Cholesky factors of the solution, the mex-function `sl_glme` is used.

3.4.4 gss2ss

Syntax

```
[SYSR,RANKE] = gss2ss(SYS)
[SYSR,RANKE] = gss2ss(SYS,TOL)
[SYSR,RANKE] = gss2ss(SYS,TOL,ESHAPE)
```

Description

`gss2ss` performs the conversion of a LTI descriptor state-space systems $(A - \lambda E, B, C, D)$ to a SVD-like coordinate form $(A_r - \lambda E_r, B_r, C_r, D_r)$ without non-dynamic modes, such that the corresponding transfer function matrices are equal.

Input data

`SYS` is a LTI system, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{120}$$

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` = 0, an internally computed default value is used.

`ESHAPE` is a character option variable to specify the shape of the leading invertible diagonal block E_{11} of the resulting descriptor matrix $E_r = \mathrm{diag}(E_{11}, 0)$ (see **Method**). The following options can be used for `ESHAPE`:

- `'diag'` – diagonal (the nonzero diagonal elements are the decreasingly ordered nonzero singular values of E);
- `'triu'` – upper triangular;
- `'ident'` – identity (default).

Output data

`SYSR` contains the resulting reduced order system without non-dynamic modes, in a descriptor system state-space form

$$\begin{aligned} E_r \lambda x_r(t) &= A_r x_r(t) + B_r u(t), \\ y_r(t) &= C_r x_r(t) + D_r u(t), \end{aligned} \tag{121}$$

where E_r has a block-diagonal form $E_r = \mathrm{diag}(E_{11}, 0)$, with E_{11} invertible. The resulting shape of E_{11} is in accordance with the specified option by `ESHAPE`. `SYSR` has the same TFM as `SYS` and the resulting order of `SYSR` is the order of `SYS` minus the number of simple infinite (non-dynamic) eigenvalues of the pole pencil $A - \lambda E$.

`RANKE` is the rank of E (and also the order of E_{11}).

Method

To remove the non-dynamic eigenvalues of the descriptor system $(A - \lambda E, B, C, D)$, the system matrices are first reduced using non-orthogonal transformation matrices Q and Z to a SVD-like coordinate form

$$(\widetilde{A} - \lambda \widetilde{E}, \widetilde{B}, \widetilde{C}, D) := (QAZ - \lambda QEZ, QB, CZ, D),$$

where

$$\begin{aligned} \widetilde{A} - \lambda \widetilde{E} &= \begin{bmatrix} A_{11} - \lambda E_{11} & A_{12} & A_{13} \\ A_{21} & I & 0 \\ A_{31} & 0 & 0 \end{bmatrix}, \quad \widetilde{B} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}, \\ \widetilde{C} &= \begin{bmatrix} C_1 & C_2 & C_3 \end{bmatrix}, \end{aligned}$$

with E_{11} invertible and either upper triangular (if `ESHAPE = 'triu'`) or diagonal (if `ESHAPE = 'diag'` or `ESHAPE = 'identity'`). To compute the above SVD-like form, the MEX-function `sl_gstra` is called by the function `gss2ss`, with appropriately set options.

If `ESHAPE = 'triu'` or `ESHAPE = 'diag'`, the reduced descriptor system without non-

dynamic modes $(A_r - \lambda E_r, B_r, C_r, D_r)$ is computed as

$$\begin{aligned} A_r - \lambda E_r &= \left[\begin{array}{cc} A_{11} - A_{12}A_{21} - \lambda E_{11} & A_{13} \\ A_{31} & 0 \end{array} \right], \\ B_r &= \left[\begin{array}{c} B_1 - A_{12}B_2 \\ B_3 \end{array} \right], \\ C_r &= \left[\begin{array}{cc} C_1 - C_2A_{21} & C_3 \end{array} \right], \\ D_r &= D - C_2B_2\,. \end{aligned} \tag{122}$$

If `ESHAPE = 'identity'`, then the above matrices are computed as

$$\begin{aligned} A_r - \lambda E_r &= \left[\begin{array}{cc} E_{11}^{-1/2}(A_{11} - A_{12}A_{21})E_{11}^{-1/2} - \lambda I & E_{11}^{-1/2}A_{13} \\ A_{31}E_{11}^{-1/2} & 0 \end{array} \right], \\ B_r &= \left[\begin{array}{c} E_{11}^{-1/2}(B_1 - A_{12}B_2) \\ B_3 \end{array} \right], \\ C_r &= \left[\begin{array}{cc} (C_1 - C_2A_{21})E_{11}^{-1/2} & C_3 \end{array} \right], \\ D_r &= D - C_2B_2\,. \end{aligned}$$

The particular case of an invertible E and with the pair (A, E) in a generalized Hessenberg form (i.e., with A in upper Hessenberg form and E upper triangular) is handled separately, in order to preserve the Hessenberg form of A. In this case, with the additional assumption that all diagonal elements of E are positive (this can be easily arranged by changing the signs of the corresponding rows of E, A and B), the matrices of the standard state-space realization $(A_r - \lambda I, B_r, C_r, D)$ are computed with

$$A_r = E^{-1/2}AE^{-1/2}, \quad B_r = E^{-1/2}B, \quad C_r = CE^{-1/2}\,, \tag{123}$$

where $E^{-1/2}$ is the square root of E computed using the method described in [19, Algorithm 6.7] (implemented in the MATLAB function `sqrtm`).

3.5 Functions for Operations on Generalized LTI Systems

These functions cover the computation of rational nullspace and range space bases, the solution of linear rational equations, the computation of additive spectral decompositions and order reductions using minimal dynamic cover based techniques.

3.5.1 `grnull`

Syntax

```
[SYSRNULL,INFO] = grnull(SYS,OPTIONS)
```

Description

`grnull` computes a proper rational basis $N_r(\lambda)$ of the right nullspace of the transfer function matrix $G_1(\lambda)$ of a LTI descriptor system, such that

$$G_1(\lambda)N_r(\lambda) = 0$$

and determines $G_2(\lambda)N_r(\lambda)$, where $G_2(\lambda)$ is a TFM having the same number of columns as $G_1(\lambda)$.

Input data

`SYS` is an output concatenated compound LTI system, `SYS = [ SYS1; SYS2 ]`, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y_1(t) &= C_1x(t) + D_1u(t), \\ y_2(t) &= C_2x(t) + D_2u(t), \end{aligned} \tag{124}$$

where `SYS1` has the transfer function matrix $G_1(\lambda)$ with the descriptor system realization $(A - \lambda E, B, C_1, D_1)$, `SYS2` has the transfer function matrix $G_2(\lambda)$ with the descriptor system realization $(A - \lambda E, B, C_2, D_2)$, and $y_1(t) \in \mathbb{R}^{p_1}$ and $y_2(t) \in \mathbb{R}^{p_2}$ are the outputs of `SYS1` and `SYS2`, respectively.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	**Description**
`tol`	relative tolerance for rank computations (Default: internally computed)
`p2`	p_2, the number of outputs of `SYS2` (Default: $p_2 = 0$)
`simple`	option to compute a simple proper basis: `true` – compute a simple basis; the orders of the basis vectors are provided in `INFO.degs`; `false` – no simple basis computed (default)
`inner`	option to compute an inner basis: `true` – compute an inner basis; if the option for simple basis has been selected then each basis vector results inner and the orders of the basis vectors are provided in `INFO.deg` (the resulting basis may not be inner) `false` – no inner basis is computed (default)
`offset`	stability boundary offset β, to be used to assess the finite zeros which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite zeros having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite zeros having moduli in the interval $[1 - \beta, 1 + \beta]$ (Default: $\beta = 1.4901 \cdot 10^{-08}$).
`tcond`	maximum allowed value for the condition numbers of the employed non-orthogonal transformation matrices (Default: 10^4) (only used if `OPTIONS.simple = true`)
`sdeg`	prescribed stability degree for the resulting right nullspace basis (Default: `[ ]`)
`poles`	a complex conjugated set of desired poles to be assigned for the resulting right nullspace basis (Default: `[ ]`)

Output data

`SYSRNULL` contains the output concatenated compound system `[ NR; SYS2*NR ]`, in a descriptor system state-space form

$$\begin{aligned} \widetilde{E}_r \lambda x_r(t) &= \widetilde{A}_r x_r(t) + \widetilde{B}_r v(t), \\ y_{r,1}(t) &= \widetilde{C}_{r,1} x_r(t) + \widetilde{D}_{r,1} v(t), \\ y_{r,2}(t) &= \widetilde{C}_{r,2} x_r(t) + \widetilde{D}_{r,2} v(t), \end{aligned} \tag{125}$$

where `NR` is the descriptor realization $(\widetilde{A}_r - \lambda\widetilde{E}_r, \widetilde{B}_r, \widetilde{C}_{r,1}, \widetilde{D}_{r,1})$ of the right nullspace basis $N_r(\lambda)$ of the transfer function matrix $G_1(\lambda)$, and `SYS2*NR` (the series coupling of `SYS2` and `NR`) is the descriptor system realization $(\widetilde{A}_r - \lambda\widetilde{E}_r, \widetilde{B}_r, \widetilde{C}_{r,2}, \widetilde{D}_{r,2})$ of the transfer function matrix $G_2(\lambda)N_r(\lambda)$.

`INFO` is a MATLAB structure containing additional information, as follows:

`INFO` fields	Description
`nrank`	normal rank of the transfer function matrix of `SYS1`;
`stdim`	dimensions of the diagonal blocks of $\widetilde{A}_r - \lambda\widetilde{E}_r$: if `OPTIONS.simple = false`, these are the row dimensions of the full row rank subdiagonal blocks of the pencil $[\,\widetilde{B}_r\ \widetilde{A}_r - \lambda\widetilde{E}_r\,]$ in controllability staircase form; if `OPTIONS.simple = true`, these are the orders of the state-space realizations of the proper rational vectors of the computed simple proper rational right nullspace basis of `SYS1`;
`degs`	increasingly ordered degrees of the vectors of a polynomial right nullspace basis of the transfer function matrix $G_1(\lambda)$ of `SYS1`, representing the right Kronecker indices of $G_1(\lambda)$; also the orders of the realizations of the proper rational vectors of a simple proper rational right nullspace basis. If `OPTIONS.simple = true`, `INFO.deg`(i) is the dimension of the i-th diagonal blocks of $\widetilde{A}_r$ and $\widetilde{E}_r$.
`tcond`	maximum of the condition numbers of the employed non-orthogonal transformation matrices; a warning is issued if `INFO.tcond` $\geq$ `OPTIONS.tcond`.
`fnorm`	the norm of the employed state-feedback used for stabilization; is zero if both `OPTION.sdeg` and `OPTIONS.pole` are empty.

Method

For the definitions related to minimal nullspace bases see Section 2.4. In what follows, we sketch the approach to compute minimal proper rational right nullspace bases proposed in [51] (see also [59, Section 10.3.2] for more details). This approach is employed if `OPTIONS.simple = false`.

Let $G_1(\lambda)$ be the $p_1 \times m$ TFM of `SYS1` and let $G_2(\lambda)$ be the $p_2 \times m$ TFM of `SYS2`. Assume $r = \operatorname{rank} G_1(\lambda)$ is the normal rank of $G_1(\lambda)$. Let $N_r(\lambda)$ be a $m \times (m-r)$ rational left nullspace

basis of $G_1(\lambda)$ satisfying

$$G_1(\lambda)N_r(\lambda) = 0.$$

Such a basis can be computed as

$$N_2(\lambda) = [\,0 \;\; I_m\,]Y_r(\lambda), \tag{126}$$

where $Y_r(\lambda)$ is a rational basis of the right nullspace of the system matrix pencil

$$S(\lambda) = \begin{bmatrix} A - \lambda E & B \\ C_1 & D_1 \end{bmatrix}. \tag{127}$$

The right nullspace $Y_r(\lambda)$ is determined using the Kronecker-like staircase form of $S(\lambda)$ computed as

$$Q^T S(\lambda) Z = \begin{bmatrix} B_r & A_r - \lambda E_r & * \\ 0 & 0 & A_l - \lambda E_l \end{bmatrix}, \tag{128}$$

where Q and Z are orthogonal transformation matrices, the subpencil $A_l - \lambda E_l$ contains the right Kronecker structure and the regular part of $S(\lambda)$, and the subpencil $[\,B_r \;\; A_r - \lambda E_r\,]$ contains the right Kronecker structure of $S(\lambda)$. $[\,B_r \;\; A_r - \lambda E_r\,]$ is obtained in an controllability staircase form with $[\,B_r \;\; A_r\,]$ as in (22) and E_r upper triangular and nonsingular, as in (23). $Y_r(\lambda)$ results as

$$Y_r(\lambda) = Z \begin{bmatrix} I \\ (\lambda E_r - A_r - B_r F)^{-1} B_r \\ 0 \end{bmatrix},$$

where F is a stabilizing state feedback ($F = 0$ if both `OPTION.sdeg` and `OPTIONS.pole` are empty). The rational basis $N_r(\lambda)$ results using (126) with the controllable state-space realization

$$(\widetilde{A}_r - \lambda\widetilde{E}_r, \widetilde{B}_r, \widetilde{C}_{r,1}, \widetilde{D}_{r,1}) := (A_r + B_r F - \lambda E_r, B_r, C_{r,1} + D_{r,1}F, D_{r,1}),$$

where

$$[\,*\;\; C_{r,1} \;\; D_{r,1}\,] := [\,0 \;\; I_m\,]Z.$$

The resulting basis is column proper, that is, $D_{r,1}$ has full column rank. The descriptor realization of $G_2(\lambda)N_r(\lambda)$ is obtained as

$$(\widetilde{A}_r - \lambda\widetilde{E}_r, \widetilde{B}_r, \widetilde{C}_{r,2}, \widetilde{D}_{r,2}) := (A_r + B_r F - \lambda E_r, B_r, C_{r,2} + D_{r,2}F, D_{r,2}),$$

where

$$[\,*\;\; C_{r,2} \;\; D_{r,2}\,] := [\,C_2 \;\; D_2\,]Z.$$

If `OPTIONS.inner = true`, an inner basis is determined as the inner factor $N_{ri}(\lambda)$ of the QR-like factorization $N_r(\lambda) = N_{ri}(\lambda)N_{ro}(\lambda)$, with $N_{ro}(\lambda)$ invertible and with all its zeros stable. $N_{ro}(\lambda)$ has the realization $(A_r + B_r F - \lambda E_r, B_r, -HF, H)$, where F is the stabilizing state feedback computed by solving an appropriate control Riccati equation and H is a suitable invertible feedthrough matrix (see Theorem 5 and Theorem 6 for details). Accordingly, the realizations for the inner basis $N_{ri}(\lambda)$ and $G_2(\lambda)N_{ri}(\lambda)$ can be explicitly computed as

$$N_{ri}(\lambda) = \left[\begin{array}{c|c} A_r + B_r F - \lambda E_r & B_r H^{-1} \\ \hline C_{r,1} + D_{r,1}F & D_{r,1}H^{-1} \end{array}\right], \quad G_2(\lambda)N_{ri}(\lambda) = \left[\begin{array}{c|c} A_r + B_r F - \lambda E_r & B_r H^{-1} \\ \hline C_{r,2} + D_{r,2}F & D_{r,2}H^{-1} \end{array}\right].$$

Remark 2. The existence of an inner basis $N_{ri}(\lambda)$ requires that $N_r(\lambda)$ has no zeros on the boundary of the stability region. This condition is ensured, for example, if $N_r(\lambda)$ is minimal, which is guaranteed if the realization $(A-\lambda E, B, C_1, D_1)$ of `SYS1` is minimal. More generally, `SYS2` must not have poles on the boundary of the stability domain, which are uncontrollable eigenvalues of the realization of `SYS1`. In this case, there is no inner basis $N_{ri}(\lambda)$ such that $G_2(\lambda)N_{ri}(\lambda)$ is stable too. Therefore, the condition on zeros is only checked if `SYS2` is provided, and, if not fulfilled, the resulting $N_r(\lambda)$ and $G_2(\lambda)N_r(\lambda)$ are returned. □

For the computation of the Kronecker-like form (128), the mex-function `sl_klf`, based on the algorithm proposed in [2], is called by the function `grnull`. `INFO.stdim` contains the dimensions ν_j, $j = 1, \ldots, k$ of the diagonal blocks in the staircase form (22). `INFO.degs` contains the degrees of a minimal polynomial left nullspace basis. These are the right Kronecker indices of the system matrix pencil $S(\lambda)$ in (127) and are determined as follows: there are $\nu_{i-1} - \nu_i$ vectors of degree $i-1$, for $i = 1, \ldots, k$, where $\nu_0 := m - r$.

If `OPTIONS.simple = true`, a simple proper left nullspace basis is computed, using the method of [55] to determine a simple basis from a proper basis as computed above. The employed dynamic covers based algorithm relies on performing non-orthogonal similarity transformations. The estimated maximum condition number used in these computations is provided in `INFO.tcond`. For a simple proper basis, $\widetilde{A}_r$, $\widetilde{E}_r$ and $\widetilde{B}_r$ are block diagonal

$$\widetilde{A}_r - \lambda\widetilde{E}_r = \mathrm{diag}(A_r^1 - \lambda E_r^1, \cdots, A_r^k - \lambda E_r^k), \quad \widetilde{B}_r = \mathrm{diag}(B_r^1, \cdots, B_r^k),$$

with $\widetilde{E}_r$ upper triangular. The state-space realization of the i-th basis (column) vector $v_i(\lambda)$ can be explicitly constructed as $(A_r^i - \lambda E_r^i, B_r^i, \widetilde{C}_{r,1}, D_{r,1}^i)$, where $D_{r,1}^i$ is the i-th column of $D_{r,1}$. `INFO.stdim` contains the dimensions of the diagonal blocks $A_r^i - \lambda E_r^i$, $i = 1, \ldots, k$ and are equal to `INFO.degs`. The corresponding realization for $G_2(\lambda)v_i(\lambda)$ is constructed as $(A_r^i - \lambda E_r^i, B_r^i, \widetilde{C}_{r,2}, D_{r,2}^i)$, where $D_{r,2}^i$ is the i-th column of $D_{r,2}$. If `OPTIONS.inner = true`, then each basis vector is determined inner, by applying the above approach separately to each basis vector.

The resulting realization of `SYSRNULL` is minimal provided the realization of `SYS` is minimal. However, `NR` is a minimal proper basis only if the realization $(A - \lambda E, B, C_1, D_1)$ of `SYS1` is minimal. In this case, `INFO.degs` are the degrees of the vectors of a minimal polynomial basis or, if `OPTIONS.simple = true`, of the resulting minimal simple proper basis.

Example

Example 2. Consider the transfer function matrix used in [22, p. 459]

$$G(s) = \begin{bmatrix} \frac{1}{s} & 0 & \frac{1}{s} & s \\ 0 & (s+1)^2 & (s+1)^2 & 0 \\ -1 & (s+1)^2 & s^2+2\,s & -s^2 \end{bmatrix}, \tag{129}$$

which has normal rank $r = 2$ and the right Kronecker indices $\nu_1 = 0$ and $\nu_2 = 2$. A simple proper minimal right nullspace basis $N_r(s)$ with the poles assigned in -1 has been computed

with the function `grnull` as

$$N_r(s) = \begin{bmatrix} -0.5547 & \dfrac{0.70165s^2}{(s+1)^2} \\ -0.5547 & \dfrac{-0.43853s^2}{(s+1)^2} \\ 0.5547 & \dfrac{0.43853s^2}{(s+1)^2} \\ 0 & \dfrac{-1.14}{(s+1)^2} \end{bmatrix}$$

and has two basis vectors of McMillan degrees 0 and 2. A polynomial basis results simply by taking the numerator polynomial vectors

$$\widetilde{N}_r(s) = \begin{bmatrix} -0.5547 & 0.70165s^2 \\ -0.5547 & -0.43853s^2 \\ 0.5547 & 0.43853s^2 \\ 0 & -1.14 \end{bmatrix}$$

To compute $N_r(s)$ and $\widetilde{N}_r(s)$, the following sequence of commands can be used:

```
% Kailath (1980), page 459: rank 2 matrix
s = tf('s');
G = [1/s 0 1/s s;
    0 (s+1)^2 (s+1)^2 0;
    -1 (s+1)^2 s^2+2*s -s^2];
sys = gir(ss(G),1.e-7);

% set options for simple basis and pole assignment
options = struct('tol',1.e-7,'simple',true,'poles',[-1,-1]);

% compute a simple right nullspace basis Nr(s)
[Nr,info] = grnull(sys,options);
tf(Nr), rki = info.degs  % right Kronecker indices

% check nullspace condition G(s)*Nr(s) = 0
gnrank(sys*Nr,1.e-7,rand)

% minimal polynomial basis computation
Nrtf = tf(Nr);
Nrp = tf(Nrtf.num,1)

% check nullspace condition G(s)*Nrp(s) = 0
gnrank(sys*Nrp,1.e-7,rand)
```

◊

For examples illustrating the computational details of determining simple and polynomial bases, see Example 12 and Example 13, respectively.

3.5.2 `glnull`

Syntax

```
[SYSLNULL,INFO] = glnull(SYS,OPTIONS)
```

Description

`glnull` computes a proper rational basis $N_l(\lambda)$ of the left nullspace of the transfer function matrix $G_1(\lambda)$ of a LTI descriptor system, such that

$$N_l(\lambda)G_1(\lambda) = 0$$

and determines $N_l(\lambda)G_2(\lambda)$, where $G_2(\lambda)$ is a TFM having the same number of rows as $G_1(\lambda)$.

Input data

`SYS` is an input concatenated compound LTI system, `SYS = [ SYS1 SYS2 ]`, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_1u_1(t) + B_2u_2(t), \\ y(t) &= Cx(t) + D_1u_1(t) + D_2u_2(t), \end{aligned} \tag{130}$$

where `SYS1` has the transfer function matrix $G_1(\lambda)$ with the descriptor system realization $(A - \lambda E, B_1, C, D_1)$, `SYS2` has the transfer function matrix $G_2(\lambda)$ with the descriptor system realization $(A - \lambda E, B_2, C, D_2)$, and $u_1(t) \in \mathbb{R}^{m_1}$ and $u_2(t) \in \mathbb{R}^{m_2}$ are the inputs of `SYS1` and `SYS2`, respectively.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	Description
`tol`	relative tolerance for rank computations (Default: internally computed);
`m2`	m_2, the number of inputs of `SYS2` (Default: $m_2 = 0$);
`simple`	option to compute a simple proper basis: `true` – compute a simple basis; the orders of the basis vectors are provided in `INFO.degs`; `false` – no simple basis computed (default);
`coinner`	option to compute a coinner basis: `true` – compute a coinner basis; if the option for simple basis has been selected then each basis vector results coinner and the orders of the basis vectors are provided in `INFO.deg` (the resulting basis may not be coinner) `false` – no coinner basis is computed (default)

`offset`	stability boundary offset β, to be used to assess the finite zeros which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite zeros having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite zeros having moduli in the interval $[1-\beta, 1+\beta]$ (Default: $\beta = 1.4901 \cdot 10^{-08}$).
`tcond`	maximum allowed value for the condition numbers of the employed non-orthogonal transformation matrices (Default: 10^4) (only used if `OPTIONS.simple = true`);
`sdeg`	prescribed stability degree for the resulting left nullspace basis (Default: `[ ]`)
`poles`	a complex conjugated set of desired poles to be assigned for the resulting left nullspace basis (Default: `[ ]`).

Output data

`SYSLNULL` contains the input concatenated compound system `[ NL NL*SYS2 ]`, in a descriptor system state-space form

$$\begin{aligned} \widetilde{E}_l \lambda x_l(t) &= \widetilde{A}_l x_l(t) + \widetilde{B}_{l,1} v_1(t) + \widetilde{B}_{l,2} v_2(t), \\ y_l(t) &= \widetilde{C}_l x_l(t) + \widetilde{D}_{l,1} v_1(t) + \widetilde{D}_{l,2} v_2(t), \end{aligned} \tag{131}$$

where `NL` is the descriptor system realization $(\widetilde{A}_l - \lambda\widetilde{E}_l, \widetilde{B}_{l,1}, \widetilde{C}_l, \widetilde{D}_{l,1})$ of the left nullspace basis $N_l(\lambda)$ of the transfer function matrix $G_1(\lambda)$, and `NL*SYS2` (the series coupling of `NL` and `SYS2`) is the descriptor system realization $(\widetilde{A}_l - \lambda\widetilde{E}_l, \widetilde{B}_{l,2}, \widetilde{C}_l, \widetilde{D}_{l,2})$ of the transfer function matrix $N_l(\lambda)G_2(\lambda)$.

`INFO` is a MATLAB structure containing additional information, as follows:

`INFO` fields	Description
`nrank`	normal rank of the transfer function matrix of `SYS1`;
`stdim`	dimensions of the diagonal blocks of $\widetilde{A}_l - \lambda\widetilde{E}_l$: if `OPTIONS.simple = false`, these are the column dimensions of the full column rank subdiagonal blocks of the pencil $\left[\begin{smallmatrix} \widetilde{A}_l - \lambda\widetilde{E}_l \\ \widetilde{C}_l \end{smallmatrix}\right]$ in observability staircase form; if `OPTIONS.simple = true`, these are the orders of the state-space realizations of the proper rational vectors of the computed simple proper rational left nullspace basis of `SYS1`;
`degs`	increasingly ordered degrees of the vectors of a polynomial left nullspace basis of the transfer function matrix $G_1(\lambda)$ of `SYS1`, representing the left Kronecker indices of $G_1(\lambda)$; also the orders of the realizations of the proper rational vectors of a simple proper rational left nullspace basis. If `OPTIONS.simple = true`, `INFO.deg`(i) is the dimension of the i-th diagonal blocks of $\widetilde{A}_l$ and $\widetilde{E}_l$.

`tcond`	maximum of the condition numbers of the employed non-orthogonal transformation matrices; a warning is issued if `INFO.tcond` $\geq$ `OPTIONS.tcond`.

Method

For the definitions related to minimal nullspace bases see Section 2.4. In what follows, we sketch the approach to compute minimal proper rational left nullspace bases, which is the dual version of the method proposed in [51] (see also [59, Section 10.3.2] for more details). This approach is employed if `OPTIONS.simple = false`.

Let $G_1(\lambda)$ be the $p \times m_1$ TFM of `SYS1` and let $G_2(\lambda)$ be the $p \times m_2$ TFM of `SYS2`. Assume $r = \operatorname{rank} G_1(\lambda)$ is the normal rank of $G_1(\lambda)$. Let $N_l(\lambda)$ be a $(p-r) \times p$ rational left nullspace basis of $G_1(\lambda)$ satisfying

$$N_l(\lambda)G_1(\lambda) = 0.$$

Such a basis can be computed as

$$N_l(\lambda) = Y_l(\lambda) \begin{bmatrix} 0 \\ I_p \end{bmatrix}, \tag{132}$$

where $Y_l(\lambda)$ is a rational basis of the left nullspace of the system matrix pencil

$$S(\lambda) = \begin{bmatrix} A - \lambda E & B_1 \\ C & D_1 \end{bmatrix}. \tag{133}$$

The left nullspace $Y_l(\lambda)$ is determined using the Kronecker-like staircase form of $S(\lambda)$ computed as

$$Q^T S(\lambda) Z = \begin{bmatrix} A_r - \lambda E_r & * \\ 0 & A_l - \lambda E_l \\ 0 & C_l \end{bmatrix}, \tag{134}$$

where Q and Z are orthogonal transformation matrices, the subpencil $A_r - \lambda E_r$ contains the right Kronecker structure and the regular part of $S(\lambda)$, and the subpencil $\left[\begin{smallmatrix} A_l - \lambda E_l \\ C_l \end{smallmatrix}\right]$ contains the left Kronecker structure of $S(\lambda)$. $\left[\begin{smallmatrix} A_l - \lambda E_l \\ C_l \end{smallmatrix}\right]$ is obtained in an observability staircase form with $\left[\begin{smallmatrix} A_l \\ C_l \end{smallmatrix}\right]$ as in (24) and E_l upper triangular and nonsingular, as in (25). $Y_l(\lambda)$ results as

$$Y_l(\lambda) = \left[\begin{array}{c|c|c} 0 & C_l(\lambda E_l - A_l - FC_l)^{-1} & I \end{array}\right] Q^T,$$

where F is a stabilizing output injection matrix. The rational basis $N_l(\lambda)$ results using (132) with the observable state-space realization

$$(\widetilde{A}_l - \lambda \widetilde{E}_l, \widetilde{B}_{l,1}, \widetilde{C}_l, \widetilde{D}_{l,1}) := (A_l + FC_l - \lambda E_l, B_{l,1} + FD_{l,1}, C_l, D_{l,1}),$$

where

$$\begin{bmatrix} * \\ B_{l,1} \\ D_{l,1} \end{bmatrix} := Q^T \begin{bmatrix} 0 \\ I_p \end{bmatrix}.$$

The resulting basis is row proper, that is, $D_{l,1}$ has full row rank. The descriptor realization of $N_l(\lambda)G_2(\lambda)$ is obtained as

$$(\widetilde{A}_l - \lambda\widetilde{E}_l, \widetilde{B}_{l,2}, \widetilde{C}_l, \widetilde{D}_{l,2}) := (A_l + FC_l - \lambda E_l, B_{l,2} + FD_{l,2}, C_l, D_{l,2}),$$

where

$$\begin{bmatrix} * \\ B_{l,2} \\ D_{l,2} \end{bmatrix} := Q^T \begin{bmatrix} B_2 \\ D_2 \end{bmatrix}.$$

If `OPTIONS.coinner = true`, a coinner basis is determined as the coinner factor $N_{li}(\lambda)$ of the RQ-like factorization $N_l(\lambda) = N_{li}(\lambda)N_{lo}(\lambda)$, with $N_{lo}(\lambda)$ invertible and with all its zeros stable. $N_{lo}(\lambda)$ has the realization $(A_l + FC_l - \lambda E_l, -FH, C_l, H)$, where F is the stabilizing output injection computed by solving an appropriate filter Riccati equation and H is a suitable invertible feedthrough matrix (for details, see Theorem 5 and Theorem 6 applied to the dual system). Accordingly, the realizations for the coinner basis $N_{li}(\lambda)$ and $N_{li}(\lambda)G_2(\lambda)$ can be explicitly computed as

$$N_{li}(\lambda) = \left[\begin{array}{c|c} A_l + FC_l - \lambda E_l & B_{l,1} + FD_{l,1} \\ \hline H^{-1}C_l & H^{-1}D_{l,1} \end{array}\right], \quad N_{li}(\lambda)G_2(\lambda) = \left[\begin{array}{c|c} A_l + FC_l - \lambda E_l & B_{l,2} + FD_{l,2} \\ \hline H^{-1}C_l & H^{-1}D_{l,2} \end{array}\right].$$

Remark 3. The existence of a coinner basis $N_{li}(\lambda)$ requires that $N_l(\lambda)$ has no zeros on the boundary of the stability region. This condition is ensured, for example, if $N_l(\lambda)$ is minimal, which is guaranteed if the realization $(A - \lambda E, B_1, C, D_1)$ of `SYS1` is minimal. More generally, `SYS2` must not have poles on the boundary of the stability domain, which are unobservable eigenvalues of the realization of `SYS1`. In this case, there is no coinner basis $N_{li}(\lambda)$ such that $N_{li}(\lambda)G_2(\lambda)$ is stable too. Therefore, the condition on zeros is only checked if `SYS2` is provided, and, if not fulfilled, the resulting $N_l(\lambda)$ and $N_l(\lambda)G_2(\lambda)$ are returned. □

For the computation of the Kronecker-like form (134), the mex-function `sl_klf`, based on the algorithm proposed in [2], is called by the function `glnull`. `INFO.stdim` contains the dimensions μ_j, $j = 1, \ldots, \ell$ of the diagonal blocks in the staircase form (24). `INFO.degs` contains the degrees of a minimal polynomial left nullspace basis. These are the left Kronecker indices of the system matrix pencil $S(\lambda)$ in (133) and are determined as follows: there are $\mu_{i-1} - \mu_i$ vectors of degree $i - 1$, for $i = 1, \ldots, \ell$, where $\mu_0 := p - r$.

If `OPTIONS.simple = true`, a simple proper left nullspace basis is computed, using the method of [55] to determine a simple basis from a proper basis as computed above. The employed dynamic covers based algorithm relies on performing non-orthogonal similarity transformations. The estimated maximum condition number used in these computations is provided in `INFO.tcond`. For a simple proper basis, A_l, E_l and C_l are block diagonal

$$\widetilde{A}_l - \lambda\widetilde{E}_l = \mathrm{diag}(A_l^1 - \lambda E_l^1, \cdots, A_l^\ell - \lambda E_l^\ell), \quad \widetilde{C}_l = \mathrm{diag}(C_l^1, \cdots, C_l^\ell),$$

with E_l upper triangular. The state-space realization of the i-th basis (row) vector $v_i(\lambda)$ can be explicitly constructed as $(A_l^i - \lambda E_l^i, \widetilde{B}_{l,1}, C_l^i, D_{l,1}^i)$, where $D_{l,1}^i$ is the i-th row of $D_{l,1}$. `INFO.stdim` contains the dimensions of the diagonal blocks $A_l^i - \lambda E_l^i$, $i = 1, \ldots, \ell$ and are equal to `INFO.degs`. The corresponding descriptor system realization for $v_i(\lambda)G_2(\lambda)$ is constructed as $(A_l^i - \lambda E_l^i, \widetilde{B}_{l,2}, C_l^i, D_{l,2}^i)$, where $D_{l,2}^i$ is the i-th row of $D_{l,2}$. If `OPTIONS.coinner = true`, then

each basis vector is determined coinner, by applying the above approach separately to each basis vector.

The resulting realization of `SYSLNULL` is minimal provided the realization of `SYS` is minimal. However, `NL` is a minimal proper basis only if the realization $(A - \lambda E, B_1, C, D_1)$ of `SYS1` is minimal. In this case, `INFO.degs` are the degrees of the vectors of a minimal polynomial basis or, if `OPTIONS.simple = true`, of the resulting minimal simple proper basis.

Example

Example 3. Consider the 3×4 transfer function matrix (129) from [22, p. 459] used in Example 2. $G(s)$ has normal rank $r = 2$ and a left Kronecker index $\mu_1 = 1$. A proper minimal left nullspace basis $N_l(s)$ with the poles assigned in -1 has been computed with the function `glnull` as

$$N_l(s) = \begin{bmatrix} -\dfrac{s}{s+1} & \dfrac{1}{s+1} & -\dfrac{1}{s+1} \end{bmatrix}$$

and consists of a single basis vector of McMillan degree 1. To compute $N_l(s)$, the following sequence of commands can be used:

```
% Kailath (1980), page 459: rank 2 matrix
s = tf('s');
G = [1/s 0 1/s s;
    0 (s+1)^2 (s+1)^2 0;
    -1 (s+1)^2 s^2+2*s -s^2];
sys = gir(ss(G),1.e-7);

% set options for simple basis and pole assignment
options = struct('tol',1.e-7,'simple',true,'poles',[-1]);

% compute a simple left nullspace basis Nl(s)
[Nl,info] = glnull(sys,options);
minreal(tf(Nl))

% check the nullspace condition Nl(s)*G(s) = 0
gnrank(Nl*sys,1.e-7,rand)
```

◊

3.5.3 `grange`

Syntax

```
[SYSR,SYSX,INFO] = grange(SYS,OPTIONS)
```

Description

`grange` computes a proper rational basis $R(\lambda)$ of the range space of the transfer function matrix $G(\lambda)$ of a LTI descriptor system, and a full row rank $X(\lambda)$ such that

$$G(\lambda) = R(\lambda)X(\lambda) \tag{135}$$

is a full-rank factorization of $G(\lambda)$.

Input data

`SYS` is a LTI system in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{136}$$

whose transfer function matrix is $G(\lambda)$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	Description
`tol`	relative tolerance for rank computations (Default: internally computed)
`offset`	stability boundary offset β, to be used to assess the finite zeros which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite zeros having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite zeros having moduli in the interval $[1-\beta, 1+\beta]$ (Default: $\beta = 1.4901 \cdot 10^{-08}$).

`zeros`	option for the selection of zeros to be included in the computed range space basis:
	`'none'` – include no zeros (default)
	`'all'` – include all zeros of `SYS`
	`'unstable'` – include all unstable zeros of `SYS`
	`'s-unstable'` – include all strictly unstable zeros of `SYS`, both finite and infinite
	`'stable'` – include all stable zeros of `SYS`
	`'finite'` – include all finite zeros of `SYS`
	`'infinite'` – include all infinite zeros of `SYS`

`inner`	option to compute an inner basis: `true` – compute an inner basis (only if `OPTIONS.zeros = 'none'` or `OPTIONS.zeros = 'unstable'`); `false` – no inner basis is computed (default)
`balance`	balancing option for the Riccati equation solvers (see functions `care` and `dare` of the Control System Toolbox): `true` – perform balancing (default); `false` – disable balancing.

Output data

`SYSR` contains the descriptor system state-space realization of the full column rank proper range basis matrix $R(\lambda)$ in the form

$$\begin{aligned} E_R\lambda x_R(t) &= A_R x_R(t) + B_R v(t), \\ y_R(t) &= C_R x_R(t) + D_R v(t), \end{aligned} \tag{137}$$

where E_R is invertible. The resulting $R(\lambda)$ contains the selected zeros of $G(\lambda)$ via the option parameter `OPTIONS.zeros`. $R(\lambda)$ is inner if `OPTIONS.inner = true` was selected.

The dimension r of the input vector $v(t)$ is the normal rank of $G(\lambda)$.

`SYSX` contains the descriptor system state-space realization of the full row rank transfer function matrix $X(\lambda)$ in the form

$$\begin{aligned} E\lambda\widetilde{x}(t) &= A\widetilde{x}(t) + Bu(t), \\ \widetilde{y}(t) &= \widetilde{C}\widetilde{x}(t) + \widetilde{D}u(t), \end{aligned} \tag{138}$$

where the dimension r of the output vector $\widetilde{y}(t)$ is the normal rank of $G(\lambda)$.

`INFO` is a MATLAB structure containing additional information, as follows:

`INFO` fields	Description
`nrank`	normal rank of the transfer function matrix $G(\lambda)$;
`nfuz`	number of finite unstable zeros of `SYS` lying on the boundary of the stability region $\partial\mathbb{C}_s$ within the offset specified by `OPTION.offset`;
`niuz`	number of infinite zeros of `SYS` in the continuous-time case and 0 in the discrete-time case
`ricrez`	diagnosis flag, as provided provided by the generalized Riccati equation solvers `care` and `dare`; if non-negative, this value represents the Frobenius norm of relative residual of the Riccati equation, while a negative value indicates failure of solving the Riccati equation.

Method

Consider a disjunct partition of the complex plane $\mathbb{C}$ as

$$\mathbb{C} = \mathbb{C}_g \cup \mathbb{C}_b, \quad \mathbb{C}_g \cap \mathbb{C}_b = \emptyset\,, \tag{139}$$

where $\mathbb{C}_g$ and $\mathbb{C}_b$ are symmetric with respect to the real axis. $\mathbb{C}_g$ and $\mathbb{C}_b$ are associated with the "good" and "bad" domains of the complex plane $\mathbb{C}$ for the poles and zeros of $G(\lambda)$. Assume $G(\lambda)$ is a $p \times m$ real rational matrix of normal rank r, with a $\mathbb{C}_b$-stabilizable descriptor system realization (136). Then, there exist two orthogonal matrices U and Z such that

$$\left[\begin{array}{cc} U & 0 \\ 0 & I \end{array}\right] \left[\begin{array}{c|c} A-\lambda E & B \\ \hline C & D \end{array}\right] Z = \left[\begin{array}{cccc} A_{rg}-\lambda E_{rg} & * & * & * \\ 0 & A_{b\ell}-\lambda E_{b\ell} & B_{b\ell} & * \\ 0 & 0 & 0 & B_n \\ \hline 0 & C_{b\ell} & D_{b\ell} & * \end{array}\right], \tag{140}$$

where

(a) The pencil $A_{rg} - \lambda E_{rg}$ has full row rank for $\lambda \in \mathbb{C}_g$ and E_{rg} has full row rank.

(b) $E_{b\ell}$ and B_n are invertible, the pencil

$$\left[\begin{array}{cc} A_{b\ell}-\lambda E_{b\ell} & B_{b\ell} \\ C_{b\ell} & D_{b\ell} \end{array}\right] \tag{141}$$

has full column rank $n_{b\ell} + r$ for $\lambda \in \mathbb{C}_g$ and the pair $(A_{b\ell} - \lambda E_{b\ell}, B_{b\ell})$ is $\mathbb{C}_b$-stabilizable.

The range matrix of $G(\lambda)$, which includes the zeros of $G(\lambda)$ in $\mathbb{C}_b$, has the proper descriptor system realization

$$R(\lambda) = \left[\begin{array}{c|c} A_{b\ell} - \lambda E_{b\ell} & B_{b\ell} \\ \hline C_{b\ell} & D_{b\ell} \end{array}\right]. \tag{142}$$

If `OPTIONS.inner = true` was selected, the inner range matrix is determined in the form

$$R(\lambda) = \left[\begin{array}{c|c} A_{b\ell} + B_{b\ell}F - \lambda E_{b\ell} & B_{b\ell}W \\ \hline C_{b\ell} + D_{b\ell}F & D_{b\ell}W \end{array}\right], \tag{143}$$

where W is a suitable invertible matrix and F is a stabilizing state-feedback matrix. $X(\lambda)$ is determined with a descriptor realization of the form

$$X(\lambda) = \left[\begin{array}{c|c} A - \lambda E & B \\ \hline \widetilde{C} & \widetilde{D} \end{array}\right], \tag{144}$$

where $[\,\widetilde{C}\ \widetilde{D}\,] = W^{-1}[0 \ \ -F \ \ I_r \ \ 0]Z^T$.

The overall factorization approach is described in [57]. The reduction of the system matrix pencil to the special Kronecker-like form (140) is described in [29] and involves the use of the mex-function `sl_klf` to compute the appropriate Kronecker-like form. For the computation of an inner basis, extensions of the standard inner-outer factorization methods of [69] are used. These methods involve the solution of appropriate (continuous- or discrete-time) generalized algebraic Riccati equations. For additional details, see [32] for continuous-time systems and [29] for discrete-time systems.

Example

Example 4. This is **Example 1** from [32] of the transfer function matrix of a continuous-time proper system:

$$G(s) = \left[\begin{array}{ccc} \dfrac{s-1}{s+2} & \dfrac{s}{s+2} & \dfrac{1}{s+2} \\ 0 & \dfrac{s-2}{(s+1)^2} & \dfrac{s-2}{(s+1)^2} \\ \dfrac{s-1}{s+2} & \dfrac{s^2+2\,s-2}{(s+1)\ (s+2)} & \dfrac{2\,s-1}{(s+1)\ (s+2)} \end{array}\right]. \tag{145}$$

$G(s)$ has zeros at $\{1, 2, \infty\}$, poles at $\{-1, -1, -2, -2\}$, and normal rank $r = 2$.

A minimum proper basis of $\mathcal{R}(G(s))$, computed with `grange`, is

$$R(s) = \frac{1}{s + 1.374}\left[\begin{array}{cc} 1.552s + 2.124 & 1.314s + 1.817 \\ 0.593s + 1.186 & -0.758s - 1.516 \\ 2.145s + 2.717 & 0.5558s + 1.059 \end{array}\right],$$

has McMillan-degree 1 and no zeros. The full row rank factor $X(s)$, satisfying $G(s) = R(s)X(s)$, has McMillan degree 4, and zeros at $\{1, 2, -1.374, \infty\}$. The zero at -1.374 is equal to the pole of $R(s)$.

The full-rank factorization of $G(s)$ has been computed using the following sequence of commands:

```
% Oara and Varga (2000), Example 1
s = tf('s');  % define the complex variable s
% enter G(s) and determine a minimal state-space realization
G = [(s-1)/(s+2) s/(s+2) 1/(s+2);
    0 (s-2)/(s+1)^2 (s-2)/(s+1)^2;
    (s-1)/(s+2) (s^2+2*s-2)/(s+1)/(s+2) (2*s-1)/(s+1)/(s+2)];
sys = minreal(ss(G));
gpole(sys)    % the system is stable
gzero(sys)    % the system has 2 unstable zeros and an infinite zero
nrank(sys)    % the normal rank of G(s) is 2

% compute the full-rank factorization G(s) = R(s)*X(s)
[sysr,sysx] = grange(sys,struct('tol',1.e-7));

% check the factorization
gnrank(sysr*sysx-sys,1.e-7,rand)   % R(s)*X(s) = G(s)

gzero(sysr)                        % R(s) has no zeros
gzero(sysx)                        % X(s) has all zeros of G(s)
```

◊

3.5.4 gcrange

Syntax

```
[SYSR,SYSX,INFO] = gcrange(SYS,OPTIONS)
```

Description

`grange` computes a proper rational basis $R(\lambda)$ of the coimage space of the transfer function matrix $G(\lambda)$ of a LTI descriptor system, and a full column rank $X(\lambda)$ such that

$$G(\lambda) = X(\lambda)R(\lambda) \tag{146}$$

is a full-rank factorization of $G(\lambda)$.

Input data

`SYS` is a LTI system in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{147}$$

whose transfer function matrix is $G(\lambda)$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

OPTIONS fields	Description
tol	relative tolerance for rank computations (Default: internally computed)
offset	stability boundary offset β, to be used to assess the finite zeros which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite zeros having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite zeros having moduli in the interval $[1-\beta, 1+\beta]$ (Default: $\beta = 1.4901 \cdot 10^{-08}$).
zeros	option for the selection of zeros to be included in the computed coimage space basis: 'none' – include no zeros (default) 'all' – include all zeros of SYS 'unstable' – include all unstable zeros of SYS 's-unstable' – include all strictly unstable zeros of SYS, both finite and infinite 'stable' – include all stable zeros of SYS 'finite' – include all finite zeros of SYS 'infinite' – include all infinite zeros of SYS
coinner	option to compute a coinner basis: true – compute a coinner basis (only if OPTIONS.zeros = 'none' or OPTIONS.zeros = 'unstable'); false – no coinner basis is computed (default)
balance	balancing option for the Riccati equation solvers (see functions care and dare of the Control System Toolbox): true – perform balancing (default); false – disable balancing.

Output data

SYSR contains the descriptor system state-space realization of the full row rank proper coimage basis matrix $R(\lambda)$ in the form

$$\begin{aligned} E_R\lambda x_R(t) &= A_R x_R(t) + B_R u(t), \\ w(t) &= C_R x_R(t) + D_R u(t), \end{aligned} \tag{148}$$

where E_R is invertible. The resulting $R(\lambda)$ contains the selected zeros of $G(\lambda)$ via the option parameter OPTIONS.zeros. $R(\lambda)$ is coinner if OPTIONS.inner = true was selected. The dimension r of the output vector $w(t)$ is the normal rank of $G(\lambda)$.

SYSX contains the descriptor system state-space realization of the full column rank transfer function matrix $X(\lambda)$ in the form

$$\begin{aligned} E\lambda\widetilde{x}(t) &= A\widetilde{x}(t) + \widetilde{B}w(t), \\ \widetilde{y}(t) &= C\widetilde{x}(t) + \widetilde{D}w(t), \end{aligned} \tag{149}$$

where the dimension r of the input vector $w(t)$ is the normal rank of $G(\lambda)$.

INFO is a MATLAB structure containing additional information, as follows:

INFO fields	Description
nrank	normal rank of the transfer function matrix $G(\lambda)$;
nfuz	number of finite unstable zeros of SYS lying on the boundary of the stability region $\partial\mathbb{C}_s$ within the offset specified by OPTION.offset;
niuz	number of infinite zeros of SYS in the continuous-time case and 0 in the discrete-time case
ricrez	diagnosis flag, as provided provided by the generalized Riccati equation solvers care and dare; if non-negative, this value represents the Frobenius norm of relative residual of the Riccati equation, while a negative value indicates failure of solving the Riccati equation.

Method

The coimage computation is performed by applying the range computation method described in [57], to the dual descriptor system realization corresponding to the transposed rational matrix $G^T(\lambda)$ to obtain the full rank factorization

$$G^T(\lambda) = \widetilde{R}(\lambda)\widetilde{X}(\lambda),$$

where $\widetilde{R}(\lambda)$ is full column rank and contains those zeros of $G(\lambda)$ which have been selected via the OPTIONS.zeros field, while $\widetilde{X}(\lambda)$ has full row rank. The coimage basis of $G(\lambda)$ is given by $R(\lambda) = \widetilde{R}^T(\lambda)$ and the corresponding $X(\lambda)$ in (146) is given by $X(\lambda) = \widetilde{X}^T(\lambda)$.

The factorization approach of [57] is based on the reduction of the system matrix pencil to a special Kronecker-like form (see (140)) and is described in [29]. This reduction involves the use of the mex-function sl_klf to compute the appropriate Kronecker-like form. For the computation of an coinner basis, extensions of the standard inner-outer factorization methods of [69] are used. These methods involve the solution of appropriate (continuous- or discrete-time) generalized algebraic Riccati equations. For additional details, see [32] for continuous-time systems and [29] for discrete-time systems.

Example

Example 5. This is **Example 2** from [32], and concerns with the transfer function matrix (145) of a continuous-time proper system, already considered in Example 4. The Moore-Penrose pseudo-inverse $G^\dagger(s)$ of the rational matrix $G(s)$ can be computed in three steps, using a simplified version of the approach described in [32]:

1. Compute a full-rank factorization $G(s) = U(s)G_1(s)$, with $U(s)$, a minimal inner range matrix, and $G_1(s)$ full row rank.

2. Compute the dual full-rank factorization $G_1(s) = G_2(s)V(s)$, with $V(s)$, a minimal coinner coimage (i.e., $V(s)V^\sim(s) = I$), and $G_2(s)$ invertible.

3. Compute
$$G^\dagger(s) = V^\sim(s)G_2^{-1}(s)U^\sim(s).$$

These computational steps are implemented in the following MATLAB code:

```
%  Oara and Varga (2000), Example 2
s = tf('s');  % define the complex variable s
% enter G(s) and determine a minimal state-space realization
Gs = [(s-1)/(s+2) s/(s+2) 1/(s+2);
    0 (s-2)/(s+1)^2 (s-2)/(s+1)^2;
    (s-1)/(s+2) (s^2+2*s-2)/(s+1)/(s+2) (2*s-1)/(s+1)/(s+2)];
G = minreal(ss(Gs));

% use tolerance 1.e-7 for rank determinations
opt = struct('tol',1.e-7,'inner',true,'coinner',true);

% compute the full-rank factorization G(s) = U(s)*G1(s) with U(s) inner
[U,G1] = grange(G,opt);

% compute the full-rank factorization G1(s) = G2(s)*V(s) with V(s) coinner
[V,G2] = gcrange(G1,opt);

% compute the pseudo-inverse Gpinv
Gpinv = gir(V'*(G2\(U')));

% check the four axioms defining the Moore-Penrose pseudo-inverse
gnrank(G*Gpinv*G-G,1.e-7,rand)               % (i)   G*Gpinv*G = G
gnrank(Gpinv*G*Gpinv-Gpinv,1.e-7,rand)       % (ii)  Gpinv*G*Gpinv = Gpinv
gnrank(G*Gpinv-(G*Gpinv)',1.e-7,rand)        % (iii) G*Gpinv = (G*Gpinv)'
gnrank(Gpinv*G-(Gpinv*G)',1.e-7,rand)        % (iv)  Gpinv*G = (Gpinv*G)'
```

$\Diamond$

3.5.5 grsol

Syntax

```
[SYSX,INFO,SYSGEN] = grsol(SYSG,SYSF,OPTIONS)
[SYSX,INFO,SYSGEN] = grsol(SYSGF,MF,OPTIONS)
```

Description

`grsol` computes the solution $X(\lambda)$ of the linear rational matrix equation

$$G(\lambda)X(\lambda) = F(\lambda), \tag{150}$$

where $G(\lambda)$ and $F(\lambda)$ are the (rational) transfer function matrices of LTI descriptor systems.

Input data

For the usage with

```
[SYSX,INFO,SYSGEN] = grsol(SYSG,SYSF,OPTIONS)
```

the input parameters are as follows:

`SYSG` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_G\lambda x_G(t) &= A_G x_G(t) + B_G u(t), \\ y_G(t) &= C_G x_G(t) + D_G u(t), \end{aligned} \tag{151}$$

where $y_G(t) \in \mathbb{R}^p$.

`SYSF` is a LTI system, whose transfer function matrix is $F(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_F\lambda x_F(t) &= A_F x_F(t) + B_F v(t), \\ y_F(t) &= C_F x_F(t) + D_F v(t), \end{aligned} \tag{152}$$

where $y_F(t) \in \mathbb{R}^p$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` **fields**	**Description**
`tol`	relative tolerance for rank computations (Default: internally computed);
`sdeg`	prescribed stability degree for the free poles of the solution $X(\lambda)$ (Default: `[ ]`, i.e., no stabilization performed);
`poles`	a complex conjugated set of desired poles to be assigned for the free poles of the solution $X(\lambda)$ (Default: `[ ]`);
`mindeg`	option to compute a minimum degree solution: `true` – determine a minimum order solution; `false` – determine a particular solution which has possibly non-minimal order (default).

For the usage with

```
[SYSX,INFO,SYSGEN] = grsol(SYSGF,MF,OPTIONS)
```

the input parameters are as follows:

`SYSGF` is an input concatenated compound LTI system, `SYSGF = [ SYSG SYSF ]`, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_G u(t) + B_F v(t), \\ y(t) &= Cx(t) + D_G u(t) + D_F v(t), \end{aligned} \tag{153}$$

where `SYSG` has the transfer function matrix $G(\lambda)$, with the descriptor system realization $(A - \lambda E, B_G, C, D_G)$, and `SYSF` has the transfer function matrix $F(\lambda)$, with the descriptor system realization $(A - \lambda E, B_F, C, D_F)$.

`MF` is the dimension of the input vector $v(t)$ of the system `SYSF`.

`OPTIONS` is a MATLAB structure to specify user options and has the same fields as described previously.

Output data

`SYSX` contains the descriptor system state-space realization of the solution $X(\lambda)$ in the form

$$\begin{aligned} \widetilde{E}\lambda\widetilde{x}(t) &= \widetilde{A}\widetilde{x}(t) + \widetilde{B}v(t), \\ u(t) &= \widetilde{C}\widetilde{x}(t) + \widetilde{D}v(t). \end{aligned} \tag{154}$$

`INFO` is a MATLAB structure containing additional information, as follows:

`INFO` fields	Description
`nrank`	normal rank of the transfer function matrix $G(\lambda)$;
`rdeg`	vector which contains the relative column degrees of $X(\lambda)$ (i.e., the numbers of integrators/delays needed to make each column of $X(\lambda)$ proper);
`tcond`	maximum of the Frobenius-norm condition numbers of the employed non-orthogonal transformation matrices (large values indicate possible loss of numerical stability);
`fnorm`	the Frobenius-norm of the employed state-feedback/feedforward used for dynamic cover computation if `OPTIONS.mindeg = true`, or for stabilization of free poles if `OPTION.sdeg` is not empty (large values indicate possible loss of numerical stability);
`nr`	the order of $A_r - \lambda E_r$, also the row dimension of B_r and also the number of freely assignable poles of the solution (see **Method**);
`nf`	the order of $A_f - \lambda E_f$ (see **Method**);
`ninf`	the order of $A_\infty - \lambda E_\infty$ (see **Method**).

`SYSGEN` contains the input concatenated compound system `[ SYSXO SYSNR ]`, in a descriptor system state-space form

$$\begin{aligned} E_g\lambda x_g(t) &= A_g x_g(t) + B_0 v_1(t) + B_N v_2(t), \\ y_g(t) &= C_g x_g(t) + D_0 v_1(t) + D_N v_2(t), \end{aligned} \tag{155}$$

where the transfer function matrix $X_0(\lambda)$ of `SYSXO`, with the descriptor system realization $(A_g - \lambda E_g, B_0, C_g, D_0)$, is a particular solution satisfying $G(\lambda)X_0(\lambda) = F(\lambda)$, and the transfer function matrix $N_r(\lambda)$ of `SYSNR`, with the descriptor system realization $(A_g - \lambda E_g, B_N, C_g, D_N)$, is a proper right nullspace basis of $G(\lambda)$, satisfying $G(\lambda)N_r(\lambda) = 0$. The transfer function matrices $X_0(\lambda)$ and $N_r(\lambda)$ can be used to generate all solutions of the system (150) as $X(\lambda) = X_0(\lambda) + N_r(\lambda)Y(\lambda)$, where $Y(\lambda)$ is an arbitrary rational matrix with suitable dimensions.

Method

The employed solution approach of the linear rational matrix equation (150) is sketched in Section 2.10. This approach is based on a realization of the form (153) of the input concatenated compound system `SYSGF = [ SYSG SYSF ]`. A detailed computational procedure to solve (150)

is described in [53] (see also [59, Section 10.3.7] for more details). For the intervening computation of the Kronecker-like form of the system matrix pencil of the descriptor system `SYSG`, the mex-function `sl_klf`, based on the algorithm proposed in [2], is employed. If the descriptor realizations of `SYSG` and `SYSF` are separately provided, then an irreducible realization of the input concatenated compound system `[ SYSG SYSF ]` is internally computed. For orthogonal transformation based order reduction purposes, the mex-function `sl_gminr` is employed. Furthermore, the mex-function `sl_gstra` is employed to count controllable infinite poles (needed to determine the relative column degrees) and for reduction to SVD-like form (needed for the elimination of non-dynamic modes)

For the computation of the solution, a so-called *generator* of all solutions is determined in the form (155), where the resulting matrices have the forms

$$A_g - \lambda E_g = \begin{bmatrix} A_r + B_r F - \lambda E_r & * & * \\ 0 & A_f - \lambda E_f & * \\ 0 & 0 & A_\infty - \lambda E_\infty \end{bmatrix}, \quad [\, B_0 \mid B_N \,] = \left[\begin{array}{c|c} B_1 & B_r \\ B_2 & 0 \\ B_3 & 0 \end{array}\right],$$

$$C_g = [\;\; C_r + D_N F \quad * \quad * \;\;],$$

with the descriptor pair $(A_r - \lambda E_r, B_r)$ controllable and E_r invertible, $A_f - \lambda E_f$ regular and E_f invertible (i.e., $A_f - \lambda E_f$ has only finite eigenvalues), and $A_\infty - \lambda E_\infty$ regular and upper triangular with A_∞ invertible and E_∞ nillpotent (i.e. $A_\infty - \lambda E_\infty$ has only infinite eigenvalues), and D_N full row rank. If $G(\lambda)$ is a $p \times m$ TFM of normal rank r, then the matrices B_r and D_N have $m - r$ columns. The pair $(A_r - \lambda E_r, B_r)$ being controllable, the eigenvalues of $A_r + B_r F - \lambda E_r$ can be freely assigned by using a suitably chosen state-feedback matrix F. The descriptor system realization (155) is usually not minimal, being uncontrollable, or having non-dynamic modes, or both. The generator contains the particular solution $X_0(\lambda)$ with the descriptor realization $(A_g - \lambda E_g, B_0, C_g, D_0)$ and a right nullspace basis $N_r(\lambda)$ of $G(\lambda)$ with the (non-minimal) descriptor system realization $(A_g - \lambda E_g, B_N, C_g, D_N)$. A minimal order descriptor system realization of $N_r(\lambda)$ is $(A_r + B_r F - \lambda E_r, B_r, C_r + D_N F, D_N)$. All solutions of the system (150) can be expressed as $X(\lambda) = X_0(\lambda) + N_r(\lambda) Y(\lambda)$, where $Y(\lambda)$ is an arbitrary rational matrix with suitable dimensions.

The resulting generator `SYSGEN` contains the descriptor system realization

$$[\, X_0(\lambda) \; N_r(\lambda) \,] = (A_g - \lambda E_g, [\, B_0 \; B_N \,], C_g, [\, D_0 \; D_N \,]).$$

The orders of the diagonal blocks of $A_g - \lambda E_g$ are provided in the `INFO` structure as follows: `INFO.nr` contains the order of $A_r - \lambda E_r$, `INFO.nf` contains the order of $A_f - \lambda E_f$, and `INFO.ninf` contains the order of $A_\infty - \lambda E_\infty$. `INFO.nrank` contains the normal rank r of $G(\lambda)$. The relative column degrees, provided in `INFO.rdeg`, are the numbers of infinite poles of the successive columns of $X_0(\lambda)$.

If `OPTIONS.mindeg = false`, the computed solution `SYSX` represents a minimal realization of the particular solution $X_0(\lambda)$, computed by eliminating the uncontrollable eigenvalues and non-dynamic modes of the realization $(A_g - \lambda E_g, B_0, C_g, D_0)$, where F is determined such that the (free) eigenvalues of $A_r + B_r F - \lambda E_r$ are moved to the stability domain specified via `OPTIONS.sdeg` or to locations specified in `OPTIONS.poles`. If both `OPTIONS.sdeg` and `OPTIONS.poles` are empty, then $F = 0$ is used.

If `OPTIONS.mindeg = true`, the computed solution `SYSX` represents a minimal realization of $X(\lambda) = X_0(\lambda) + N_r(\lambda) Y(\lambda)$, where $Y(\lambda)$ is determined such that $X(\lambda)$ has the least achievable

McMillan degree. For this computation, order reduction based on computing minimum dynamic covers is employed (see **Procedure GRMCOVER2** in [59, Section 10.4.3]).

Example

Example 6. This example is taken from [14], where an one-sided model matching problem is solved, which involves the computation of a stable and proper solution of $G(s)X(s) = F(s)$ with

$$G(s) = \left[\begin{array}{cc} \dfrac{s-1}{s\,(s+1)} & \dfrac{s-1}{s\,(s+2)} \end{array} \right], \quad F(s) = \left[\begin{array}{cc} \dfrac{s-1}{(s+1)\,(s+3)} & \dfrac{s-1}{(s+1)\,(s+4)} \end{array} \right].$$

Both $G(s)$ and $[\,G(s)\ F(s)\,]$ have rank equal to 1 and zeros $\{1, \infty\}$. It follows, according to Lemma 9, that the linear rational matrix equation $G(s)X(s) = F(s)$ has a stable and proper solution. A fourth order stable and proper solution has been computed in [14]. A least order solution, with McMillan degree equal to 2, has been computed with `grsol` as

$$X(s) = \left[\begin{array}{cc} \dfrac{0.48889(s-1.045)}{s+3} & \dfrac{0.41333(s-1.419)}{s+4} \\ \dfrac{0.51111(s+2)}{s+3} & \dfrac{0.58667(s+2)}{s+4} \end{array} \right].$$

To compute a least order solution $X(s)$, the following sequence of commands can be used:

```
% Gao & Antsaklis (1989)
s = tf('s');
G = [(s-1)/(s*(s+1))  (s-1)/(s*(s+2))];
F = [(s-1)/((s+1)*(s+3))  (s-1)/((s+1)*(s+4))];

% build a minimal realization of [G(s) F(s)]
sysgf = minreal(ss([G F]));

% solve G(s)*X(s) = F(s) for the least order solution
[X,info] = grsol(sysgf,2,struct('mindeg',true)); info
minreal(zpk(X))

% check solution
gnrank(G*X-F)
```

◊

3.5.6 `glsol`

Syntax

```
[SYSX,INFO,SYSGEN] = glsol(SYSG,SYSF,OPTIONS)
[SYSX,INFO,SYSGEN] = glsol(SYSGF,MF,OPTIONS)
```

Description

`glsol` computes the solution $X(\lambda)$ of the linear rational matrix equation

$$X(\lambda)G(\lambda) = F(\lambda), \tag{156}$$

where $G(\lambda)$ and $F(\lambda)$ are the (rational) transfer function matrices of LTI descriptor systems.

Input data

For the usage with

```
[SYSX,INFO,SYSGEN] = glsol(SYSG,SYSF,OPTIONS)
```

the input parameters are as follows:

`SYSG` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_G\lambda x_G(t) &= A_G x_G(t) + B_G u(t), \\ y_G(t) &= C_G x_G(t) + D_G u(t), \end{aligned} \tag{157}$$

where $u(t) \in \mathbb{R}^m$.

`SYSF` is a LTI system, whose transfer function matrix is $F(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_F\lambda x_F(t) &= A_F x_F(t) + B_F v(t), \\ y_F(t) &= C_F x_F(t) + D_F v(t), \end{aligned} \tag{158}$$

where $v(t) \in \mathbb{R}^m$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	**Description**
`tol`	relative tolerance for rank computations (Default: internally computed);
`sdeg`	prescribed stability degree for the free poles of the solution $X(\lambda)$ (Default: `[ ]`, i.e., no stabilization performed);
`poles`	a complex conjugated set of desired poles to be assigned for the free poles of the solution $X(\lambda)$ (Default: `[ ]`);
`mindeg`	option to compute a minimum degree solution: `true` – determine a minimum order solution; `false` – determine a particular solution which has possibly non-minimal order (default).

For the usage with

```
[SYSX,INFO,SYSGEN] = glsol(SYSGF,MF,OPTIONS)
```

the input parameters are as follows:

SYSGF is an output concatenated compound LTI system, SYSGF = [SYSG; SYSF], in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y_G(t) &= C_G x(t) + D_G u(t), \\ y_F(t) &= C_F x(t) + D_F u(t), \end{aligned} \tag{159}$$

where SYSG has the transfer function matrix $G(\lambda)$, with the descriptor system realization $(A - \lambda E, B, C_G, D_G)$, and SYSF has the transfer function matrix $F(\lambda)$, with the descriptor system realization $(A - \lambda E, B, CF, D_F)$.

MF is the dimension of the output vector $y_F(t)$ of the system SYSF.

OPTIONS is a MATLAB structure to specify user options and has the same fields as described previously.

Output data

SYSX contains the descriptor system state-space realization of the solution $X(\lambda)$ in the form

$$\begin{aligned} \widetilde{E}\lambda\widetilde{x}(t) &= \widetilde{A}\widetilde{x}(t) + \widetilde{B}y_G(t), \\ y(t) &= \widetilde{C}\widetilde{x}(t) + \widetilde{D}y_G(t). \end{aligned} \tag{160}$$

INFO is a MATLAB structure containing additional information, as follows:

INFO fields	Description
nrank	normal rank of the transfer function matrix $G(\lambda)$;
rdeg	vector which contains the relative row degrees of $X(\lambda)$ (i.e., the numbers of integrators/delays needed to make each row of $X(\lambda)$ proper);
tcond	maximum of the Frobenius-norm condition numbers of the employed non-orthogonal transformation matrices (large values indicate possible loss of numerical stability);
fnorm	the Frobenius-norm of the employed state-feedback/feedforward used for dynamic cover computation if OPTIONS.mindeg = true, or for stabilization of free poles if OPTION.sdeg is not empty (large values indicate possible loss of numerical stability);
ninf	the order of $A_\infty - \lambda E_\infty$ (see **Method**);
nf	the order of $A_f - \lambda E_f$ (see **Method**);
nl	the order of $A_l - \lambda E_l$, also the column dimension of C_l, and also the number of freely assignable poles of the solution (see **Method**), if SYSGEN is computed; otherwise, empty.

SYSGEN contains the output concatenated compound system [SYSX0; SYSNL], in a descriptor

system state-space form

$$\begin{aligned} E_g\lambda x_g(t) &= A_g x_g(t) + B_g v(t), \\ y_0(t) &= C_0 x_g(t) + D_0 v(t), \\ y_N(t) &= C_N x_g(t) + D_N v(t), \end{aligned} \tag{161}$$

where the transfer function matrix $X_0(\lambda)$ of `SYSX0`, with the descriptor system realization $(A_g - \lambda E_g, B_g, C_0, D_0)$, is a particular solution satisfying $X_0(\lambda)G(\lambda) = F(\lambda)$, and the transfer function matrix $N_l(\lambda)$ of `SYSNL`, with the descriptor system realization $(A_g - \lambda E_g, B_g, C_N, D_N)$, is a proper left nullspace basis of $G(\lambda)$, satisfying $N_l(\lambda)G(\lambda) = 0$. The transfer function matrices $X_0(\lambda)$ and $N_l(\lambda)$ can be used to generate all solutions of the system (156) as $X(\lambda) = X_0(\lambda) + Y(\lambda)N_l(\lambda)$, where $Y(\lambda)$ is an arbitrary rational matrix with suitable dimensions.

Method

To solve the linear rational equation (156), the function `glsol` calls `grsol` to solve the dual system $G^T(\lambda)\widetilde{X}(\lambda) = F^T(\lambda)$ and obtain the solution as $X(\lambda) = \widetilde{X}^T(\lambda)$. Thus, the employed solution method corresponds to the dual approach sketched in Section 2.10 (see also [53] and [59, Section 10.3.7] for more details). The function `grsol` relies on the mex-functions `sl_klf`, `sl_gminr`, and `sl_gstra`.

For the computation of the solution, a so-called *generator* of all solutions is determined in the form (161), where the resulting matrices have the forms

$$A_g - \lambda E_g = \begin{bmatrix} A_\infty - \lambda E_\infty & * & * \\ 0 & A_f - \lambda E_f & * \\ 0 & 0 & A_l + KC_l - \lambda E_l \end{bmatrix}, \quad B_g = \begin{bmatrix} * \\ * \\ B_l + KD_N \end{bmatrix},$$

$$\begin{bmatrix} C_0 \\ C_N \end{bmatrix} = \begin{bmatrix} C_1 & C_2 & C_3 \\ 0 & 0 & C_l \end{bmatrix},$$

with $A_\infty - \lambda E_\infty$ regular and upper triangular with A_∞ invertible and E_∞ nillpotent (i.e. $A_\infty - \lambda E_\infty$ has only infinite eigenvalues), $A_f - \lambda E_f$ regular and E_f invertible (i.e., $A_f - \lambda E_f$ has only finite eigenvalues), the descriptor pair $(A_l - \lambda E_l, C_l)$ observable and E_l invertible, and D_N full column rank. If $G(\lambda)$ is a $p \times m$ TFM of normal rank r, then the matrices C_l and D_N have $p - r$ rows. The pair $(A_l - \lambda E_l, C_l)$ being observable, the eigenvalues of $A_l + KC_l - \lambda E_l$ can be freely assigned by using a suitably chosen output-injection matrix K.

The descriptor system realization (161) is usually not minimal, being unobservable, or having non-dynamic modes, or both. The generator contains the particular solution $X_0(\lambda)$ with the descriptor realization $(A_g - \lambda E_g, B_g, C_0, D_0)$ and a left nullspace basis $N_l(\lambda)$ of $G(\lambda)$ with the (non-minimal) descriptor system realization $(A_g - \lambda E_g, B_g, C_N, D_N)$. A minimal order descriptor system realization of $N_r(\lambda)$ is $(A_l + KC_l - \lambda E_l, B_l + KD_N, C_l, D_N)$. All solutions of the system (150) can be expressed as $X(\lambda) = X_0(\lambda) + Y(\lambda)N_l(\lambda)$, where $Y(\lambda)$ is an arbitrary rational matrix with suitable dimensions.

The resulting generator `SYSGEN` contains the descriptor system realization

$$\begin{bmatrix} X_0(\lambda) \\ N_l(\lambda) \end{bmatrix} = \left(A_g - \lambda E_g, B_g, \begin{bmatrix} C_0 \\ C_N \end{bmatrix}, \begin{bmatrix} D_0 \\ D_N \end{bmatrix} \right).$$

The orders of the diagonal blocks of $A_g - \lambda E_g$ are provided in the `INFO` structure as follows: `INFO.ninf` contains the order of $A_\infty - \lambda E_\infty$, `INFO.nf` contains the order of $A_f - \lambda E_f$, and `INFO.nl` contains the order of $A_l - \lambda E_l$. `INFO.nrank` contains the normal rank r of $G(\lambda)$. The relative row degrees, provided in `INFO.rdeg`, are the numbers of infinite poles of the successive rows of $X_0(\lambda)$.

If `OPTIONS.mindeg = false`, the computed solution `SYSX` represents a minimal realization of the particular solution $X_0(\lambda)$, computed by eliminating the unobservable eigenvalues and non-dynamic modes of the realization $(A_g - \lambda E_g, B_g, C_0, D_0)$, where K is determined such that the (free) eigenvalues of $A_l + KC_l - \lambda E_l$ are moved to the stability domain specified via `OPTIONS.sdeg` or to locations specified in `OPTIONS.poles`. If both `OPTIONS.sdeg` and `OPTIONS.poles` are empty, then $K = 0$ is used.

If `OPTIONS.mindeg = true`, the computed solution `SYSX` represents a minimal realization of $X(\lambda) = X_0(\lambda) + Y(\lambda)N_l(\lambda)$, where $Y(\lambda)$ is determined such that $X(\lambda)$ has the least achievable McMillan degree. For this computation, order reduction based on computing minimum dynamic covers is employed (see **Procedure GRMCOVER2** in [59, Section 10.4.3]).

Examples

Example 7. This example illustrates the computation of a left inverse of least McMillan degree used in [67]. The transfer function matrices $G(s)$ and $F(s)$ are

$$G(s) = \frac{1}{s^2+3s+2}\begin{bmatrix} s+1 & s+2 \\ s+3 & s^2+2s \\ s^2+3s & 0 \end{bmatrix}, \quad F(s) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The solution $X(s)$ of the rational equation $X(s)G(s) = I$ is a left inverse of $G(s)$. A third order minimal state-space realization of $G(s)$ can be computed using the function `gir` (or `minreal`). To compute a least McMillan order proper left inverse, the following MATLAB commands can be used:

```
% Wang and Davison Example (1973)
s = tf('s');
g = [ s+1 s+2; s+3 s^2+2*s; s^2+3*s 0 ]/(s^2+3*s+2); f = eye(2);
sysg = gir(ss(g)); sysf = ss(f);

% compute a least order solution of X(s)G(s) = I
[sysx,info] = glsol(sysg,sysf,struct('mindeg',true)); info
gpole(sysx)   % the left inverse is unstable

% check solution
gnrank(sysx*sysg-sysf)
```

The computed solution is unstable. In fact, it turns out that all least order left inverses are unstable. ♢

Example 8. This example illustrates the computation of a stable left inverse for the example of [67], used previously. Since both $G(s)$ and $\begin{bmatrix} G(s) \\ I_2 \end{bmatrix}$ have no zeros (in particular no unstable zeros), a stable solution of equation $X(s)G(s) = I$ exists, in accordance with Lemma 9. In Example

7, we computed `INFO.nl` = 3, and, therefore, we can assign three free poles of the inverse to $\{-1, -2, -3\}$ to obtain a stable left inverse. This can be done by calling `glsol` as follows:

```
sysx = glsol(sysg,sysf,struct('poles',[-1 -2 -3]));
gpole(sysx)
```

The resulting minimal realization of left inverse `sysx` has the poles equal to $\{-1, -2, -3\}$. $\Diamond$

3.5.7 `ginv`

Syntax

```
[SYSINV,INFO] = ginv(SYS,OPTIONS)
```

Description

`ginv` computes a generalized inverse $X(\lambda)$ of the (rational) transfer function matrix $G(\lambda)$ of a LTI descriptor system, such that $X(\lambda)$ satisfies at least the first two of the Moore-Penrose conditions

$$\begin{array}{ll} (1) & G(\lambda)X(\lambda)G(\lambda) = G(\lambda), \\ (2) & X(\lambda)G(\lambda)X(\lambda) = X(\lambda), \\ (3) & G(\lambda)X(\lambda) = X^{\sim}(\lambda)G^{\sim}(\lambda), \\ (4) & X(\lambda)G(\lambda) = G^{\sim}(\lambda)X^{\sim}(\lambda), \end{array} \tag{162}$$

where $G^{\sim}(s) = G^T(-s)$ for a continuous-time system and $G^{\sim}(z) = G^T(1/z)$ for a discrete-time system.

Input data

`SYS` is a LTI system, with a $p \times m$ transfer function matrix $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{163}$$

where $y(t) \in \mathbb{R}^p$ and $u(t) \in \mathbb{R}^m$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	Description
`tol`	relative tolerance for rank computations (Default: internally computed);
`sdeg`	prescribed stability degree for the free poles of the generalized inverse $X(\lambda)$ (Default: `[ ]`, i.e., no stabilization performed);
`poles`	a complex conjugated set of desired poles to be assigned for the free poles of the generalized inverse $X(\lambda)$ (Default: `[ ]`);
`mindeg`	option to compute a minimum degree generalized inverse: `true` – determine, when appropriate, a minimum order generalized inverse; `false` – determine a particular generalized inverse which has possibly non-minimal order (default).

`type`	type of the desired generalized inverse with the following allowed values: `'1-2'` – for a (1,2)-generalized inverse which satisfies conditions (1) and (2) in (162) (default); `'1-2-3'` – for a (1,2,3)-generalized inverse which satisfies conditions (1), (2) and (3) in (162); `'1-2-4'` – for a (1,2,4)-generalized inverse which satisfies conditions (1), (2) and (4) in (162); `'1-2-3-4'` – for the Moore-Penrose pseudoinverse, which satisfies all conditions (1)-(4) in (162).
`freq`	complex vector containing the frequency values to be used to estimate the normal rank of $G(\lambda)$ (it is recommended to use frequency values which are distinct from the poles and zeros of the system). (Default: `[ ]`, i.e., a random real frequency value used)

Output data

`SYSINV` contains the descriptor system state-space realization of the $m \times p$ generalized inverse $X(\lambda)$ in the form

$$\begin{aligned} \widetilde{E}\lambda\widetilde{x}(t) &= \widetilde{A}\widetilde{x}(t) + \widetilde{B}v(t), \\ w(t) &= \widetilde{C}\widetilde{x}(t) + \widetilde{D}v(t), \end{aligned} \tag{164}$$

where $w(t) \in \mathbb{R}^m$ and $v(t) \in \mathbb{R}^p$.

`INFO` is a MATLAB structure containing additional information, as follows:

`INFO` fields	Description
`nrank`	normal rank of the transfer function matrix $G(\lambda)$;
`tcond`	maximum of the Frobenius-norm condition numbers of the employed non-orthogonal transformation matrices (large values indicate possible loss of numerical stability);
`fnorm`	the Frobenius-norm of the employed state-feedback/feedforward gains used for dynamic cover computation if `OPTIONS.mindeg = true`, or for stabilization of free poles if `OPTION.sdeg` is not empty (large values indicate possible loss of numerical stability);
`nfp`	number of freely assignable poles of the generalized inverse $X(\lambda)$. This value is set to zero if the option `OPTIONS.mindeg = true` is used.

Method

The computation of the selected generalized inverse is preceded by the computation of the normal rank r of $G(\lambda)$, in terms of its descriptor representation. We use the relation

$$r = \operatorname{rank} S(\lambda) - n,$$

where rank $S(\lambda)$ is the normal rank of the system matrix pencil $S(\lambda)$, defined as

$$S(\lambda) := \begin{bmatrix} A - \lambda E & B \\ C & D \end{bmatrix}, \tag{165}$$

and n is the order of the descriptor state-space realization. If `OPTIONS.freq` is a complex vector with n_f test frequency values, which form the set Ω, then the normal rank of $S(\lambda)$ is evaluated as

$$\operatorname{rank} S(\lambda) = \max_{\lambda_z \in \Omega} \operatorname{rank} S(\lambda_z). \tag{166}$$

For the computation of the (1,2)-inverse, if `OPTIONS.mindeg = false`, then in the general case when $r < \min(p, m)$ the method proposed in [50] is employed. If $r = p$ (i.e., $G(\lambda)$ is full row rank), then the (1,2)-inverse $X(\lambda)$ is called a *right inverse* (denoted as $G^{-R}(\lambda)$) and is determined as a particular solution of the linear rational equation $G(\lambda)X(\lambda) = I_p$ using the method described in [53]. If $r = m$ (i.e., $G(\lambda)$ is full column rank), then the (1,2)-inverse $X(\lambda)$ is called a *left inverse* (denoted as $G^{-L}(\lambda)$) and is determined as a particular solution of the linear rational equation $X(\lambda)G(\lambda) = I_m$ using the dual of the method of [53] (i.e., applied to the transposed matrix $G^T(\lambda)$). Depending on the left and right Kronecker structures of the system matrix pencil (see Section 2.3), a number of so-called *spurious* poles of the resulting generalized (1,2)-inverse can be freely assigned using the information provided in `OPTIONS.sdeg` and `OPTIONS.poles`. If the option `OPTIONS.mindeg = true` is used, then if $r = \min(p, m)$, then least order left or right inverses are determined using the method of [53]. However, in the general case $r < \min(p, m)$, only a reduced order (1,2)-inverse is attempted to be obtained, using an alternative approach based on a full rank factorization $G(\lambda) = U(\lambda)V(\lambda)$, where $U(\lambda)$ is full column rank and $V(\lambda)$ is full row rank. Using this factorization, the (1,2)-inverse is determined as

$$X(\lambda) = V^{-R}(\lambda)U^{-L}(\lambda).$$

where $V^{-R}(\lambda)$ is a least order right inverse of $V(\lambda)$ and $U^{-L}(\lambda)$ is a least order left inverse of $U(\lambda)$. Note that, in general, there is no guarantee that the resulting overall McMillan degree of $X(\lambda)$ is the least possible one (see **Example**).

For the computation of an (1,2,3)-inverse, if `OPTIONS.mindeg = false`, the computational approach involves the determination of a full rank factorization $G(\lambda) = U(\lambda)G_1(\lambda)$, with $U(\lambda)$ inner and $G_1(\lambda)$ full row rank, using the method described in [57]. The (1,2,3)-inverse is computed as

$$X(\lambda) = G_1^{-R}(\lambda)U^{\sim}(\lambda),$$

where $G_1^{-R}(\lambda)$ is any right inverse of $G_1(\lambda)$. A number of *spurious* poles of the right inverse can be freely assigned using the information provided in `OPTIONS.sdeg` and `OPTIONS.poles`. If `OPTIONS.mindeg = true`, a least order right inverse is determined. Simpler computations are performed if $r = m$ (i.e., when $G(\lambda)$ is full column rank), in which case $G_1^{-R}(\lambda) = G_1^{-1}(\lambda)$.

For the computation of an (1,2,4)-inverse), if `OPTIONS.mindeg = false`, the computational approach involves the determination of a full rank factorization $G(\lambda) = G_1(\lambda)U(\lambda)V(\lambda)$, with $V(\lambda)$ co-inner and $G_1(\lambda)$ full column rank, using the dual of method described in [57]. The (1,2,4)-inverse is computed as

$$X(\lambda) = V^{\sim}(\lambda)G_1^{-L}(\lambda),$$

where $G_1^{-L}(\lambda)$ is any left inverse of $G_1(\lambda)$. A number of *spurious* poles of the left inverse can be freely assigned using the information provided in `OPTIONS.sdeg` and `OPTIONS.poles`. A

least order left inverse is determined provided `OPTIONS.mindeg = true`. Simpler computations are performed if $r = p$ (i.e., when $G(\lambda)$ is full row rank), in which case $G_1^{-L}(\lambda) = G_1^{-1}(\lambda)$ is computed.

For the computation of a Moore-Penrose pseudo-inverse (i.e., an (1,2,3,4)-inverse, denoted $G^{\dagger}(\lambda)$), the computational approach involves the determination of a factorization of $G(\lambda)$ of the form

$$G(\lambda) = U(\lambda)\Sigma(\lambda)V(\lambda), \tag{167}$$

where $U(\lambda)$ is inner, $V(\lambda)$ is co-inner and $\Sigma(\lambda)$ is square and invertible. The Moore-Penrose pseudo-inverse results as

$$X(\lambda) = V^{\sim}(\lambda)\Sigma^{-1}(\lambda)U^{\sim}(\lambda).$$

The determination of the factorization (167) can be done in two steps. In the first step, we compute a full-rank factorization $G(\lambda) = U(\lambda)G_1(\lambda)$, with $U(\lambda)$ inner and $G_1(\lambda)$ full row rank, using the method described in [57]. In the second step, the full rank factorization $G_1(\lambda) = \Sigma(\lambda)V(\lambda)$, with $V(\lambda)$ co-inner and $\Sigma(\lambda)$ invertible, is determined using the method [57] applied to $G_1^T(\lambda)$. Simpler computations are performed if $r = \min(p, m)$. For example, if $r = m$, there is no need to perform the second step, while if $r = p$ there is non need to perform the first step.

If `OPTIONS.mindeg = false`, then the number of freely assignable poles is provided in `INFO.nfp`. `INFO.nfp = 0` is returned if `OPTIONS.mindeg = true` or `OPTIONS.type = '1-2-3-4'` are used.

Example

Example 9. This is **Example 1** from [32] of the transfer function matrix of a continuous-time proper system:

$$G(s) = \begin{bmatrix} \dfrac{s-1}{s+2} & \dfrac{s}{s+2} & \dfrac{1}{s+2} \\ 0 & \dfrac{s-2}{(s+1)^2} & \dfrac{s-2}{(s+1)^2} \\ \dfrac{s-1}{s+2} & \dfrac{s^2+2\,s-2}{(s+1)\,(s+2)} & \dfrac{2\,s-1}{(s+1)\,(s+2)} \end{bmatrix}. \tag{168}$$

$G(s)$ has zeros at $\{1, 2, \infty\}$, poles at $\{-1, -1, -2, -2\}$, and normal rank $r = 2$.

A full-order (1,2)-inverse computed with the function `ginv` has McMillan degree 4, four poles, of which three are the zeros of $G(s)$ and an additional pole is at -1.6667, and four zeros, which are the poles of $G(s)$. This (1,2)-inverse of $G(\lambda)$ has been computed using the following sequence of commands:

```
% Oara and Varga (2000), Example 1
s = tf('s');  % define the complex variable s
% enter G(s) and determine a minimal state-space realization
G = [(s-1)/(s+2) s/(s+2) 1/(s+2);
    0 (s-2)/(s+1)^2 (s-2)/(s+1)^2;
    (s-1)/(s+2) (s^2+2*s-2)/(s+1)/(s+2) (2*s-1)/(s+1)/(s+2)];
sys = minreal(ss(G));
```

```
% compute poles and zeros
pol = gpole(sys)
zer = gzero(sys)

% compute a full order (1,2)-inverse
[sysinv,info] = ginv(sys,struct('tol',1.e-7)); info

% check the conditions for an (1,2)-inverse
gnrank(sys*sysinv*sys-sys,1.e-7,rand)
gnrank(sysinv*sys*sysinv-sysinv,1.e-7,rand)

% compute poles and zeros of the inverse
polinv = gpole(sysinv)
zerinv = gzero(sysinv)
```

We can attempt to determine a lower order (1,2)-inverse, using the setting `OPTIONS.mindeg = true`. This leads to an inverse of the same order as before. However, by computing the (1,2)-inverse of $G^T(s)$, the resulting McMillan degree is three, and the poles of the generalized inverse are exactly the zeros of $G(s)$ and therefore, this inverse is guaranteed to be the least order one. The least order (1,2)-inverse can be computed using the following sequence of commands:

```
% compute a least order (1,2)-inverse using the dual system
sysinv = ginv(sys.',struct('tol',1.e-7,'mindeg',true)).';

% check the conditions for an (1,2)-inverse
gnrank(sys*sysinv*sys-sys,1.e-7,rand)
gnrank(sysinv*sys*sysinv-sysinv,1.e-7,rand)

% compute poles and zeros of the inverse
polinv = gpole(sysinv)
zerinv = gzero(sysinv)
```

3.5.8 `gsdec`

Syntax

```
[SYS1,SYS2,Q,Z] = gsdec(SYS,OPTIONS)
```

Description

`gsdec` computes, for the transfer function matrix $G(\lambda)$ of a LTI descriptor system, additive spectral decompositions in the form

$$G(\lambda) = G_1(\lambda) + G_2(\lambda), \tag{169}$$

where $G_1(\lambda)$ has only poles in a certain domain of interest $\mathbb{C}_g \subset \mathbb{C}$ and $G_2(\lambda)$ has only poles in the complementary domain $\mathbb{C} \setminus \mathbb{C}_g$.

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{170}$$

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	**Description**
`tol`	relative tolerance for rank computations (Default: internally computed)
`smarg`	stability margin for the stable poles of the transfer function matrix $G(\lambda)$ of `SYS`, such that, in the continuous-time case, the stable eigenvalues have real parts less than or equal to `OPTIONS.smarg`, and in the discrete-time case, the stable eigenvalues have moduli less than or equal to `OPTIONS.smarg`. (Default: `-sqrt(eps)` for a continuous-time system `SYS`; `1-sqrt(eps)` for a discrete-time system `SYS`.)
`job`	option for specific spectral separation tasks: `'finite'` – $G_1(\lambda)$ has only finite poles and $G_2(\lambda)$ has only infinite poles (default); `'infinite'` – $G_1(\lambda)$ has only infinite poles and $G_2(\lambda)$ has only finite poles; `'stable'` – $G_1(\lambda)$ has only stable poles and $G_2(\lambda)$ has only unstable and infinite poles; `'unstable'` – $G_1(\lambda)$ has only unstable and infinite poles and $G_2(\lambda)$ has only stable poles.

Output data

`SYS1` contains the descriptor system state-space realization of the transfer function matrix $G_1(\lambda)$ in the form

$$\begin{aligned} E_1\lambda x_1(t) &= A_1x_1(t) + B_1u(t), \\ y_1(t) &= C_1x_1(t) + Du(t). \end{aligned} \tag{171}$$

The pair (A_1, E_1) is in a GRSF.

`SYS2` contains the descriptor system state-space realization of the transfer function matrix $G_2(\lambda)$ in the form

$$\begin{aligned} E_2\lambda x_2(t) &= A_2x_2(t) + B_2u(t), \\ y_2(t) &= C_2x_2(t). \end{aligned} \tag{172}$$

The pair (A_2, E_2) is in a GRSF.

`Q` is the employed left transformation matrix used to reduce the pole pencil $A - \lambda E$ to a block-diagonal form (see **Method**).

`Z` is the employed right transformation matrix used to reduce the pole pencil $A - \lambda E$ to a block-diagonal form (see **Method**).

Method

The employed additive decomposition approach of the transfer function matrix $G(\lambda)$ in the form (169) is described in Section 2.7, and is based on the method proposed in [21]. For the computation of spectral separations of the descriptor system `SYS`, the mex-function `sl_gsep` is employed.

For a certain domain of interest $\mathbb{C}_g$, the basic computation consist in determining two invertible matrices Q and Z, to obtain an equivalent descriptor system representation of $G(\lambda)$ with a block-diagonal pole pencil, in the form

$$G(\lambda) = \left[\begin{array}{c|c} QAZ - \lambda QEZ & QB \\ \hline CZ & D \end{array}\right] = \left[\begin{array}{cc|c} A_1 - \lambda E_1 & 0 & B_1 \\ 0 & A_2 - \lambda E_2 & B_2 \\ \hline C_1 & C_2 & D \end{array}\right], \tag{173}$$

where $\Lambda(A_1 - \lambda E_1) \subset \mathbb{C}_g$ and $\Lambda(A_2 - \lambda E_2) \subset \mathbb{C} \setminus \mathbb{C}_g$. This leads to the additive decomposition of $G(\lambda)$ as

$$G(\lambda) = G_1(\lambda) + G_2(\lambda), \tag{174}$$

where

$$G_1(\lambda) = \left[\begin{array}{c|c} A_1 - \lambda E_1 & B_1 \\ \hline C_1 & D \end{array}\right], \quad G_2(\lambda) = \left[\begin{array}{c|c} A_2 - \lambda E_2 & B_2 \\ \hline C_2 & 0 \end{array}\right]. \tag{175}$$

The computation of the descriptor realization (173), with a block-diagonal pole pencil, is achieved in two steps. The first step involves the separation of the spectrum of $A - \lambda E$, using orthogonal transformations, in accordance with the selected option in `OPTIONS.job`, which defines the domain of interest $\mathbb{C}_g$. In all cases, a preliminary finite-infinite or infinite-finite separation of the eigenvalues of the pole pencil $A - \lambda E$ is performed (only for descriptor systems) using the orthogonal staircase reduction algorithm proposed in [26] (for the computation of system zeros). The finite part of the reduced pole pencil is further reduced to a GRSF, and if necessary, the finite eigenvalues are additionally separated into stable-unstable or unstable-stable blocks using eigenvalue reordering techniques (see [16]). In the second step, the block-diagonalization of the reduced pole pencil is performed using the approach of [21].

Four spectral separations of the poles of $G(\lambda)$ can be selected via the option parameter `OPTIONS.job`. The resulting pole pencils of the corresponding descriptor system realizations (171) of $G_1(\lambda)$ and (172) of $G_2(\lambda)$, exhibit several particular features, which are shortly addressed below:

If `OPTIONS.job = 'finite'`, the resulting A_2 is nonsingular and upper triangular, while the resulting E_2 is nilpotent and upper triangular.

If `OPTIONS.job = 'infinite'`, the resulting A_1 is nonsingular and upper triangular, while the resulting E_1 is nilpotent and upper triangular. In this case, `SYS2` contains the strictly proper part of `SYS`.

If `OPTIONS.job = 'stable'`, the resulting pair (A_2, E_2), in a GRSF, contains also all infinite eigenvalues of the pole pencil $A - \lambda E$ (the trailing part of E_2 contains a nilpotent matrix).

If `OPTIONS.job = 'unstable'`, the resulting pair (A_1, E_1), in a GRSF, contains also all infinite eigenvalues of the pole pencil $A - \lambda E$ (the leading part of E_1 contains a nilpotent matrix).

Example

Example 10. Consider the 2×2 improper transfer function matrix $G(s)$ of a continuous-time system

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

To compute the proper-polynomial additive decomposition of $G(s)$, we can use the following command sequence:

```
s = tf('s');              % define the complex variable s
Gc = [s^2 s/(s+1); 0 1/s] % define the 2-by-2 improper Gc(s)
sysc = ss(Gc);            % build continuous-time descriptor system realization

% compute the separation of proper and polynomial parts of Gc(s)
[sysf,sysi] = gsdec(sysc);

% for checking the results, convert the terms to zeros/poles/gain form
Gf = zpk(sysf)   % proper part
Gp = zpk(sysi)   % polynomial part
```

The resulting transfer function matrices of the proper part $G_f(s)$ and polynomial part $G_p(s)$ are

$$G_f(s) = \begin{bmatrix} 0 & \dfrac{s}{s+1} \\ 0 & \dfrac{1}{s} \end{bmatrix}, \qquad G_p(s) = \begin{bmatrix} s^2 & 0 \\ 0 & 0 \end{bmatrix}.$$

$\Diamond$

Example 11. For the transfer function matrix employed in Example 10, we can compute a polynomial-strictly proper additive decomposition of $G(s)$, using the following command sequence:

```
s = tf('s');              % define the complex variable s
Gc = [s^2 s/(s+1); 0 1/s] % define the 2-by-2 improper Gc(s)
sysc = ss(Gc);            % build continuous-time descriptor system realization

% compute the separation of polynomial and strictly proper parts of Gc(s)
[syspol,syssp] = gsdec(sysc,struct('job','infinite'));

% for checking the results, convert the terms to zeros/poles/gain form
Gsp  = zpk(syssp)   % strictly proper part
Gpol = zpk(syspol)  % polynomial part
```

The resulting strictly proper part $G_{sp}(s)$ and polynomial part $G_{pol}(s)$ are

$$G_{sp}(s) = \begin{bmatrix} 0 & -\dfrac{1}{s+1} \\ 0 & \dfrac{1}{s} \end{bmatrix}, \qquad G_{pol}(s) = \begin{bmatrix} s^2 & 1 \\ 0 & 0 \end{bmatrix}.$$

$\Diamond$

3.5.9 grmcover1

Syntax

```
[SYSX,INFO,SYSY] = grmcover1(SYS1,SYS2,TOL)
[SYSX,INFO,SYSY] = grmcover1(SYS,M1,TOL)
```

Description

`grmcover1` computes, for given proper transfer function matrices $X_1(\lambda)$ and $X_2(\lambda)$, a proper $X(\lambda)$ and a strictly proper $Y(\lambda)$, both with minimal McMillan degrees, which satisfy

$$X(\lambda) = X_1(\lambda) + X_2(\lambda)Y(\lambda) \tag{176}$$

and represent the solution of a right minimum cover problem. An approach based on the computation of a minimum dynamic cover of Type 1 is used to determine $X(\lambda)$ and $Y(\lambda)$, such that $\delta(X(\lambda)) \leq \delta(X_1(\lambda))$. If $X_1(\lambda)$ is improper, then a maximum reduction of $\delta(X(\lambda))$ is achieved, by maximally reducing the order of the proper part of $X_1(\lambda)$ and keeping unaltered its polynomial part.

Input data

For the usage with

```
[SYSX,INFO,SYSY] = grmcover1(SYS1,SYS2,TOL)
```

the input parameters are as follows:

`SYS1` is a LTI system, whose transfer function matrix is $X_1(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_1\lambda x_1(t) &= A_1 x_1(t) + B_1 u(t), \\ y_1(t) &= C_1 x_1(t) + D_1 u(t), \end{aligned} \tag{177}$$

where $y_1(t) \in \mathbb{R}^p$ and $u(t) \in \mathbb{R}^{m_1}$.

`SYS2` is a proper LTI system, whose transfer function matrix is $X_2(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_2\lambda x_2(t) &= A_2 x_2(t) + B_2 v(t), \\ y_2(t) &= C_2 x_2(t) + D_2 v(t), \end{aligned} \tag{178}$$

where $y_2(t) \in \mathbb{R}^p$.

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

For the usage with

```
[SYSX,INFO,SYSY] = grmcover1(SYS,M1,TOL)
```

the input parameters are as follows:

SYS is an input concatenated compound LTI system, SYS = [SYS1 SYS2], in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_1u(t) + B_2v(t), \\ y(t) &= Cx(t) + D_1u(t) + D_2v(t), \end{aligned} \tag{179}$$

where SYS1 has the transfer function matrix $X_1(\lambda)$, with the descriptor system realization $(A - \lambda E, B_1, C, D_1)$, and SYS2 has the proper transfer function matrix $X_2(\lambda)$, with the descriptor system realization $(A - \lambda E, B_2, C, D_2)$.

M1 is the dimension m_1 of the input vector $u(t)$ of the system SYS1.

TOL is a relative tolerance used for rank determinations. If TOL is not specified as input or if TOL $= 0$, an internally computed default value is used.

Output data

SYSX contains, in the case when both $X_1(\lambda)$ and $X_2(\lambda)$ are proper, the descriptor system state-space realization of the resulting reduced order $X(\lambda)$ in (176), in the form

$$\begin{aligned} E_r\lambda x_{r,1}(t) &= A_r x_{r,1}(t) + B_r v(t), \\ y_{r,1}(t) &= C_{r,1}x_{r,1}(t) + D_1 v(t), \end{aligned} \tag{180}$$

with the pair $(A_r - \lambda E_r, B_r)$ in a controllability staircase form with $[\, B_r \;\; A_r \,]$ as in (22) and E_r upper triangular and nonsingular, as in (23). If $X_1(\lambda)$ is improper, but $X_2(\lambda)$ is proper, then SYSX contains the descriptor system realization corresponding to the sum of the reduced proper part of $X_1(\lambda)$ (in the form (180)) and the polynomial part of $X_1(\lambda)$.

INFO is a MATLAB structure containing additional information, as follows:

INFO fields	Description
stdim	vector which contains the dimensions of the diagonal blocks of $A_r - \lambda E_r$, which are the row dimensions of the full row rank diagonal blocks of the pencil $[\, B_r \;\; A_r - \lambda E_r \,]$ in controllability staircase form;
tcond	maximum of the Frobenius-norm condition numbers of the employed non-orthogonal transformation matrices (large values indicate possible loss of numerical stability);
fnorm	the Frobenius-norm of the employed state-feedback F used for minimum dynamic cover computation (see **Method**) (large values indicate possible loss of numerical stability).

SYSY contains the descriptor system state-space realization of the resulting (strictly proper) $Y(\lambda)$ in (176), in the form

$$\begin{aligned} E_r\lambda x_{r,2}(t) &= A_r x_{r,2}(t) + B_r v(t), \\ y_{r,2}(t) &= C_{r,2}x_{r,2}(t), \end{aligned} \tag{181}$$

with the pair $(A_r - \lambda E_r, B_r)$ in a controllability staircase form with $[\, B_r \;\; A_r \,]$ as in (22) and E_r upper triangular and nonsingular, as in (23).

Method

The approach to determine $X(\lambda)$ and $Y(\lambda)$, with least McMillan degrees, satisfying (176) with both $X_1(\lambda)$ and $X_2(\lambda)$ proper, is based on computing a Type 1 minimum dynamic cover [23] and is described in Section 2.11. This approach is based on a realization of the form (179) of the input concatenated proper compound system `SYS = [ SYS1 SYS2 ]`, with invertible E. A detailed computational procedure to determine $X(\lambda)$ in (176) is described in [54] (see also **Procedure GRMCOVER1** in [59, Section 10.4.2] for more details). For the intervening reduction of the partitioned pair $(A - \lambda E, [\, B_1 \;\; B_2 \,]$ to a special controllability staircase form (see **Procedure GSCSF** in [59, Section 10.4.1]), the mex-function `sl_gstra` is employed. If the descriptor realizations of `SYS1` and `SYS2` are separately provided, then an irreducible realization of the input concatenated compound system `[ SYS1 SYS2 ]` is internally computed. For orthogonal transformation based order reduction purposes, the mex-function `sl_gminr` is employed.

For the given descriptor system realization of the transfer function matrix $[\, X_1(\lambda) \;\; X_2(\lambda) \,]$ in the form (179), a state-feedback matrix F is determined to obtain the resulting reduced order $X(\lambda)$ and $Y(\lambda)$ in (176) with the descriptor realization

$$\left[\begin{array}{c} X(\lambda) \\ Y(\lambda) \end{array}\right] := \left[\begin{array}{c|c} A + B_2F - \lambda E & B_1 \\ \hline C + D_2F & D_1 \\ F & 0 \end{array}\right] . \tag{182}$$

The matrix F is determined such that the pair $(A + B_2F - \lambda E, B_1)$ is maximally uncontrollable. Then, the resulting realizations of $X(\lambda)$ and $Y(\lambda)$ contain a maximum number of uncontrollable eigenvalues, which are eliminated by the Type 1 dynamic cover computation algorithm [54]. This algorithm involves the use of non-orthogonal similarity transformations, whose maximum Frobenius-norm condition number is provided in `INFO.tcond`. The resulting minimal realizations of $X(\lambda)$ and $Y(\lambda)$ have the forms (187) and (188), respectively.

As already mentioned, the underlying **Procedure GRMCOVER1** of [59, Section 10.4.2] is based on the algorithm proposed in [54] for descriptor systems with invertible E. However, the function `grmcover1` also works if the original descriptor system realization has E singular, but `SYS2` is proper. In this case, the order reduction is performed working only with the proper part of `SYS1`. The polynomial part of `SYS1` is included, without modification, in the resulting realization of `SYSX`. To compute the intervening finite-infinite spectral separation, the mex-function `sl_gsep` is employed.

Examples

Example 12. For a given transfer function matrix $G(\lambda)$, a simple minimal proper right nullspace basis can be computed in two steps: first, compute a minimal proper right nullspace basis $N_r(\lambda)$ of $G(\lambda)$, and then in a second step, determine successively the basis vectors of a minimal simple proper basis $N_r^s(\lambda)$ by solving a sequence of right minimal cover problems. Specifically, if $\mathrm{col}_i(N_r(\lambda))$ is the i-th basis vector (i.e., column) of $N_r(\lambda)$ and $V_i^c(\lambda)$ is the matrix formed from the rest of basis vectors, then the i-th basis vector $\mathrm{col}_i(N_r^s(\lambda))$ of a minimal simple proper basis

$N_r^s(\lambda)$ is the solution of the right cover problem

$$\mathrm{col}_i(N_r^s(\lambda)) = \mathrm{col}_i(N_r(\lambda)) + V_i^c(\lambda)Y_i(\lambda),$$

where $Y_i(\lambda)$ is a suitable strictly proper vector of least McMillan degree.

The following MATLAB commands illustrate this approach:

```
% computation of a minimal simple proper right nullspace basis
sys = rss(6,2,6);  % generate a random system with a 2x6 TFM

% compute a minimal proper right nullspace basis Nr with a 6x4 TFM
[Nr,info] = grnull(sys);
info.degs        % the expected orders of vectors of a minimal simple basis
nb = size(Nr,2); % number of basis vectors

% let Vi be the i-th basis vector and Vci the rest of basis vectors in Nr
% to compute the i-th basis vector Vsi of a simple basis Nrs,
% solve the right minimum cover problem:
% Vsi = Vi + Vci*Yi (for a suitable strictly proper Yi)

Nrs = ss(zeros(size(Nr)));   % initialize Nrs
for i = 1:nb;
    % apply the cover computation to Nr with permuted columns
    Vsi = grmcover1(Nr(:,[i,1:i-1,i+1:nb]),1);
    % check orders
    if order(Vsi) ~= info.degs(i)
       warning('Expected order not achieved')
    end
    Nrs(:,i) = Vsi;
end

% check nullspace condition sys*Nrs = 0
gnrank(sys*Nrs,1.e-7,rand)
```

The same approach can be used by calling the function `grmcover2` instead `grmcover1`. ♢

Example 13. For a given transfer function matrix $G(\lambda)$, a minimal polynomial right nullspace basis can be computed in three steps: (1) compute a minimal proper right nullspace basis $N_r(\lambda)$ of $G(\lambda)$; (2) determine successively the basis vectors of a minimal simple proper basis $N_r^s(\lambda)$ by solving a sequence of right minimal cover problems; and (3) determine the polynomial numerator of each vector of the simple basis. The first two steps have been already described in Example 12.

The following MATLAB commands illustrates the three step approach to compute polynomial bases:

```
% minimal polynomial nullspace computation
sys = rss(6,2,6);  % generate a random system with a 2x6 TFM

% compute a proper right nullspace basis Nr with a 6x4 TFM
[Nr,info] = grnull(sys);
info.degs        % the degrees of vectors of a minimal polynomial basis
nb = size(Nr,2); % number of basis vectors
% let Vi the i-th basis vector and Vci the rest of basis vectors in Nr
% to compute the i-th basis vector Vsi of a simple basis Nrs,
% solve the right minimum cover problem:
% Vsi = Vi + Vci*Yi (for a suitable strictly proper Yi)
% to obtain the corresponding polynomial basis vector, cancel all finite poles
% of Vsi
%
Nrp = tf(zeros(6,nb));   % initialize Nrs
for i = 1:nb;
    % apply the cover computation to Nr with permuted columns
    Vsi = grmcover1(Nr(:,[i,1:i-1,i+1:nb]),1);
    % check orders
    if order(Vsi) ~= info.degs(i)
       warning('Expected order not achieved')
    end
    Nrp(:,i) = minreal(tf(Vsi*tf(poly(eig(Vsi)),1)));
end

% check nullspace condition sys*Nrp = 0
gnrank(sys*Nrp,1.e-7,rand)
```

The same approach can be used by calling the function `grmcover2` instead `grmcover1`. ◊

3.5.10 `glmcover1`

Syntax

```
[SYSX,INFO,SYSY] = glmcover1(SYS1,SYS2,TOL)
[SYSX,INFO,SYSY] = glmcover1(SYS,P1,TOL)
```

Description

`glmcover1` computes, for given proper transfer function matrices $X_1(\lambda)$ and $X_2(\lambda)$, a proper $X(\lambda)$ and a strictly proper $Y(\lambda)$, both with minimal McMillan degrees, which satisfy

$$X(\lambda) = X_1(\lambda) + Y(\lambda)X_2(\lambda), \tag{183}$$

and represent the solution of a left minimum cover problem. An approach based on the computation of a minimum dynamic cover of Type 1 is used to determine $X(\lambda)$ and $Y(\lambda)$, such that $\delta(X(\lambda)) \leq \delta(X_1(\lambda))$. If $X_1(\lambda)$ is improper, then a maximum reduction of $\delta(X(\lambda))$ is achieved, by maximally reducing the order of the proper part of $X_1(\lambda)$ and keeping unaltered its polynomial part.

Input data

For the usage with

```
[SYSX,INFO,SYSY] = glmcover1(SYS1,SYS2,TOL)
```

the input parameters are as follows:

`SYS1` is a LTI system, whose transfer function matrix $X_1(\lambda)$ has a descriptor system state-space realization of the form

$$\begin{aligned} E_1\lambda x_1(t) &= A_1 x_1(t) + B_1 u(t), \\ y_1(t) &= C_1 x_1(t) + D_1 u(t), \end{aligned} \tag{184}$$

where $y_1(t) \in \mathbb{R}^{p_1}$ and $u(t) \in \mathbb{R}^m$.

`SYS2` is a proper LTI system, whose transfer function matrix $X_2(\lambda)$ has a descriptor system state-space realization of the form

$$\begin{aligned} E_2\lambda x_2(t) &= A_2 x_2(t) + B_2 v(t), \\ y_2(t) &= C_2 x_2(t) + D_2 v(t), \end{aligned} \tag{185}$$

where $v(t) \in \mathbb{R}^m$.

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

For the usage with

```
[SYSX,INFO,SYSY] = glmcover1(SYS,P1,TOL)
```

the input parameters are as follows:

`SYS` is an output concatenated compound LTI system, `SYS = [ SYS1; SYS2 ]`, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y_1(t) &= C_1 x(t) + D_1 u(t), \\ y_2(t) &= C_2 x(t) + D_2 u(t), \end{aligned} \tag{186}$$

where `SYS1` has the transfer function matrix $X_1(\lambda)$, with the descriptor system realization $(A - \lambda E, B, C_1, D_1)$, and `SYS2` has the proper transfer function matrix $X_2(\lambda)$, with the descriptor system realization $(A - \lambda E, B, C_2, D_2)$.

`P1` is the dimension p_1 of the output vector $y_1(t)$ of the system `SYS1`.

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

Output data

`SYSX` contains, in the case when both $X_1(\lambda)$ and $X_2(\lambda)$ are proper, the descriptor system state-space realization of the resulting reduced order $X(\lambda)$ in (183), in the form

$$\begin{aligned} E_l\lambda x_{l,1}(t) &= A_l x_{l,1}(t) + B_{l,1}v(t), \\ y_{l,1}(t) &= C_l x_{l,1}(t) + D_1 v(t), \end{aligned} \tag{187}$$

with the pair $(A_l - \lambda E_l, C_l)$ in an observability staircase form with $\left[\begin{smallmatrix} A_l \\ C_l \end{smallmatrix}\right]$ as in (24) and E_l upper triangular and nonsingular, as in (25). If $X_1(\lambda)$ is improper, but $X_2(\lambda)$ is proper, then `SYSX` contains the descriptor system realization corresponding to the sum of the reduced proper part of $X_1(\lambda)$ (in the form (187)) and the polynomial part of $X_1(\lambda)$.

`INFO` is a MATLAB structure containing additional information, as follows:

`INFO` fields	Description
`stdim`	vector containing the dimensions of the diagonal blocks of $A_l - \lambda E_l$, which are the column dimensions of the full column rank diagonal blocks of the pencil $\left[\begin{smallmatrix} A_l - \lambda E_l \\ C_l \end{smallmatrix}\right]$ in observability staircase form;
`tcond`	maximum of the Frobenius-norm condition numbers of the employed non-orthogonal transformation matrices (large values indicate possible loss of numerical stability);
`fnorm`	the Frobenius-norm of the employed state-feedback F used for minimum dynamic cover computation (see **Method**) (large values indicate possible loss of numerical stability).

`SYSY` contains the descriptor system state-space realization of the resulting (strictly proper) $Y(\lambda)$ in (183), in the form

$$\begin{aligned} E_l\lambda x_{l,2}(t) &= A_l x_{l,2}(t) + B_{l,2}v(t), \\ y_{l,2}(t) &= C_l x_{l,2}(t), \end{aligned} \tag{188}$$

with the pair $(A_l - \lambda E_l, C_l)$ in an observability staircase form with $\left[\begin{smallmatrix} A_l \\ C_l \end{smallmatrix}\right]$ as in (24) and E_l upper triangular and nonsingular, as in (25).

Method

To determine $X(\lambda)$ and $Y(\lambda)$, with least McMillan degrees, satisfying (183), the function `glmcover1` calls `grmcover1` to determine the dual quantities $\widetilde{X}(\lambda)$ and $\widetilde{Y}(\lambda)$ satisfying

$$\widetilde{X}(\lambda) = X_1^T(\lambda) + X_2^T(\lambda)\widetilde{Y}(\lambda)$$

and obtain $X(\lambda) = \widetilde{X}^T(\lambda)$ and $Y(\lambda) = \widetilde{Y}^T(\lambda)$. Thus, the employed solution method corresponds to the dual of the approach sketched in Section 2.11 (see also [54] and **Procedure GRMCOVER1** in [59, Section 10.4.2] for more details). The function `grmcover1` relies on

the mex-functions `sl_gstra` to compute a special controllability staircase form (see **Procedure GSCSF** in [59, Section 10.4.1]) and `sl_gminr` to compute irreducible realizations.

For the given descriptor system realization of the transfer function matrix $\left[\begin{smallmatrix} X_1(\lambda) \\ X_2(\lambda) \end{smallmatrix}\right]$ in the form (186), an output injection F is determined to obtain the resulting reduced order $X(\lambda)$ and $Y(\lambda)$ in (183) with the descriptor realization

$$[\, X(\lambda) \;\; Y(\lambda) \,] := \left[\begin{array}{c|cc} A + FC_2 - \lambda E & B + FD_2 & F \\ \hline C_1 & D_1 & 0 \end{array}\right] . \tag{189}$$

The matrix F is determined such that the pair $(A + FC_2 - \lambda E, C_1)$ is maximally unobservable. Then, the resulting realizations of $X(\lambda)$ and $Y(\lambda)$ contain a maximum number of unobservable eigenvalues, which are eliminated by the Type 1 dynamic cover computation algorithm [54]. This algorithm involves the use of non-orthogonal similarity transformations, whose maximum Frobenius-norm condition number is provided in `INFO.tcond`. The resulting minimal realizations of $X(\lambda)$ and $Y(\lambda)$ have the forms (187) and (188), respectively.

The underlying **Procedure GRMCOVER1** of [59, Section 10.4.2], used in a dual setting, is based on the algorithm proposed in [54] for descriptor systems with invertible E. However, the function `glmcover1` also works if the original descriptor system realization has E singular, but `SYS2` is proper. In this case, the order reduction is performed working only with the proper part of `SYS1`. The polynomial part of `SYS1` is included, without modification, in the resulting realization of `SYSX`.

3.5.11 `grmcover2`

Syntax

```
[SYSX,INFO,SYSY] = grmcover2(SYS1,SYS2,TOL)
[SYSX,INFO,SYSY] = grmcover2(SYS,M1,TOL)
```

Description

`grmcover2` computes, for given proper transfer function matrices $X_1(\lambda)$ and $X_2(\lambda)$, a proper $X(\lambda)$ and a proper $Y(\lambda)$, both with minimal McMillan degrees, which satisfy

$$X(\lambda) = X_1(\lambda) + X_2(\lambda)Y(\lambda) \tag{190}$$

and represent the solution of a right minimum cover problem. An approach based on the computation of a minimum dynamic cover of Type 2 is used to determine $X(\lambda)$ and $Y(\lambda)$, such that $\delta(X(\lambda)) \leq \delta(X_1(\lambda))$. If $X_1(\lambda)$ is improper, then a maximum reduction of $\delta(X(\lambda))$ is achieved, by maximally reducing the order of the proper part of $X_1(\lambda)$ and keeping unaltered its polynomial part.

Input data

For the usage with

```
[SYSX,INFO,SYSY] = grmcover2(SYS1,SYS2,TOL)
```

the input parameters are as follows:

`SYS1` is a LTI system, whose transfer function matrix $X_1(\lambda)$ has a descriptor system state-space realization of the form

$$\begin{aligned} E_1\lambda x_1(t) &= A_1 x_1(t) + B_1 u(t), \\ y_1(t) &= C_1 x_1(t) + D_1 u(t), \end{aligned} \tag{191}$$

where $y_1(t) \in \mathbb{R}^p$ and $u(t) \in \mathbb{R}^{m_1}$.

`SYS2` is a proper LTI system, whose transfer function matrix $X_2(\lambda)$ has a descriptor system state-space realization of the form

$$\begin{aligned} E_2\lambda x_2(t) &= A_2 x_2(t) + B_2 v(t), \\ y_2(t) &= C_2 x_2(t) + D_2 v(t), \end{aligned} \tag{192}$$

where $y_2(t) \in \mathbb{R}^p$.

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

For the usage with

```
[SYSX,INFO,SYSY] = grmcover2(SYS,M1,TOL)
```

the input parameters are as follows:

`SYS` is an input concatenated compound LTI system, `SYS = [ SYS1 SYS2 ]`, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_1 u(t) + B_2 v(t), \\ y(t) &= Cx(t) + D_1 u(t) + D_2 v(t), \end{aligned} \tag{193}$$

where `SYS1` has the transfer function matrix $X_1(\lambda)$ with the descriptor system realization $(A - \lambda E, B_1, C, D_1)$ and `SYS2` has the proper transfer function matrix $X_2(\lambda)$ with the descriptor system realization $(A - \lambda E, B_2, C, D_2)$.

`M1` is the dimension m_1 of the input vector $u(t)$ of the system `SYS1`.

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

Output data

`SYSX` contains, in the case when both $X_1(\lambda)$ and $X_2(\lambda)$ are proper, the descriptor system state-space realization of the resulting reduced order $X(\lambda)$ in (190), in the form

$$\begin{aligned} E_r\lambda x_{r,1}(t) &= A_r x_{r,1}(t) + B_r v(t), \\ y_{r,1}(t) &= C_{r,1} x_{r,1}(t) + D_1 v(t), \end{aligned} \tag{194}$$

with the pair $(A_r - \lambda E_r, B_r)$ in a controllability staircase form with $[\, B_r \;\; A_r \,]$ as in (22) and E_r upper triangular and nonsingular, as in (23). If $X_1(\lambda)$ is improper, but $X_2(\lambda)$ is proper, then `SYSX` contains the descriptor system realization corresponding to the sum of the reduced proper part of $X_1(\lambda)$ (in the form (194)) and the polynomial part of $X_1(\lambda)$.

`INFO` is a MATLAB structure containing additional information, as follows:

INFO fields	Description
stdim	vector which contains the dimensions of the diagonal blocks of $A_r - \lambda E_r$, which are the row dimensions of the full row rank diagonal blocks of the pencil $[\, B_r \;\; A_r - \lambda E_r \,]$ in controllability staircase form;
tcond	maximum of the Frobenius-norm condition numbers of the employed non-orthogonal transformation matrices (large values indicate possible loss of numerical stability);
fnorm	the Frobenius-norm of the employed state-feedback F used for minimum dynamic cover computation (see **Method**) (large values indicate possible loss of numerical stability);
gnorm	the Frobenius-norm of the employed feedforward matrix G used for minimum dynamic cover computation (see **Method**) (large values indicate possible loss of numerical stability).

SYSY contains the descriptor system state-space realization of the resulting $Y(\lambda)$ in (190), in the form

$$\begin{aligned} E_r \lambda x_r(t) &= A_r x_{r,2}(t) + B_r v(t), \\ y_{r,2}(t) &= C_{r,2} x_{r,2}(t) + D_{r,2} v(t), \end{aligned} \tag{195}$$

with the pair $(A_r - \lambda E_r, B_r)$ in a controllability staircase form with $[\, B_r \;\; A_r \,]$ as in (22) and E_r upper triangular and nonsingular, as in (23).

Method

The approach to determine $X(\lambda)$ and $Y(\lambda)$, with least McMillan degrees, satisfying (176), with both $X_1(\lambda)$ and $X_2(\lambda)$ proper, is based on computing a Type 2 minimum dynamic cover [23] and is described in Section 2.11. This approach is based on a realization of the form (193) of the input concatenated compound proper system SYS = [SYS1 SYS2], with invertible E. A detailed computational procedure to determine $X(\lambda)$ in (190) is described in [54] (see also **Procedure GRMCOVER2** in [59, Section 10.4.2] for more details). For the intervening reduction of the partitioned pair $(A - \lambda E, [\, B_2 \;\; B_1 \,]$ to a special controllability staircase form (see **Procedure GSCSF** in [59, Section 10.4.1]), the mex-function sl_gstra is employed. If the descriptor realizations of SYS1 and SYS2 are separately provided, then an irreducible realization of the input concatenated compound system [SYS1 SYS2] is internally computed. For orthogonal transformation based order reduction purposes, the mex-function sl_gminr is employed.

For the given descriptor system realization of the transfer function matrix $[\, X_1(\lambda) \;\; X_2(\lambda) \,]$ in the form (193), a state-feedback matrix F and a feedforward gain G are determined to obtain the resulting reduced order $X(\lambda)$ and $Y(\lambda)$ in (190) with the descriptor realization

$$\left[\begin{array}{c} X(\lambda) \\ Y(\lambda) \end{array} \right] := \left[\begin{array}{c|c} A + B_2 F - \lambda E & B_1 + B_2 G \\ \hline C + D_2 F & D_1 + D_2 G \\ F & G \end{array} \right]. \tag{196}$$

The feedback matrix F and feedforward matrix G are determined such that the descriptor pair $(A + B_2 F - \lambda E, B_1 + B_2 G)$ is maximally uncontrollable. Then, the resulting realizations of $X(\lambda)$ and $Y(\lambda)$ contain a maximum number of uncontrollable eigenvalues, which are eliminated by

the Type 2 dynamic cover computation algorithm of [54] for descriptor systems and of [52] for standard systems. This algorithm involves the use of non-orthogonal similarity transformations, whose maximum Frobenius-norm condition number is provided in `INFO.tcond`. The resulting minimal realizations of $X(\lambda)$ and $Y(\lambda)$ have the forms (194) and (195), respectively.

As already mentioned, the underlying **Procedure GRMCOVER2** of [59, Section 10.4.2] is based on the algorithm proposed in [54] for descriptor systems with invertible E. However, the function `grmcover2` also works if the original descriptor system realization has E singular, but `SYS2` is proper. In this case, the order reduction is performed working only with the proper part of `SYS1`. The polynomial part of `SYS1` is included, without modification, in the resulting realization of `SYSX`. To compute the intervening finite-infinite spectral separation, the mex-function `sl_gsep` is employed.

3.5.12 `glmcover2`

Syntax

```
[SYSX,INFO,SYSY] = glmcover2(SYS1,SYS2,TOL)
[SYSX,INFO,SYSY] = glmcover2(SYS,P1,TOL)
```

Description

`glmcover2` computes, for given proper transfer function matrices $X_1(\lambda)$ and $X_2(\lambda)$, a proper $X(\lambda)$ and a proper $Y(\lambda)$, both with minimal McMillan degrees, which satisfy

$$X(\lambda) = X_1(\lambda) + Y(\lambda)X_2(\lambda), \tag{197}$$

and represent the solution of a left minimum cover problem. An approach based on the computation of a minimum dynamic cover of Type 2 is used to determine $X(\lambda)$ and $Y(\lambda)$, such that $\delta(X(\lambda)) \leq \delta(X_1(\lambda))$. If $X_1(\lambda)$ is improper, then a maximum reduction of $\delta(X(\lambda))$ is achieved, by maximally reducing the order of the proper part of $X_1(\lambda)$ and keeping unaltered its polynomial part.

Input data

For the usage with

```
[SYSX,INFO,SYSY] = glmcover2(SYS1,SYS2,TOL)
```

the input parameters are as follows:

`SYS1` is a LTI system, whose transfer function matrix $X_1(\lambda)$ has a descriptor system state-space realization of the form

$$\begin{aligned} E_1\lambda x_1(t) &= A_1x_1(t) + B_1u(t), \\ y_1(t) &= C_1x_1(t) + D_1u(t), \end{aligned} \tag{198}$$

where $y_1(t) \in \mathbb{R}^{p_1}$ and $u(t) \in \mathbb{R}^m$.

`SYS2` is a proper LTI system, whose transfer function matrix $X_2(\lambda)$ has a descriptor system state-space realization of the form

$$\begin{aligned} E_2\lambda x_2(t) &= A_2x_2(t) + B_2v(t), \\ y_2(t) &= C_2x_2(t) + D_2v(t), \end{aligned} \tag{199}$$

where $v(t) \in \mathbb{R}^m$.

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

For the usage with

```
[SYSX,INFO,SYSY] = glmcover2(SYS,P1,TOL)
```

the input parameters are as follows:

`SYS` is an output concatenated compound LTI system, `SYS = [ SYS1; SYS2 ]`, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y_1(t) &= C_1x(t) + D_1u(t), \\ y_2(t) &= C_2x(t) + D_2u(t), \end{aligned} \tag{200}$$

where `SYS1` has the transfer function matrix $X_1(\lambda)$ with the descriptor system realization $(A - \lambda E, B, C_1, D_1)$ and `SYS2` has the proper transfer function matrix $X_2(\lambda)$ with the descriptor system realization $(A - \lambda E, B, C_2, D_2)$.

`P1` is the dimension p_1 of the output vector $y_1(t)$ of the system `SYS1`.

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

Output data

`SYSX` contains, in the case when both $X_1(\lambda)$ and $X_2(\lambda)$ are proper, the descriptor system state-space realization of the resulting reduced order $X(\lambda)$ in (197), in the form

$$\begin{aligned} E_l\lambda x_{l,1}(t) &= A_l x_{l,1}(t) + B_{l,1}v(t), \\ y_{l,1}(t) &= C_l x_{l,1}(t) + D_1 v(t), \end{aligned} \tag{201}$$

with the pair $(A_l - \lambda E_l, C_l)$ in an observability staircase form with $\left[\begin{smallmatrix} A_l \\ C_l \end{smallmatrix}\right]$ as in (24) and E_l upper triangular and nonsingular, as in (25). If $X_1(\lambda)$ is improper, but $X_2(\lambda)$ is proper, then `SYSX` contains the descriptor system realization corresponding to the sum of the reduced proper part of $X_1(\lambda)$ (in the form (201)) and the polynomial part of $X_1(\lambda)$.

`INFO` is a MATLAB structure containing additional information, as follows:

`INFO` fields	Description
`stdim`	vector containing the dimensions of the diagonal blocks of $A_l - \lambda E_l$, which are the column dimensions of the full column rank diagonal blocks of the pencil $\left[\begin{smallmatrix} A_l - \lambda E_l \\ C_l \end{smallmatrix}\right]$ in observability staircase form;
`tcond`	maximum of the Frobenius-norm condition numbers of the employed non-orthogonal transformation matrices (large values indicate possible loss of numerical stability);

`fnorm`	the Frobenius-norm of the employed state-feedback F used for minimum dynamic cover computation (see **Method**) (large values indicate possible loss of numerical stability).
`gnorm`	the Frobenius-norm of the employed feedforward matrix G used for minimum dynamic cover computation (see **Method**) (large values indicate possible loss of numerical stability).

`SYSY` contains the descriptor system state-space realization of the resulting $Y(\lambda)$ in (197), in the form

$$\begin{aligned} E_l\lambda x_{l,2}(t) &= A_l x_{l,2}(t) + B_{l,2}v(t), \\ y_{l,2}(t) &= C_l x_{l,2}(t) + D_{l,2}v(t), \end{aligned} \tag{202}$$

with the pair $(A_l - \lambda E_l, C_l)$ in an observability staircase form with $\left[\begin{smallmatrix} A_l \\ C_l \end{smallmatrix}\right]$ as in (24) and E_l upper triangular and nonsingular, as in (25).

Method

To determine $X(\lambda)$ and $Y(\lambda)$, with least McMillan degrees, satisfying (183), the function `glmcover2` calls `grmcover2` to determine the dual quantities $\widetilde{X}(\lambda)$ and $\widetilde{Y}(\lambda)$ satisfying

$$\widetilde{X}(\lambda) = X_1^T(\lambda) + X_2^T(\lambda)\widetilde{Y}(\lambda)$$

and obtain $X(\lambda) = \widetilde{X}^T(\lambda)$ and $Y(\lambda) = \widetilde{Y}^T(\lambda)$. Thus, the employed solution method corresponds to the dual of the approach sketched in Section 2.11 (see also [54] and **Procedure GRMCOVER2** in [59, Section 10.4.2] for more details). The function `grmcover2` relies on the mex-functions `sl_gstra` to compute a special controllability staircase form (see **Procedure GSCSF** in [59, Section 10.4.1]) and `sl_gminr` to compute irreducible realizations.

For the given descriptor system realization of the transfer function matrix $\left[\begin{smallmatrix} X_1(\lambda) \\ X_2(\lambda) \end{smallmatrix}\right]$ in the form (200), an output injection F is determined to obtain the resulting reduced order $X(\lambda)$ and $Y(\lambda)$ in (197) with the descriptor realization

$$[\, X(\lambda) \;\; Y(\lambda)\,] := \left[\begin{array}{c|cc} A + FC_2 - \lambda E & B + FD_2 & F \\ \hline C_1 + GC_2 & D_1 + GD_2 & G \end{array}\right]. \tag{203}$$

The matrices F and G are determined such that the pair $(A+FC_2-\lambda E, C_1+GC_2)$ is maximally unobservable. Then, the resulting realizations of $X(\lambda)$ and $Y(\lambda)$ contain a maximum number of unobservable eigenvalues, which are eliminated by the Type 2 dynamic cover computation algorithm [54]. This algorithm involves the use of non-orthogonal similarity transformations, whose maximum Frobenius-norm condition number is provided in `INFO.tcond`. The resulting minimal realizations of $X(\lambda)$ and $Y(\lambda)$ have the forms (201) and (202), respectively.

The underlying **Procedure GRMCOVER2** of [59, Section 10.4.2], used in a dual setting, is based on the algorithm proposed in [54] for descriptor systems with invertible E. However, the function `glmcover2` also works if the original descriptor system realization has E singular, but `SYS2` is proper. In this case, the order reduction is performed working only with the proper part of `SYS1`. The polynomial part of `SYS1` is included, without modification, in the resulting realization of `SYSX`.

Example

Example 14. This example, taken from [67], has been already considered in Example 7 to illustrate the computation of a left inverse of least McMillan degree of a full column rank transfer function matrix $G(s)$, using the function `glsol`. In this example, we illustrates an alternative way to compute a left inverse of least McMillan degree, using the generators of all solution of the equation $X(\lambda)G(\lambda) = I$. A generator of all solutions is formed from a pair $(X_0(s), N_l(s))$, where $X_0(\lambda)$ is any particular solution (i.e., any particular left inverse) and $N_l(s)$ is a proper basis of the left nullspace of $G(s)$. Such a generator can be also computed using `glsol`.

The determination of a least order left inverse can be alternatively formulated in terms of the solution of the following left minimal cover problem: for a given generator $(X_0(s), N_l(s))$, compute least order $X(s)$ and $Y(s)$ such that

$$X(s) = X_0(s) + N_l(s)Y(s).$$

Recall transfer function matrix used in Example 7

$$G(s) = \frac{1}{s^2+3s+2}\begin{bmatrix} s+1 & s+2 \\ s+3 & s^2+2s \\ s^2+3s & 0 \end{bmatrix},$$

which has a third order minimal state-space realization, which can be computed using the function `gir` (or `minreal`). This realization is used to compute, using `glsol`, a pair $(X_0(s), N_l(s))$ representing a generator of all solutions of $X(s)G(s) = I$. The function `glmcover2` is then used to solve the above left minimal cover problem, to determine a least McMillan order proper left inverse $X(s)$. The following MATLAB commands illustrate this approach:

```
% Wang and Davison Example (1973)
s = tf('s');
g = [ s+1 s+2; s+3 s^2+2*s; s^2+3*s 0 ]/(s^2+3*s+2);
sysg = gir(ss(g));

% compute a generator (X0,Nl) of all solutions of X(s)G(s) = I
[~,~,sysgen] = glsol(sysg,ss(eye(2)));  % X0 = sysgen(:,1:2), Nl = sysgen(:,3)
gpole(sysgen)   % the generator is proper (no infinite poles)

% compute a least order inverse as X(s) = X0(s)+NL(s)*Y(s), by using
% order reduction based on a left minimal dynamic cover
[sysx,~,sysy] = glmcover2(sysgen,2);
gpole(sysx)     % resulting 2nd (least order) left inverse is unstable

% check left inverse condition
gnrank(sysx*sysg-eye(2))
```

The computed solution is unstable. Since the resulting $Y(s)$ is strictly proper, the same approach can be used by calling the function `glmcover1` instead `glmcover2`. ◊

3.5.13 `gbilin`

Syntax

```
[SYST,SYSI1] = gbilin(SYS,SYS1)
```

Description

`gbilin` computes for a LTI descriptor system realization $(A-\lambda E, B, C, D)$ and a first order transfer function $g(\delta)$, a transformed descriptor system realization $(\widetilde{A}-\delta\widetilde{E}, \widetilde{B}, \widetilde{C}, \widetilde{D})$ corresponding to the bilinear transformation $\lambda = g(\delta)$.

Input data

`SYS` is a LTI system, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{204}$$

`SYS1` is a LTI system of McMillan degree one, given as a real transfer function of the form

$$g(\delta) = \frac{a\delta + b}{c\delta + d}, \tag{205}$$

where, for a continuous-time system `SYS1`, $\delta = s$, the complex variable in the Laplace transform, while for a discrete-time system `SYS1`, $\delta = z$, the complex variable in the Z-transform. The parameters a, b, c and d must be real and must satisfy $ad - bd \neq 0$ (to prevent cancellation). The transformation (205) is a particular case of the *Möbius transformation* (with complex parameters).

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	**Description**
`tol`	tolerance for rank determinations (Default: internally computed)
`compact`	option to compute a compact descriptor system realization for the transformed system, without non-dynamic modes: `true` – determine a compact descriptor realization (default); `false` – disable elimination of non-dynamic modes.
`minimal`	option to compute a minimal descriptor system realization for the transformed system: `true` – determine a standard state-space realization; `false` – no minimal realization computed (default)
`ss`	option to compute a standard state-space realization (if possible) for the transformed system: `true` – determine a standard state-space realization; `false` – determine a descriptor system realization (default)

Output data

`SYST` contains the resulting transformed system in a descriptor system state-space form

$$\begin{aligned} \widetilde{E}\delta\widetilde{x}(t) &= \widetilde{A}\widetilde{x}(t) + \widetilde{B}u(t), \\ y(t) &= \widetilde{C}x(t) + \widetilde{D}u(t). \end{aligned} \tag{206}$$

corresponding to the bilinear transformation $\lambda = g(\delta)$. It follows, that if `SYS` has the TFM $G(\lambda)$, then `SYST` has the TFM $G(g(\delta))$. If the original system (204) has a standard state-space representation with $E = I$ and $g(\delta)$ is a first order polynomial, then the resulting system (206) is also determined in a standard state-space form with $\widetilde{E} = I$.

`SYSI1` is a LTI system of McMillan degree one, obtained as a transfer function of the form

$$g^{-1}(\lambda) = \frac{d\lambda - b}{-c\lambda + a}, \tag{207}$$

representing the inverse of the bilinear transformation $\lambda = g(\delta)$.

Method

The following cases are explicitly considered:

1. If $g(\delta)$ is a rational transfer function of the form (205), then the resulting matrices of the descriptor realization $(\widetilde{A} - \delta\widetilde{E}, \widetilde{B}, \widetilde{C}, \widetilde{D})$ are determined as follows:

$$\widetilde{A} - \delta\widetilde{E} = \begin{bmatrix} dA - bE - \delta(aE - cA) & dB + \delta cB \\ 0 & -I \end{bmatrix}, \quad \widetilde{B} = \begin{bmatrix} 0 \\ I \end{bmatrix}, \quad \widetilde{C} = [\, C \;\; D \,], \quad \widetilde{D} = 0.$$

   This realization is generally not minimal, even if the original realization (204) is minimal, and therefore, by default, a more compact equivalent descriptor realization, without non-dynamic modes, is determined, unless `OPTION.compact = false`. The elimination of non-dynamic modes is performed using the approach implemented in the function `gss2ss` (see equations (122)) and the resulting $\widetilde{E}$ is upper triangular. A minimal realization can be obtained by using `OPTIONS.minimal = true`. A standard state-space realization of `SYST` (if exists) can be obtained using `OPTIONS.ss = true`.

2. If $g(\delta)$ is a polynomial transfer function of the form $g(\delta) = a\delta + b$ and $E \neq I$, then the resulting matrices of the descriptor realization $(\widetilde{A} - \delta\widetilde{E}, \widetilde{B}, \widetilde{C}, \widetilde{D})$ are determined as follows:

$$\widetilde{A} - \delta\widetilde{E} = A - bE - \delta aE, \quad \widetilde{B} = B, \quad \widetilde{C} = C, \quad \widetilde{D} = D.$$

   This realization is minimal, if the original realization (204) is minimal. A standard state-space realization of `SYST` (if exists) can be obtained using `OPTIONS.ss = true`.

3. If $g(\delta)$ is a polynomial transfer function of the form $g(\delta) = a\delta + b$ and $E = I$, then the resulting matrices of the standard state-space realization $(\widetilde{A} - \delta I, \widetilde{B}, \widetilde{C}, \widetilde{D})$ with $\widetilde{E} = I$, are determined as follows:

$$\widetilde{A} = (A - bI)/a, \quad \widetilde{B} = B/a, \quad \widetilde{C} = C, \quad \widetilde{D} = D.$$

   This realization is minimal, if the original realization (204) is minimal.

Note: The function `gbilin1` (not documented) can be employed to generate the transfer function $g(\delta)$ and its inverse $g^{-1}(\lambda)$ for several commonly used bilinear transformations.

Example

Example 15. Consider the 2×2 transfer function matrix $G(s)$ of a continuous-time system

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

It is straightforward to observe that $G(s)$ is improper and has poles and zeros in the origin and at infinity. We can try to perturb $G(s)$ to simultaneously move the poles and zeros in the origin into the stable domain and make the infinite pole and zero finite by using a bilinear transformation $s \leftarrow g(s) = \frac{s+0.01}{1+0.01s}$. Here, $g(s)$ is formed as a composition of a *translation* to shift the poles and zeros (i.e., $s \leftarrow s+0.01$) and of a *reflection* and *inversion* with respect to the real axis to make infinite poles finite (i.e., $s \leftarrow \frac{1}{1+0.01s}$). To compute the descriptor realization of the proper and stable $G(g(s))$, we can use the following command sequence:

```
s = tf('s');              % define the complex variable s
G  = [s^2 s/(s+1); 0 1/s] % define the 2-by-2 improper G(s)
sys = ss(G);              % build continuous-time descriptor system realization
[p,m] = size(sys);        % get system dimensions

% pole-zero analysis
pol = gpole(sys), zer = gzero(sys)

% make all poles and zeros stable and finite using g(s) = (s+0.01)/(1+0.01*s)
g = (s+0.01)/(1+0.01*s);

% compute the transformed system
syst = gbilin(sys,g,struct('tol',1.e-7,'minimal',true));
minreal(zpk(syst),1.e-5)

% check resulting poles and zeros
pol_new = gpole(syst), zer_new = gzero(syst)

% compute the Vinnicombe's  nugap distance between models
nugap = gnugap(sys,syst)

% plot the Bode magnitude plot
bodemag(sys,syst,{0.01 100})
```

The resulting proper transfer function matrix $G\big(g(s)\big)$, with stable and finite poles and zeros, is

$$G\big(g(s)\big) = \begin{bmatrix} \dfrac{(s+0.01)^2}{(0.01s+1)^2} & 0.9901\dfrac{s+0.01}{s+1} \\ 0 & \dfrac{0.01s+1}{s+0.01} \end{bmatrix}$$

and the resulting ν-gap distance [65] between the transfer function matrices $G(s)$ (improper) and $G\big(g(s)\big)$ (proper), is $\delta_\nu\big(G(s), G\big(g(s)\big)\big) = 0.0192$ (see Section 2.13). The Bode-magnitude plots provide an alternative way to compare the frequency responses of $G(s)$ and $G(g(s))$ on a relevant frequency range. $\Diamond$

3.6 Functions for Factorizations

These functions cover the computation of several factorizations of rational matrices, such as the coprime factorizations, inner-outer factorizations, and special spectral factorizations.

3.6.1 `grcf`

Syntax

```
[SYSN,SYSM] = grcf(SYS,OPTIONS)
```

Description

`grcf` computes, for the transfer function matrix $G(\lambda)$ of a LTI descriptor state-space system, a right coprime factorization in the form

$$G(\lambda) = N(\lambda)M^{-1}(\lambda)\,, \tag{208}$$

such that $N(\lambda)$ and $M(\lambda)$ are proper transfer function matrices with poles in a specified stability region $\mathbb{C}_g \subset \mathbb{C}$.

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{209}$$

with $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	Description
`tol`	tolerance for the singular values based rank determination of E (Default: $n^2\|E\|_1$`eps`)
`tolmin`	tolerance for the singular values based controllability tests (Default: $nm\|B\|_1$`eps`)
`smarg`	stability margin which specifies the stability region $\mathbb{C}_g$ of the eigenvalues of the pole pencil as follows: in the continuous-time case, the stable eigenvalues have real parts less than or equal to `OPTIONS.smarg`, and in the discrete-time case, the stable eigenvalues have moduli less than or equal to `OPTIONS.smarg`. (Default: `-sqrt(eps)` for a continuous-time system `SYS`; `1-sqrt(eps)` for a discrete-time system `SYS`.)

`sdeg`	prescribed stability degree for the poles of the factors assigned within $\mathbb{C}_g$ (Default: `[ ]`)
`poles`	complex conjugated set of desired poles to be assigned for the factors (Default: `[ ]`)
`mindeg`	option to compute a minimum degree denominator: `true` – determine a minimum degree denominator; `false` – determine both factors with the same order (default)
`mininf`	option for the removal of simple infinite eigenvalues (non-dynamic modes) of the factors: `true` – remove simple infinite eigenvalues; `false` – keep simple infinite eigenvalues (default)

Output data

`SYSN` contains the descriptor system state-space realization of the numerator factor $N(\lambda)$ in the form

$$\begin{aligned} E_N\lambda x_N(t) &= A_N x_N(t) + B_N v(t), \\ y_N(t) &= C_N x_N(t) + D_N v(t), \end{aligned} \tag{210}$$

where the pair (A_N, E_N) is in a GRSF. The eigenvalues of $A_N - \lambda E_N$ include all eigenvalues of $A - \lambda E$ in $\mathbb{C}_g$, and, additionally, all assigned eigenvalues in accordance with the specified stability degree in `OPTIONS.sdeg` and poles in `OPTIONS.poles`. The resulting E_N is invertible if `OPTIONS.mininf = true`.

`SYSM` contains the descriptor system state-space realization of the denominator factor $M(\lambda)$ in the form

$$\begin{aligned} E_M\lambda x_M(t) &= A_M x_M(t) + B_M w(t), \\ y_M(t) &= C_M x_M(t) + D_M w(t), \end{aligned} \tag{211}$$

where the pair (A_M, E_M) is in a GRSF. If `OPTIONS.mindeg = false`, then $N(\lambda)$ and $M(\lambda)$ have realizations of the same order with $E_M = E_N$, $A_M = A_N$, and $B_M = B_N$ and the eigenvalues of $A_M - \lambda E_M$ include all eigenvalues of $A - \lambda E$ in $\mathbb{C}_g$, which are however unobservable. Additionally, the eigenvalues of $A_M - \lambda E_M$ include the assigned eigenvalues in accordance with the specified stability degree in `OPTIONS.sdeg` and poles in `OPTIONS.poles`. The resulting E_M is invertible if `OPTIONS.mininf = true`. If `OPTIONS.mindeg = true` then the eigenvalues of $A_M - \lambda E_M$ include only the assigned eigenvalues in accordance with the specified stability degree in `OPTIONS.sdeg` and poles in `OPTIONS.poles`. In this case, the resulting E_M is always invertible.

Method

For the definitions related to coprime factorizations of transfer function matrices see Section 2.8. The implemented computational methods to compute the right coprime factorizations of general rational matrices rely on a preliminary orthogonal reduction of the pole pencil $A - \lambda E$ to a special GRSF, which allows to preserve all eigenvalues of $A - \lambda E$ in $\mathbb{C}_g$ in the resulting factors. The underlying reduction is described in [58]. The function `grcf` implements the **Procedure**

GRCF of [58], which represents an extension of the recursive factorization approach of [49] to cope with infinite poles. In this procedure, all infinite poles are first assigned to finite real values.

If `OPTIONS.poles` is empty, then a stabilization oriented factorization is performed (see [58]), where the infinite poles are assigned to real values specified by `OPTIONS.sdeg` and all finite poles lying outside $\mathbb{C}_g$ are assigned to the nearest values having a stability margin specified by `OPTIONS.sdeg`. If `OPTIONS.poles` is not empty, then a pole assignment oriented factorization is performed, by assigning first all infinite poles to real values specified in `OPTIONS.poles`. If `OPTIONS.poles` does not contain a sufficient number of real values, then a part or all of infinite poles are assigned to the value specified by `OPTIONS.sdeg`. Then, all finite poles lying outside $\mathbb{C}_g$ are assigned to values specified in `OPTIONS.poles` or assigned to the values having a stability margin specified by `OPTIONS.sdeg`.

Example

Example 16. Consider the continuous-time improper TFM

$$G(s) = \begin{bmatrix} s^2 & \dfrac{s}{s+1} \\ 0 & \dfrac{1}{s} \end{bmatrix}, \tag{212}$$

which has the following set of poles: $\{-1, 0, \infty, \infty\}$. To compute a stable and proper right coprime factorization of $G(s)$, we employed the pole assignment oriented factorization, with a stability degree of -1, a desired set of poles $\{-1, -2, -3\}$, and with the option to eliminate the simple infinite eigenvalues. The resulting factors have the transfer function matrices

$$N(s) = \begin{bmatrix} -\dfrac{s^2}{(s+1)(s+2)} & \dfrac{s^2}{(s+1)(s+3)} \\ 0 & \dfrac{1}{s+3} \end{bmatrix}, \quad M(s) = \begin{bmatrix} -\dfrac{1}{(s+1)(s+2)} & 0 \\ 0 & \dfrac{s}{s+3} \end{bmatrix}.$$

The McMillan degree of $M(s)$ is three, thus the least possible one. The above factors have been computed with the following sequence of commands:

```
% Varga (2017), Example 1
s = tf('s');  % define the complex variable s
% enter G(s) and determine a minimal state-space realization
G = [s^2 s/(s+1);
    0 1/s];
sys = ss(G);
gpole(sys)    % the system is unstable and improper

% compute the right coprime factorization G(s) = N(s)*inv(M(s))
[sysn,sysm] = grcf(sys,struct('poles',[-1,-2,-3,-4],'sdeg',-1,'mininf',true));

% check the factorization G(s)*M(s) = N(s)
gnrank(sys*sysm-sysn)
```

```
% check the poles of the factors
gpole(sysm), gpole(sysn)

% check coprimeness of the factors
gzero(gir([sysn;sysm]))   % [N(s); M(s)] has no zeros
```

$\Diamond$

3.6.2 glcf

Syntax

```
[SYSN,SYSM] = glcf(SYS,OPTIONS)
```

Description

`glcf` computes, for the transfer function matrix $G(\lambda)$ of a LTI descriptor state-space system, a left coprime factorization in the form

$$G(\lambda) = M^{-1}(\lambda)N(\lambda)\,, \tag{213}$$

such that $N(\lambda)$ and $M(\lambda)$ are proper transfer function matrices with poles in a specified stability region $\mathbb{C}_g \subset \mathbb{C}$.

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{214}$$

with $x(t) \in \mathbb{R}^n$ and $y(t) \in \mathbb{R}^p$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	Description
`tol`	tolerance for the singular values based rank determination of E (Default: $n^2\|E\|_1$`eps`)
`tolmin`	tolerance for the singular values based observability tests (Default: $np\|C\|_\infty$`eps`)
`smarg`	stability margin which specifies the stability region $\mathbb{C}_g$ of the eigenvalues of the pole pencil as follows: in the continuous-time case, the stable eigenvalues have real parts less than or equal to `OPTIONS.smarg`, and in the discrete-time case, the stable eigenvalues have moduli less than or equal to `OPTIONS.smarg`. (Default: `-sqrt(eps)` for a continuous-time system `SYS`; `1-sqrt(eps)` for a discrete-time system `SYS`.)

`sdeg`	prescribed stability degree for the poles of the factors assigned within $\mathbb{C}_g$ (Default: `[ ]`)
`poles`	complex conjugated set of desired poles to be assigned for the factors (Default: `[ ]`)
`mindeg`	option to compute a minimum degree denominator: `true` – determine a minimum degree denominator; `false` – determine both factors with the same order (default)
`mininf`	option for the removal of simple infinite eigenvalues (non-dynamic modes) of the factors: `true` – remove simple infinite eigenvalues; `false` – keep simple infinite eigenvalues (default)

Output data

`SYSN` contains the descriptor system state-space realization of the numerator factor $N(\lambda)$ in the form

$$\begin{aligned} E_N\lambda x_N(t) &= A_N x_N(t) + B_N u(t), \\ y_N(t) &= C_N x_N(t) + D_N u(t), \end{aligned} \tag{215}$$

where the pair (A_N, E_N) is in a GRSF. The eigenvalues of $A_N - \lambda E_N$ include all eigenvalues of $A - \lambda E$ in $\mathbb{C}_g$, and, additionally, all assigned eigenvalues in accordance with the specified stability degree in `OPTIONS.sdeg` and poles in `OPTIONS.poles`. The resulting E_N is invertible if `OPTIONS.mininf = true`.

`SYSM` contains the descriptor system state-space realization of the denominator factor $M(\lambda)$ in the form

$$\begin{aligned} E_M\lambda x_M(t) &= A_M x_M(t) + B_M w(t), \\ y_M(t) &= C_M x_M(t) + D_M w(t), \end{aligned} \tag{216}$$

where the pair (A_M, E_M) is in a GRSF. If `OPTIONS.mindeg = false`, then $N(\lambda)$ and $M(\lambda)$ have realizations of the same order with $E_M = E_N$, $A_M = A_N$, and $C_M = C_N$ and the eigenvalues of $A_M - \lambda E_M$ include all eigenvalues of $A - \lambda E$ in $\mathbb{C}_g$, which are however uncontrollable. Additionally, the eigenvalues of $A_M - \lambda E_M$ include the assigned eigenvalues in accordance with the specified stability degree in `OPTIONS.sdeg` and poles in `OPTIONS.poles`. The resulting E_M is invertible if `OPTIONS.mininf = true`. If `OPTIONS.mindeg = true` then the eigenvalues of $A_M - \lambda E_M$ include only the assigned eigenvalues in accordance with the specified stability degree in `OPTIONS.sdeg` and poles in `OPTIONS.poles`. In this case, the resulting E_M is always invertible.

Method

For the definitions related to coprime factorizations of transfer function matrices see Section 2.8. To compute the left coprime factorization (213), the function `glcf` calls `grcf` to compute the right coprime factorization of $G^T(\lambda)$ in the form

$$G^T(\lambda) = \widetilde{N}(\lambda)\widetilde{M}^{-1}(\lambda)$$

and obtain the factors as $N(\lambda) = \widetilde{N}^T(\lambda)$ and $M(\lambda) = \widetilde{M}^T(\lambda)$. The function `grcf` implements the **Procedure GRCF** of [58], which represents an extension of the recursive factorization approach of [49] to cope with infinite poles.

If `OPTIONS.poles` is empty, then a stabilization oriented factorization is performed (see [58]), where the infinite poles are assigned to real values specified by `OPTIONS.sdeg` and all finite poles lying outside $\mathbb{C}_g$ are assigned to the nearest values having a stability margin specified by `OPTIONS.sdeg`. If `OPTIONS.poles` is not empty, then a pole assignment oriented factorization is performed, by assigning first all infinite poles to real values specified in `OPTIONS.poles`. If `OPTIONS.poles` does not contain a sufficient number of real values, then a part or all of infinite poles are assigned to the value specified by `OPTIONS.sdeg`. Then, all finite poles lying outside $\mathbb{C}_g$ are assigned to values specified in `OPTIONS.poles` or assigned to the values having a stability margin specified by `OPTIONS.sdeg`.

Example

Example 17. Consider the continuous-time improper TFM

$$G(s) = \begin{bmatrix} s^2 & \dfrac{s}{s+1} \\ 0 & \dfrac{1}{s} \end{bmatrix}, \tag{217}$$

which has the following set of poles: $\{-1, 0, \infty, \infty\}$. To compute a stable and proper left coprime factorization of $G(s)$, we employed the pole assignment oriented factorization, with a stability degree of -1, a desired set of poles $\{-1, -2, -3\}$, and with the option to eliminate the simple infinite eigenvalues. The resulting factors have the transfer function matrices

$$N(s) = \begin{bmatrix} -\dfrac{s^2}{(s+1)(s+2)} & -\dfrac{s}{(s+1)^2(s+2)} \\ 0 & \dfrac{1}{s+3} \end{bmatrix}, \quad M(s) = \begin{bmatrix} -\dfrac{1}{(s+1)(s+2)} & 0 \\ 0 & \dfrac{s}{s+3} \end{bmatrix}.$$

The McMillan degree of $M(s)$ is three, thus the least possible one. The above factors have been computed with the following sequence of commands:

```
% Varga (2017), Example 1
s = tf('s');  % define the complex variable s
% enter G(s) and determine a minimal state-space realization
G = [s^2 s/(s+1);
    0 1/s];
sys = ss(G);
gpole(sys)    % the system is unstable and improper

% compute the left coprime factorization G(s) = inv(M(s))*N(s)
[sysn,sysm] = glcf(sys,struct('poles',[-1,-2,-3,-4],'sdeg',-1,'mininf',true));

% check the factorization M(s)*G(s) = N(s)
gnrank(sysm*sys-sysn,1.e-7,rand)
```

```
% check the poles of the factors
gpole(sysm), gpole(sysn)

% check coprimeness of the factors
gzero(gir([sysn sysm]))  % [N(s) M(s)] has no zeros
```

$\Diamond$

3.6.3 grcfid

Syntax

```
[SYSN,SYSM] = grcfid(SYS,OPTIONS)
```

Description

`grcfid` computes, for the transfer function matrix $G(\lambda)$ of a LTI descriptor state-space system, a right coprime factorization with inner denominator in the form

$$G(\lambda) = N(\lambda)M^{-1}(\lambda)\,, \tag{218}$$

such that $N(\lambda)$ and $M(\lambda)$ are proper and stable transfer function matrices, and $M(\lambda)$ is inner.

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{219}$$

with $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$. $G(\lambda)$ must not have poles in $\partial\mathbb{C}_s$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	Description
`tol`	tolerance for the singular values based rank determination of E (Default: $n^2\|E\|_1$`eps`)
`tolmin`	tolerance for the singular values based controllability tests (Default: $nm\|B\|_1$`eps`)
`mindeg`	option to compute a minimum degree denominator: `true` – determine a minimum degree denominator; `false` – determine both factors with the same order (default)
`mininf`	option for the removal of simple infinite eigenvalues (non-dynamic modes) of the factors: `true` – remove simple infinite eigenvalues; `false` – keep simple infinite eigenvalues (default)

Output data

`SYSN` contains the descriptor system state-space realization of the numerator factor $N(\lambda)$ in the form

$$\begin{aligned} E_N\lambda x_N(t) &= A_N x_N(t) + B_N v(t), \\ y_N(t) &= C_N x_N(t) + D_N v(t), \end{aligned} \tag{220}$$

where the pair (A_N, E_N) is in a GRSF. The eigenvalues of $A_N - \lambda E_N$ include all stable eigenvalues of $A - \lambda E$ (i.e., eigenvalues located in $\mathbb{C}_s$). Additionally, to each unstable eigenvalue of $A - \lambda E$ corresponds a stable eigenvalue of $A_N - \lambda E_N$ located in a symmetric location with respect to the imaginary axis, in the continuous-time case, or with respect to the unit circle centered in the origin, in the discrete-time case. The resulting E_N is invertible if `OPTIONS.mininf = true`.

`SYSM` contains the descriptor system state-space realization of the inner denominator factor $M(\lambda)$ in the form

$$\begin{aligned} E_M\lambda x_M(t) &= A_M x_M(t) + B_M w(t), \\ y_M(t) &= C_M x_M(t) + D_M w(t), \end{aligned} \tag{221}$$

where the pair (A_M, E_M) is in a GRSF. The resulting E_M is invertible if `OPTIONS.mininf = true`. If `OPTIONS.mindeg = false`, then $N(\lambda)$ and $M(\lambda)$ have realizations of the same order with $E_M = E_N$, $A_M = A_N$, and $B_M = B_N$ and the eigenvalues of $A_M - \lambda E_M$ include all stable eigenvalues of $A - \lambda E$, which are however unobservable. Additionally, to each unstable eigenvalue of $\lambda_u \in \Lambda(A - \lambda E)$ corresponds a stable eigenvalue of $A_M - \lambda E_M$ located in a symmetric location with respect to the boundary of the appropriate stability domain (i.e., $-\bar{\lambda}_u \in \Lambda(A_M - \lambda E_M)$, in the continuous-time case, or $1/\lambda_u \in \Lambda(A_M - \lambda E_M)$ in the discrete-time case). If `OPTIONS.mindeg = true`, only the latter eigenvalues are present and the resulting E_M is always invertible.

Method

For the definitions related to coprime factorizations of transfer function matrices see Section 2.8. The implemented computational methods to compute the right coprime factorizations with inner denominators of transfer function matrices rely on a preliminary orthogonal reduction of the pole pencil $A - \lambda E$ to a special GRSF, which allows to isolate all eigenvalues of $A - \lambda E$ lying outside of the stability domain $\mathbb{C}_s$. The underlying reduction is described in [58]. The function `grcfid` implements the **Procedure GRCFID** of [58], which represents an extension of the corresponding recursive factorization approach of [49] to cope with infinite poles in the discrete-time case. In this procedure, all infinite poles of a discrete-time system are reflected into poles in the origin of the factors.

Example

Example 18. Consider the discrete-time improper TFM

$$G(z) = \begin{bmatrix} z^2 & \dfrac{z}{z-2} \\ 0 & \dfrac{1}{z} \end{bmatrix}, \tag{222}$$

which has the following set of poles: $\{2, 0, \infty, \infty\}$ and therefore, the right coprime factorization with inner denominator exists. With the option to eliminate the simple infinite eigenvalues, the function `grcfid` computes the following factors having the transfer function matrices

$$N(z) = \begin{bmatrix} 1 & \dfrac{1}{2z-1} \\ 0 & \dfrac{z-2}{z(2z-1)} \end{bmatrix}, \quad M(z) = \begin{bmatrix} \dfrac{1}{z^2} & 0 \\ 0 & \dfrac{z-2}{2z-1} \end{bmatrix}.$$

The McMillan degree of $M(z)$ is three, thus the least possible one. Interestingly, the McMillan degree of $N(z)$ is only two, because two unobservable eigenvalues in 0 have been removed. These eigenvalues are the zeros of the (improper) all-pass factor $\mathrm{diag}(z^2, 1)$ with two infinite poles, which is contained in $G(z)$.

The above factors have been computed with the following sequence of commands:

```
% Varga (2017), Example 2
z = tf('z');  % define the complex variable z
% enter G(z) and determine a minimal state-space realization
G = [z^2 z/(z-2);
    0 1/z];
sys = ss(G);
gpole(sys)    % the system is unstable and improper

% compute the right coprime factorization G(z) = N(z)*inv(M(z)),
% with inner denominator M(z)
[sysn,sysm] = grcfid(sys,struct('mininf',true));

% check the factorization G(z)*M(z) = N(z)
gnrank(sys*sysm-sysn,1.e-7,rand)

% check the innerness of M(z): M'(z)*M(z) = I
gnrank(sysm'*sysm-eye(2),1.e-7,rand)

% check the poles of the factors
gpole(sysm), gpole(sysn)

% check coprimeness of the factors
gzero(gir([sysn;sysm]))   % [N(z); M(z)] has no zeros
```

$\Diamond$

3.6.4 `glcfid`

Syntax

```
[SYSN,SYSM] = glcfid(SYS,OPTIONS)
```

Description

`glcfid` computes, for the transfer function matrix $G(\lambda)$ of a LTI descriptor state-space system, a left coprime factorization with inner denominator in the form

$$G(\lambda) = M^{-1}(\lambda)N(\lambda)\,, \tag{223}$$

such that $N(\lambda)$ and $M(\lambda)$ are proper and stable transfer function matrices, and $M(\lambda)$ is inner.

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{224}$$

with $x(t) \in \mathbb{R}^n$ and $y(t) \in \mathbb{R}^p$. $G(\lambda)$ must not have poles in $\partial\mathbb{C}_s$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	**Description**
`tol`	tolerance for the singular values based rank determination of E (Default: $n^2\|E\|_1$`eps`)
`tolmin`	tolerance for the singular values based observability tests (Default: $np\|C\|_\infty$`eps`)
`mindeg`	option to compute a minimum degree denominator: `true` – determine a minimum degree denominator; `false` – determine both factors with the same order (default)
`mininf`	option for the removal of simple infinite eigenvalues (non-dynamic modes) of the factors: `true` – remove simple infinite eigenvalues; `false` – keep simple infinite eigenvalues (default)

Output data

`SYSN` contains the descriptor system state-space realization of the numerator factor $N(\lambda)$ in the form

$$\begin{aligned} E_N\lambda x_N(t) &= A_N x_N(t) + B_N u(t), \\ y_N(t) &= C_N x_N(t) + D_N u(t), \end{aligned} \tag{225}$$

where the pair (A_N, E_N) is in a GRSF. The eigenvalues of $A_N - \lambda E_N$ include all stable eigenvalues of $A - \lambda E$ (i.e., eigenvalues located in $\mathbb{C}_s$). Additionally, to each unstable eigenvalue of $A - \lambda E$ corresponds a stable eigenvalue of $A_N - \lambda E_N$ located in a symmetric location with respect to the imaginary axis, in the continuous-time case, or with respect to the unit circle centered in the origin, in the discrete-time case. The resulting E_N is invertible if `OPTIONS.mininf = true`.

`SYSM` contains the descriptor system state-space realization of the inner denominator factor

$M(\lambda)$ in the form

$$\begin{aligned} E_M \lambda x_M(t) &= A_M x_M(t) + B_M w(t), \\ y_M(t) &= C_M x_M(t) + D_M w(t), \end{aligned} \tag{226}$$

where the pair (A_M, E_M) is in a GRSF. The resulting E_M is invertible if `OPTIONS.mininf = true`. If `OPTIONS.mindeg = false`, then $N(\lambda)$ and $M(\lambda)$ have realizations of the same order with $E_M = E_N$, $A_M = A_N$, and $C_M = C_N$ and the eigenvalues of $A_M - \lambda E_M$ include all stable eigenvalues of $A - \lambda E$, which are however unobservable. Additionally, to each unstable eigenvalue of $\lambda_u \in \Lambda(A - \lambda E)$ corresponds a stable eigenvalue of $A_M - \lambda E_M$ located in a symmetric location with respect to the boundary of the appropriate stability domain (i.e., $-\bar{\lambda}_u \in \Lambda(A_M - \lambda E_M)$, in the continuous-time case, or $1/\lambda_u \in \Lambda(A_M - \lambda E_M)$ in the discrete-time case). If `OPTIONS.mindeg = true`, only the latter eigenvalues are present and the resulting E_M is always invertible.

Method

For the definitions related to coprime factorizations of transfer function matrices see Section 2.8. To compute the left coprime factorization (223), the function `glcfid` calls `grcfid` to compute the right coprime factorization with inner denominator of $G^T(\lambda)$ in the form

$$G^T(\lambda) = \widetilde{N}(\lambda)\widetilde{M}^{-1}(\lambda)$$

and obtain the factors as $N(\lambda) = \widetilde{N}^T(\lambda)$ and $M(\lambda) = \widetilde{M}^T(\lambda)$. The function `grcfid` implements the **Procedure GRCFID** of [58], which represents an extension of the corresponding recursive factorization approach of [49] to cope with infinite poles in the discrete-time case.

Example

Example 19. Consider the discrete-time improper TFM

$$G(z) = \begin{bmatrix} z^2 & \dfrac{z}{z-2} \\ 0 & \dfrac{1}{z} \end{bmatrix}, \tag{227}$$

which has the following set of poles: $\{2, 0, \infty, \infty\}$ and therefore, the left coprime factorization with inner denominator exists. With the option to eliminate the simple infinite eigenvalues, the function `glcfid` computes the following factors having the transfer function matrices

$$N(z) = \begin{bmatrix} \dfrac{z-2}{2z-1} & \dfrac{1}{z(2z-1)} \\ 0 & \dfrac{1}{z} \end{bmatrix}, \quad M(z) = \begin{bmatrix} \dfrac{z-2}{z^2(2z-1)} & 0 \\ 0 & 1 \end{bmatrix}.$$

The McMillan degree of $M(z)$ is three, thus the least possible one. Interestingly, the McMillan degree of $N(z)$ is only two, because two unobservable eigenvalues in 0 have been removed. These eigenvalues are the zeros of the (improper) all-pass factor $\mathrm{diag}(z^2, 1)$ with two infinite poles, which is contained in $G(z)$.

The above factors have been computed with the following sequence of commands:

```
% Varga (2017), Example 2
z = tf('z');  % define the complex variable z
% enter G(z) and determine a minimal state-space realization
G = [z^2 z/(z-2);
    0 1/z];
sys = ss(G);
gpole(sys)    % the system is unstable and improper

% compute the right coprime factorization G(z) = inv(M(z))*N(z),
% with inner denominator M(z)
[sysn,sysm] = glcfid(sys,struct('mininf',true));

% check the factorization M(z)*G(z) = N(z)
gnrank(sysm*sys-sysn,1.e-7,rand)

% check the innerness of M(z): M'(z)*M(z) = I
gnrank(sysm'*sysm-eye(2),1.e-7,rand)

% check the poles of the factors
gpole(sysm), gpole(sysn)

% check coprimeness of the factors
gzero(gir([sysn sysm]))   % [N(z) M(z)] has no zeros
```

$\Diamond$

3.6.5 gnrcf

Syntax

```
[SYSN,SYSM] = gnrcf(SYS,OPTIONS)
```

Description

`gnrcf` computes, for the transfer function matrix $G(\lambda)$ of a LTI descriptor state-space system, a normalized right coprime factorization in the form

$$G(\lambda) = N(\lambda)M^{-1}(\lambda)\,, \tag{228}$$

such that $N(\lambda)$ and $M(\lambda)$ are proper and stable transfer function matrices and $\left[\begin{smallmatrix} N(\lambda) \\ M(\lambda) \end{smallmatrix}\right]$ is inner.

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{229}$$

with $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	**Description**
`tol`	tolerance for rank determinations (Default: internally computed)
`ss`	option to compute standard state-space realizations of the factors: `true` – determine standard state-space realizations; `false` – determine descriptor system realizations (default)
`balance`	balancing option for the Riccati equation solvers (see functions `care` and `dare` of the Control System Toolbox): `true` – apply balancing (default); `false` – disable balancing.

Output data

`SYSN` contains the descriptor system state-space realization of the numerator factor $N(\lambda)$ in the form

$$\begin{aligned} E_N \lambda x_N(t) &= A_N x_N(t) + B_N v(t), \\ y_N(t) &= C_N x_N(t) + D_N v(t). \end{aligned} \tag{230}$$

The resulting $E_N = I_n$ if `OPTIONS.ss = true`.

`SYSM` contains the descriptor system state-space realization of the denominator factor $M(\lambda)$ in the form

$$\begin{aligned} E_N \lambda x_M(t) &= A_N x_M(t) + B_N w(t), \\ y_M(t) &= C_M x_M(t) + D_M w(t). \end{aligned} \tag{231}$$

The resulting $E_N = I_n$ if `OPTIONS.ss = true`.

Method

The factors $N(\lambda)$ and $M(\lambda)$ are computed from a minimal inner basis $R(\lambda)$ of the range space of $\left[\begin{smallmatrix} G(\lambda) \\ I_m \end{smallmatrix}\right]$ satisfying

$$\begin{bmatrix} G(\lambda) \\ I_m \end{bmatrix} = R(\lambda) X(\lambda),$$

with

$$R(\lambda) = \begin{bmatrix} N(\lambda) \\ M(\lambda) \end{bmatrix}, \qquad X(\lambda) = M^{-1}(\lambda).$$

For the computation of an inner range space of a rational matrix the method discussed in [57] is employed.

Example 20. For the polynomial transfer function matrix $G(s)$ considered in the **Example 3** of [32] with

$$G(s) = \begin{bmatrix} s^2 + s + 1 & 4\,s^2 + 3\,s + 2 & 2\,s^2 - 2 \\ s & 4\,s - 1 & 2\,s - 2 \\ s^2 & 4\,s^2 - s & 2\,s^2 - 2\,s \end{bmatrix},$$

the following MATLAB code can be used to compute a normalized right coprime factorization of $G(s)$:

```
% Oara and Varga (2000), Example 3
s = tf('s');  % define the complex variable s
% enter G(s) and determine a minimal state-space realization
G = [s^2+s+1 4*s^2+3*s+2 2*s^2-2;
    s 4*s-1 2*s-2;
    s^2 4*s^2-s 2*s^2-2*s];
sys = gir(ss(G));

% compute the normalized right coprime factorization
[N,M] = gnrcf(sys,1.e-7);

% check the coprime factorization
gnrank(N*inv(M)-sys,1.e-7,rand)                    % N*inv(M) = G

% checking the innerness of [N;M]
gnrank([N;M]'*[N;M]-eye(size(sys,2)),1.e-7,rand)   % [N;M]'*[N;M] = I
gpole(N), gpole(M)                                 % N and M are stable
```

◊

3.6.6 gnlcf

Syntax

```
[SYSN,SYSM] = gnlcf(SYS,OPTIONS)
```

Description

`gnlcf` computes, for the transfer function matrix $G(\lambda)$ of a LTI descriptor state-space system, a normalized left coprime factorization in the form

$$G(\lambda) = M^{-1}(\lambda)N(\lambda)\,, \tag{232}$$

such that $N(\lambda)$ and $M(\lambda)$ are proper and stable transfer function matrices and $[\,N(\lambda)\ M(\lambda)\,]$ is coinner.

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{233}$$

with $x(t) \in \mathbb{R}^n$ and $y(t) \in \mathbb{R}^p$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

OPTIONS fields	Description
tol	tolerance for rank determinations (Default: internally computed)
ss	option to compute standard state-space realizations of the factors: true – determine standard state-space realizations; false – determine descriptor system realizations (default)
balance	balancing option for the Riccati equation solvers (see functions care and dare of the Control System Toolbox): true – apply balancing (default); false – disable balancing.

Output data

SYSN contains the descriptor system state-space realization of the numerator factor $N(\lambda)$ in the form

$$\begin{aligned} E_N \lambda x_N(t) &= A_N x_N(t) + B_N u(t), \\ y_N(t) &= C_N x_N(t) + D_N u(t). \end{aligned} \tag{234}$$

The resulting $E_N = I_n$ if OPTIONS.ss = true.

SYSM contains the descriptor system state-space realization of the denominator factor $M(\lambda)$ in the form

$$\begin{aligned} E_N \lambda x_M(t) &= A_N x_M(t) + B_M w(t), \\ y_M(t) &= C_N x_M(t) + D_M w(t), \end{aligned} \tag{235}$$

The resulting $E_N = I_n$ if OPTIONS.ss = true.

Method

The factors $N(\lambda)$ and $M(\lambda)$ are computed from a minimal coinner basis $R(\lambda)$ of the co-image space of $[\, G(\lambda)\ I_p \,]$ satisfying

$$[\, G(\lambda)\ I_p \,] = X(\lambda) R(\lambda),$$

with

$$R(\lambda) = [\, N(\lambda)\ M(\lambda) \,], \qquad X(\lambda) = M^{-1}(\lambda).$$

For the computation of a coinner basis of the co-image space of a rational matrix, the inner range space basis of the transposed rational matrix is computed, for which the computational method discussed in [57] is employed.

Example 21. For the polynomial transfer function matrix $G(z)$ considered in the **Example 3** of [32], obtained by replacing the complex Laplace transform variable s with the complex Z-transform variable z,

$$G(z) = \begin{bmatrix} z^2 + z + 1 & 4\,z^2 + 3\,z + 2 & 2\,z^2 - 2 \\ z & 4\,z - 1 & 2\,z - 2 \\ z^2 & 4\,z^2 - z & 2\,z^2 - 2\,z \end{bmatrix},$$

the following MATLAB code can be used to compute a normalized right coprime factorization of $G(z)$:

```
% Oara and Varga (2000), Example 3
z = tf('z');  % define the complex variable z
% enter G(s) and determine a minimal state-space realization
G = [z^2+z+1 4*z^2+3*z+2 2*z^2-2;
    z 4*z-1 2*z-2;
    z^2 4*z^2-z 2*z^2-2*z];
sys = gir(ss(G));

% compute the normalized left coprime factorization
[N,M] = gnlcf(sys,struct('tol',1.e-7));

% check the coprime factorization
gnrank(inv(M)*N-sys,1.e-7,rand)                    % inv(M)*N = G

% checking the coinnerness of [N M]
gnrank([N M]*[N M]'-eye(size(sys,1)),1.e-7,rand) % [N M]*[N M]' = I
gpole(N), gpole(M)                                 % N and M are stable
```

$\Diamond$

3.6.7 giofac

Syntax

```
[SYSI,SYSO,INFO] = giofac(SYS,OPTIONS)
```

Description

`giofac` computes, for the transfer function matrix $G(\lambda)$ of a LTI descriptor state-space system, the extended inner–quasi-outer or the extended QR-like factorization in the form

$$G(\lambda) = G_i(\lambda) \begin{bmatrix} G_o(\lambda) \\ 0 \end{bmatrix}, \tag{236}$$

where $G_i(\lambda)$ is square and inner, and $G_o(\lambda)$ is quasi-outer or full row rank, respectively.

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{237}$$

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	Description
`tol`	relative tolerance for rank computations and observability tests (Default: internally computed)

`offset`	stability boundary offset β, to be used to assess the finite zeros which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite zeros having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite zeros having moduli in the interval $[1-\beta, 1+\beta]$ (Default: $\beta = 1.4901 \cdot 10^{-08}$).
`minphase`	option to compute a minimum-phase quasi-outer factor: `true` – compute a minimum phase quasi-outer factor, with all zeros stable, excepting possibly zeros on the boundary of the appropriate stability domain (default); `false` – compute a full row rank factor, which includes all zeros of $G(\lambda)$.
`balance`	balancing option for the Riccati equation solvers (see functions `care` and `dare` of the Control System Toolbox): `true` – apply balancing (default); `false` – disable balancing.

Output data

`SYSI` contains the descriptor system state-space realization of the square inner transfer function matrix $G_i(\lambda)$ in the form

$$\begin{aligned} E_i\lambda x_i(t) &= A_i x_i(t) + B_i v(t), \\ y_i(t) &= C_i x_i(t) + D_i v(t), \end{aligned} \tag{238}$$

where E_i is invertible and $\Lambda(A_i - \lambda E_i) \subset \mathbb{C}_s$. The realization of the inner factor is a standard system with $E_i = I$ if the original system (237) is a standard system with $E = I$. If `OPTIONS.minphase = false`, then $G_i(\lambda)$ has the least possible McMillan degree.

`SYSO` contains the descriptor system state-space realization of the transfer function matrix $G_o(\lambda)$ in the form

$$\begin{aligned} E\lambda x_o(t) &= A x_o(t) + B u(t), \\ y_o(t) &= C_o x_o(t) + D_o u(t), \end{aligned} \tag{239}$$

where the dimension r of $y_o(t)$ is the normal rank of $G(\lambda)$. If `OPTIONS.minphase = true`, then $G_o(\lambda)$ is quasi-outer and all zeros of $G_o(\lambda)$ lie in $\overline{\mathbb{C}}_s$. If `OPTIONS.minphase = false`, then $G_o(\lambda)$ is full row rank and contains all zeros of $G(\lambda)$.

`INFO` is a MATLAB structure containing additional information, as follows:

`INFO` fields	Description
`nrank`	normal rank of the transfer function matrix $G(\lambda)$;
`nfuz`	number of finite unstable zeros of `SYSO`; these are the finite zeros of `SYS` lying on the boundary of the stability region $\partial\mathbb{C}_s$ within the offset specified by `OPTION.offset`;
`niuz`	number of infinite unstable zeros of `SYSO`; these are the infinite zeros of `SYS`, in the continuous-time case, and 0 in the discrete-time case;

`ricrez`	diagnosis flag, as provided provided by the generalized Riccati equation solvers `care` and `dare`; if non-negative, this value represents the Frobenius norm of relative residual of the Riccati equation, while a negative value indicates failure of solving the Riccati equation.

Method

For the definitions related to inner-outer and QR-like factorizations of transfer function matrices see Section 2.9. Assume that the transfer function matrix $G(\lambda)$ has normal rank r and the inner factor $G_i(\lambda)$ in (236) is partitioned as $G_i(\lambda) = [\, G_{i,1}(\lambda)\ G_{i,2}(\lambda)\,]$, where $G_{i,1}(\lambda)$ has r columns and is inner. Then $G_{i,2}(\lambda)$ represents the complementary inner factor $G_{i,1}^{\perp}(\lambda)$, which is the inner orthogonal complement of $G_{i,1}(\lambda)$. Thus, the full rank inner–quasi-outer or QR-like factorization of $G(\lambda)$ has the form

$$G(\lambda) = G_{i,1}(\lambda)G_o(\lambda)\,. \tag{240}$$

If `OPTIONS.minphase = true`, the resulting factor $G_o(\lambda)$ has full row rank r and is minimum phase, excepting possibly zeros in $\partial\mathbb{C}_s$, the boundary of the appropriate stability domain. If `OPTIONS.minphase = false`, $G_o(\lambda)$ has full row rank r and contains all zeros of $G(\lambda)$. In this case, the resulting inner factor has the least possible McMillan degree. If $G(\lambda)$ contains a so-called free inner factor, then the resulting realization (239) of $G_o(\lambda)$ is unobservable. The unobservable eigenvalues of the pencil $\left[\begin{smallmatrix} A-\lambda E \\ C_o \end{smallmatrix}\right]$ are precisely the (stable) poles of the free inner factor and can be readily eliminated (e.g., by using the function `gir`).

The implemented computational methods to determine the inner–quasi-outer or QR-like factorizations of general rational matrices rely on a preliminary orthogonal reduction of the system matrix pencil

$$S(\lambda) = \left[\begin{array}{cc} A-\lambda E & B \\ C & D \end{array}\right]$$

to a special Kronecker-like form (see (140)), which allows to reduce the original computational problem to a standard inner-outer factorization problem for a system with full column rank transfer function matrix and without zeros in $\partial\mathbb{C}_s$. The underlying reduction is described in [32] and involves the use of the mex-function `sl_klf` to compute the appropriate Kronecker-like form. For the computation of the inner and outer factors for the reduced problem, extensions of the standard factorization methods of [69] are used. These methods involve the solution of appropriate (continuous- or discrete-time) generalized algebraic Riccati equations. The overall factorization procedures are described in [32] for continuous-time systems and in [29] for discrete-time systems. The formulas for the determination of the complementary inner factors have been derived extending the results of [69]. The function `giofac` employs the mex-function `sl_gminr` to obtain the inner factor with a standard state-space realization with $E_i = I$.

Examples

Example 22. This is Example 1 from [32] of the transfer function matrix of a continuous-time proper system:

$$G(s) = \begin{bmatrix} \dfrac{s-1}{s+2} & \dfrac{s}{s+2} & \dfrac{1}{s+2} \\ 0 & \dfrac{s-2}{(s+1)^2} & \dfrac{s-2}{(s+1)^2} \\ \dfrac{s-1}{s+2} & \dfrac{s^2+2\,s-2}{(s+1)\,(s+2)} & \dfrac{2\,s-1}{(s+1)\,(s+2)} \end{bmatrix}. \tag{241}$$

$G(s)$ has zeros at $\{1, 2, \infty\}$, poles at $\{-1, -1, -2, -2\}$, and normal rank $r = 2$. The extended inner–quasi-outer factorization of $G(s)$ can be computed with the following command sequence:

```
% Oara and Varga (2000), Example 1
s = tf('s');  % define the complex variable s
% enter G(s) and determine a minimal state-space realization
G = [(s-1)/(s+2) s/(s+2) 1/(s+2);
    0 (s-2)/(s+1)^2 (s-2)/(s+1)^2;
    (s-1)/(s+2) (s^2+2*s-2)/(s+1)/(s+2) (2*s-1)/(s+1)/(s+2)];
sys = minreal(ss(G));
gpole(sys)    % the system is stable
gzero(sys)    % the system has 2 unstable zeros and an infinite zero
nrank(sys)    % the normal rank of G(s) is 2

% compute the extended inner-quasi-outer factorization G(s) = Gi(s)*[Go(s);0]
[sysi,syso,info] = giofac(sys,struct('tol',1.e-7));  % use tolerance 1.e-7

% check the factorization
nr = size(syso,1);                          %  nr = 2 is the normal rank of G(s)
gnrank(sysi(:,1:nr)*syso-sys,1.e-7,rand)    %  Gi(:,1:nr)(s)*Go(s) = G(s)

% checking the innerness of Gi(s)
gnrank(sysi'*sysi-eye(3),1.e-7,rand)        %  Gi'(s)*Gi(s) = I

syso = gir(syso);                 % a free inner factor is present in G(s)
gzero(syso)                       % Go(s) has no zeros in open right-half plane
info.niuz                         % Go(s) has an infinite zero
info.nfuz                         % Go(s) has no zeros on the imaginary axis
```

♢

3.6.8 goifac

Syntax

```
[SYSI,SYSO,INFO] = goifac(SYS,OPTIONS)
```

Description

`goifac` computes, for the transfer function matrix $G(\lambda)$ of a LTI descriptor state-space system, the extended quasi-co-outer–inner or the extended RQ-like factorization in the form

$$G(\lambda) = [\, G_o(\lambda) \; 0 \,] G_i(\lambda) \,, \tag{242}$$

where $G_i(\lambda)$ is square and inner, and $G_o(\lambda)$ is quasi-co-outer or full column rank, respectively.

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{243}$$

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	Description
`tol`	relative tolerance for rank computations and observability tests (Default: internally computed)
`offset`	stability boundary offset β, to be used to assess the finite zeros which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite zeros having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite zeros having moduli in the interval $[1-\beta, 1+\beta]$ (Default: $\beta = 1.4901 \cdot 10^{-08}$).
`minphase`	option to compute a minimum-phase quasi-co-outer factor: `true` – compute a minimum phase quasi-co-outer factor, with all zeros stable, excepting possibly zeros on the boundary of the appropriate stability domain (default); `false` – compute a full row rank factor, which includes all zeros of $G(\lambda)$.
`balance`	balancing option for the Riccati equation solvers (see functions `care` and `dare` of the Control System Toolbox): `true` – apply balancing (default); `false` – disable balancing.

Output data

`SYSI` contains the descriptor system state-space realization of the square inner transfer function matrix $G_i(\lambda)$ in the form

$$\begin{aligned} E_i\lambda x_i(t) &= A_i x_i(t) + B_i u(t), \\ y_i(t) &= C_i x_i(t) + D_i u(t), \end{aligned} \tag{244}$$

where E_i is invertible and $\Lambda(A_i - \lambda E_i) \subset \mathbb{C}_s$. The realization of the inner factor is a standard system with $E_i = I$ if the original system (243) is a standard system with $E = I$. If `OPTIONS.minphase = false`, then $G_i(\lambda)$ has the least possible McMillan degree.

`SYSO` contains the descriptor system state-space realization of the transfer function matrix $G_o(\lambda)$ in the form

$$\begin{aligned} E\lambda x_o(t) &= Ax_o(t) + B_o v(t), \\ y_o(t) &= Cx_o(t) + D_o v(t), \end{aligned} \tag{245}$$

where the dimension r of $v(t)$ is the normal rank of $G(\lambda)$. If `OPTIONS.minphase = true`, then $G_o(\lambda)$ is quasi-co-outer and all zeros of $G_o(\lambda)$ lie in $\overline{\mathbb{C}}_s$. If `OPTIONS.minphase = false`, then $G_o(\lambda)$ is full column rank and contains all zeros of $G(\lambda)$.

`INFO` is a MATLAB structure containing additional information, as follows:

`INFO` fields	Description
`nrank`	normal rank of the transfer function matrix $G(\lambda)$;
`nfuz`	number of finite unstable zeros of `SYSO`; these are the finite zeros of `SYS` lying on the boundary of the stability region $\partial\mathbb{C}_s$ within the offset specified by `OPTION.offset`;
`niuz`	number of infinite unstable zeros of `SYSO`; these are the infinite zeros of `SYS`, in the continuous-time case, and 0 in the discrete-time case;
`ricrez`	diagnosis flag, as provided provided by the generalized Riccati equation solvers `care` and `dare`; if non-negative, this value represents the Frobenius norm of relative residual of the Riccati equation, while a negative value indicates failure of solving the Riccati equation.

Method

For the definitions related to co-outer–coinner or RQ-like factorizations of transfer function matrices see Section 2.9. Assume that the transfer function matrix $G(\lambda)$ has normal rank r and the inner factor $G_i(\lambda)$ in (242) is partitioned as $G_i(\lambda) = \begin{bmatrix} G_{i,1}(\lambda) \\ G_{i,2}(\lambda) \end{bmatrix}$, where $G_{i,1}(\lambda)$ has r rows and is coinner. Then $G_{i,2}(\lambda)$ represents the complementary coinner factor $G_{i,1}^{\perp}(\lambda)$, which is the coinner orthogonal complement of $G_{i,1}(\lambda)$. Thus, the full rank quasi-co-outer–coinner or full rank RQ-like factorization of $G(\lambda)$ has the form

$$G(\lambda) = G_o(\lambda) G_{i,1}(\lambda)\,. \tag{246}$$

If `OPTIONS.minphase = true`, the resulting factor $G_o(\lambda)$ has full row rank r and is minimum phase, excepting possibly zeros in $\partial\mathbb{C}_s$, the boundary of the appropriate stability domain. If `OPTIONS.minphase = false`, $G_o(\lambda)$ has full row rank r and contains all zeros of $G(\lambda)$. In this case, the resulting inner factor has the least possible McMillan degree. If $G(\lambda)$ contains a so-called free inner factor, then the resulting realization (245) of $G_o(\lambda)$ is uncontrollable. The uncontrollable eigenvalues of the pencil $[\, A - \lambda E \;\; B_o \,]$ are precisely the (stable) poles of the free inner factor and can be readily eliminated (e.g., by using the function `gir`).

To compute the extended quasi-co-outer–inner or the extended RQ-like factorization (242), the function `goifac` calls `giofac` to compute the extended inner–quasi-outer or extended QR-like factorization of $G^T(\lambda)$ in the form

$$G^T(\lambda) = \widetilde{G}_i(\lambda) \begin{bmatrix} \widetilde{G}_o(\lambda) \\ 0 \end{bmatrix}$$

and obtain the inner and quasi-co-outer/full column rank factors as $G_i(\lambda) = \widetilde{G}_i^T(\lambda)$ and $G_o(\lambda) = \widetilde{G}_o^T(\lambda)$, respectively. The factorization procedures underlying the function `giofac` are described in [32] for continuous-time systems and in [29] for discrete-time systems. The function `giofac` relies on the mex-functions `sl_klf` and `sl_gminr`.

Examples

Example 23. This is Example 1 from [29] of the transfer function matrix of a discrete-time proper system:

$$G(z) = \begin{bmatrix} \dfrac{z^4 - \frac{z^3}{2} - 16\,z^2 - \frac{29\,z}{2} + 18}{z^4 + \frac{5\,z^3}{2} + 2\,z^2 + \frac{z}{2}} & \dfrac{z^4 + 5\,z^3 - z^2 - 11\,z + 6}{z^4 + \frac{5\,z^3}{2} + 2\,z^2 + \frac{z}{2}} & \dfrac{\frac{11\,z^3}{2} + 15\,z^2 + \frac{7\,z}{2} - 12}{z^4 + \frac{5\,z^3}{2} + 2\,z^2 + \frac{z}{2}} \\ \dfrac{-3\,z^2 + 12}{z^4 + \frac{5\,z^3}{2} + 2\,z^2 + \frac{z}{2}} & \dfrac{z^3 - z^2 - 4\,z + 4}{z^4 + \frac{5\,z^3}{2} + 2\,z^2 + \frac{z}{2}} & \dfrac{z^3 + 2\,z^2 - 4\,z - 8}{z^4 + \frac{5\,z^3}{2} + 2\,z^2 + \frac{z}{2}} \\ \dfrac{z^4 - \frac{z^3}{2} - 19\,z^2 - \frac{23\,z}{2} + 24}{z^4 + \frac{5\,z^3}{2} + 2\,z^2 + \frac{z}{2}} & \dfrac{z^4 + 6\,z^3 - 3\,z^2 - 12\,z + 8}{z^4 + \frac{5\,z^3}{2} + 2\,z^2 + \frac{z}{2}} & \dfrac{\frac{13\,z^3}{2} + 16\,z^2 - \frac{z}{2} - 16}{z^4 + \frac{5\,z^3}{2} + 2\,z^2 + \frac{z}{2}} \end{bmatrix}.$$

$G(z)$ has the zeros $\{1, 2, \infty\}$, the poles $\{0, -0.5, -1, -1\}$, and normal rank $r = 2$. The extended quasi-co-outer–inner factorization of $G(z)$ can be computed with the following sequence of commands:

```
% Oara (2005), Example 1
z = tf('z');  % define the complex variable z
% enter G(z) and determine a minimal state-space realization
G = 1/(z^4+5/2*z^3+2*z^2+z/2)*...
    [z^4-z^3/2-16*z^2-29/2*z+18 z^4+5*z^3-z^2-11*z+6 11/2*z^3+15*z^2+7/2*z-12
    -3*z^2+12 z^3-z^2-4*z+4 z^3+2*z^2-4*z-8;
    z^4-z^3/2-19*z^2-23/2*z+24 z^4+6*z^3-3*z^2-12*z+8 13/2*z^3+16*z^2-z/2-16];
sys = gir(ss(G),1.e-7);
gpole(sys)    % the system is marginally stable
gzero(sys)    % the system has an unstable zero and an infinite zero
nrank(sys)    % the normal rank of G(z) is 2

% compute the extended quasi-co-outer-inner factorization
% G(z) = [Go(z) 0]*Gi(z)
[sysi,syso,info] = goifac(sys,struct('tol',1.e-7));  % use tolerance 1.e-7

% check the factorization
nr = size(syso,2);                          % nr = 2 is the normal rank of G(z)
gnrank(syso*sysi(1:nr,:)-sys,1.e-7,rand)  % Go(z)*Gi(1:nr,:)(z) = G(z)
% check the innerness of Gi(z)
gnrank(sysi'*sysi-eye(3),1.e-7,rand)      % Gi'(z)*Gi(z) = I
```

```
gzero(syso)                    % Go(z) has no zeros outside the unit circle
info.nfuz                      % Go(z) has one zero on the unit circle
```

◊

3.6.9 grsfg

Syntax

```
SYSF = grsfg(SYS,GAMMA,OPTIONS)
```

Description

`grsfg` solves, for the transfer function matrix $G(\lambda)$ of a LTI descriptor state-space system, and a given γ satisfying $\gamma > \|G(\lambda)\|_\infty$, the right stable and minimum-phase spectral factorization problem

$$\gamma^2 I - G^\sim(\lambda)G(\lambda) = F^\sim(\lambda)F(\lambda)\,, \tag{247}$$

such that the resulting spectral factor $F(\lambda)$ is proper, stable and minimum-phase.

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{248}$$

with $x(t) \in \mathbb{R}^n$ and $y(t) \in \mathbb{R}^p$. $G(\lambda)$ must not have poles in $\partial\mathbb{C}_s$.

`GAMMA` is a given scalar γ, which must satisfy $\gamma > \|G(\lambda)\|_\infty$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	**Description**
`tol`	tolerance for the singular values based rank determination of E (Default: $n^2\|E\|_1$`eps`)
`tolmin`	tolerance for the singular values based observability tests (Default: $np\|C\|_\infty$`eps`)
`stabilize`	stabilization option: `true` – perform a preliminary stabilization using a left coprime factorization with inner denominator of $G(\lambda)$ (see **Method**) (default); `false` – no preliminary stabilization is performed.

Output data

`SYSF` contains the descriptor system state-space realization of the spectral factor $F(\lambda)$ in the form

$$\begin{aligned} E_F\lambda x_F(t) &= A_F x_F(t) + B_F v(t), \\ y_F(t) &= C_F x_F(t) + D_F v(t). \end{aligned} \tag{249}$$

Method

For the computation of the right spectral factorization (247) the dual of the two-step approach sketched in Section 2.9 is employed. In the first step, a preliminary left coprime factorization with inner denominator of $G(\lambda)$ is computed such that $G(\lambda) = M^{-1}(\lambda)N(\lambda)$, with both $N(\lambda)$ and $M(\lambda)$ stable, and $M(\lambda)$ inner. For this purpose, the dual algorithm to compute right coprime factorizations with inner denominators, given in **Procedure GRCFID** of [58], is employed. This step is not performed if `OPTIONS.stabilize = false`, in which case $N(\lambda) := G(\lambda)$ and $M(\lambda) = I_p$. In the second step, the spectral factorization problem is solved

$$\gamma^2 I - G^{\sim}(\lambda)G(\lambda) = \gamma^2 I - N^{\sim}(\lambda)N(\lambda) = F^{\sim}(\lambda)F(\lambda)$$

for the minimum-phase phase factor $F(\lambda)$. For this computation, the formulas provided by the dual versions of Lemma 6 and Lemma 7 are employed. These lemmas extend to proper descriptor system the formulas developed in [69].

Example

Example 24. Consider the discrete-time improper TFM

$$G(z) = \begin{bmatrix} z^2+z+1 & 4\,z^2+3\,z+2 & 2\,z^2-2 \\ z & 4\,z-1 & 2\,z-2 \\ z^2 & 4\,z^2-z & 2\,z^2-2\,z \end{bmatrix}, \tag{250}$$

which has two infinite poles (i.e., McMillan-degree of $G(z)$ is equal to 2) and has a minimal descriptor state-space realization of order 4. Therefore, the spectral factorization problem (247) has a solution for all $\gamma > \|G(z)\|_\infty = 10.4881$. With $\gamma = 1.1\|G(z)\|_\infty$, the function `grsfg` computes the proper minimal-phase spectral factor $F(z)$, having two poles in 0 and two stable zeros in $\{-0.2908, 0.4188\}$.

The spectral factor $F(z)$ can be computed with the following sequence of commands:

```
z = tf('z');  % define the complex variable z
% enter G(z) and determine a minimal state-space realization
G = [z^2+z+1 4*z^2+3*z+2 2*z^2-2;
    z 4*z-1 2*z-2;
    z^2 4*z^2-z 2*z^2-2*z];
sys = gir(ss(G));
gpole(sys)                % the system is unstable and improper
gamma = 1.1*norm(sys,inf) % set gamma = 1.1*||G||_inf

% compute the minimum-phase stable spectral factor F(z) satisfying
% gamma^2*I-conj(G(z))*G(z) = conj(F(z))*F(z)
sysf = grsfg(sys,gamma);

% check the factorization F'(z)*F(z)+G'(z)*G(z) = gamma^2*I
gnrank(sysf'*sysf+sys'*sys-gamma^2*eye(size(sys,2)),1.e-7,rand)

% check the stability of poles and zeros of F(z)
gpole(sysf), gzero(sysf)  % F(z) is stable and minimum-phase
```

◊

3.6.10 `glsfg`

Syntax

```
SYSF = glsfg(SYS,GAMMA,OPTIONS)
```

Description

`glsfg` solves, for the transfer function matrix $G(\lambda)$ of a LTI descriptor state-space system, and a given γ satisfying $\gamma > \|G(\lambda)\|_\infty$, the left stable and minimum-phase spectral factorization problem

$$\gamma^2 I - G(\lambda)G^\sim(\lambda) = F(\lambda)F^\sim(\lambda)\,, \tag{251}$$

such that the resulting spectral factor $F(\lambda)$ is proper, stable and minimum-phase.

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{252}$$

with $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$. $G(\lambda)$ must not have poles in $\partial\mathbb{C}_s$.

`GAMMA` is a given scalar γ, which must satisfy $\gamma > \|G(\lambda)\|_\infty$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	Description
`tol`	tolerance for the singular values based rank determination of E (Default: $n^2\|E\|_1$`eps`)
`tolmin`	tolerance for the singular values based controllability tests (Default: $nm\|B\|_1$`eps`)
`stabilize`	stabilization option: `true` – perform a preliminary stabilization using a right coprime factorization with inner denominator of $G(\lambda)$ (see **Method**) (default); `false` – no preliminary stabilization is performed.

Output data

`SYSF` contains the descriptor system state-space realization of the spectral factor $F(\lambda)$ in the form

$$\begin{aligned} E_F\lambda x_F(t) &= A_F x_F(t) + B_F v(t), \\ y_F(t) &= C_F x_F(t) + D_F v(t). \end{aligned} \tag{253}$$

Method

For the computation of the left spectral factorization (251) the two-step approach sketched in Section 2.9 is employed. In the first step, a preliminary right coprime factorization with inner denominator of $G(\lambda)$ is computed such that $G(\lambda) = N(\lambda)M^{-1}(\lambda)$, with both $N(\lambda)$ and $M(\lambda)$

stable, and $M(\lambda)$ inner. For this purpose, the algorithm to compute right coprime factorizations with inner denominators, given in **Procedure GRCFID** of [58], is employed. This step is not performed if `OPTIONS.stabilize = false`, in which case $N(\lambda) := G(\lambda)$ and $M(\lambda) = I_m$. In the second step, the left spectral factorization problem is solved

$$\gamma^2 I - G(\lambda)G^{\sim}(\lambda) = \gamma^2 I - N(\lambda)N^{\sim}(\lambda) = F(\lambda)F^{\sim}(\lambda)$$

for the minimum-phase phase factor $F(\lambda)$. For this computation, the formulas provided by Lemma 6 and Lemma 7 are employed. These lemmas extend to proper descriptor system the formulas developed in [69].

Example

Example 25. Consider the discrete-time improper TFM

$$G(z) = \begin{bmatrix} z^2+z+1 & 4\,z^2+3\,z+2 & 2\,z^2-2 \\ z & 4\,z-1 & 2\,z-2 \\ z^2 & 4\,z^2-z & 2\,z^2-2\,z \end{bmatrix}, \tag{254}$$

which has two infinite poles (i.e., McMillan-degree of $G(z)$ is equal to 2) and has a minimal descriptor state-space realization of order 4. Therefore, the spectral factorization problem (251) has a solution for all $\gamma > \|G(z)\|_\infty = 10.4881$. With $\gamma = 1.1\|G(z)\|_\infty$, the function `glsfg` computes the proper minimal-phase spectral factor $F(z)$, having two poles in 0 and two stable zeros in $\{-0.2908, 0.4188\}$.

The spectral factor $F(z)$ can be computed with the following sequence of commands:

```
z = tf('z');  % define the complex variable z
% enter G(z) and determine a minimal state-space realization
G = [z^2+z+1 4*z^2+3*z+2 2*z^2-2;
    z 4*z-1 2*z-2;
    z^2 4*z^2-z 2*z^2-2*z];
sys = gir(ss(G));
gpole(sys)                % the system is unstable and improper
gamma = 1.1*norm(sys,inf) % set gamma = 1.1*||G||_inf

% compute the minimum-phase stable spectral factor F(z) satisfying
% gamma^2*I-G(z)*conj(G(z)) = F(z)*conj(F(z))
sysf = glsfg(sys,gamma);

% check the factorization F(z)*F'(z)+G(z)*G'(z) = gamma^2*I
gnrank(sysf*sysf'+sys*sys'-gamma^2*eye(size(sys,1)),1.e-7,rand)

% check the stability of poles and zeros of F(z)
gpole(sysf), gzero(sysf)  % F(z) is stable and minimum-phase
```

$\Diamond$

3.7 Functions for Approximations

These functions cover the computation of Nehari approximations and the solution of the 1-block and 2-block least distance problems.

3.7.1 gnehari

Syntax

```
[SYSX,S1] = gnehari(SYS)
[SYSX,S1] = gnehari(SYS,GAMMA)
[SYSX,S1] = gnehari(SYS,GAMMA,TOL)
```

Description

`gnehari` computes an optimal or suboptimal stable Nehari approximation of the transfer function matrix $G(\lambda)$ of a LTI descriptor state-space system. The optimal Nehari approximation $X(\lambda)$ satisfies

$$\|G(\lambda) - X(\lambda)\|_\infty = \|G_u^\sim(\lambda)\|_H\,, \tag{255}$$

where $G_u(\lambda)$ is the anti-stable part of $G(\lambda)$. For a given $\gamma > \|G_u^\sim(\lambda)\|_H$, the suboptimal approximation satisfies

$$\|G(\lambda) - X(\lambda)\|_\infty < \gamma\,. \tag{256}$$

Input data

`SYS` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t),\\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{257}$$

$G(\lambda)$ must not have poles in $\partial\mathbb{C}_s$.

`GAMMA`, if specified, is the desired suboptimality level γ for the suboptimal Nehari approximation problem (256) and must satisfy $\gamma > \|G_u^\sim(\lambda)\|_H$, where $G_u(\lambda)$ is the anti-stable part of $G(\lambda)$. If `GAMMA = [ ]`, then the optimal Nehari approximation problem (255) is solved.

`TOL`, if specified, is a relative tolerance used for rank computations
(Default: internally computed).

Output data

`SYSX` contains the descriptor system state-space realization of the optimal or suboptimal stable Nehari approximation $X(\lambda)$ in the form

$$\begin{aligned} E_X\lambda x_X(t) &= A_X x_X(t) + B_X u(t),\\ y_X(t) &= C_X x_X(t) + D_X u(t). \end{aligned} \tag{258}$$

`S1` is the Hankel-norm of the antistable part of $G(\lambda)$ (also the $\mathcal{L}_\infty$-norm of the optimal approximation error).

Method

The case when $G(\lambda)$ is anstistable, is discussed Section 2.15. For a general $G(\lambda)$ without poles in $\partial\mathbb{C}_s$, a preliminary spectral separation is performed as

$$G(\lambda) = G_s(\lambda) + G_u(\lambda), \tag{259}$$

where $G_s(\lambda)$ is the stable part (i.e., all poles of $G_s(\lambda)$ are in $\mathbb{C}_s$) and $G_u(\lambda)$ is the antistable part (i.e., all poles of $G_u(\lambda)$ are in $\mathbb{C}_u$). The optimal or suboptimal Nehari approximation of $G_u(\lambda)$ is then computed, by determining $Y^{\sim}(\lambda)$, the optimal zeroth-order Hankel-norm approximation or the suboptimal Hankel-norm approximation of $G_u^{\sim}(\lambda)$, respectively, using the methods proposed in [15] (see also [34]) with straightforward extensions for proper descriptor systems. For the computation of the optimal Nehari approximation, the system balancing-based **Procedure GNEHARI** in [59] (extended to non-square systems) is used. The solution of the Nehari approximation problem for the original problem is obtained as

$$X(\lambda) = G_s(\lambda) + Y(\lambda).$$

Explicit approximation formulas developed in [15] are employed for continuous-time systems, while for discrete-time systems, the bilinear transformation based approach suggested in [15] is used. For the computation of additive spectral separation (259) of the descriptor system `SYS`, the mex-function `sl_gsep` is employed. In the computation of the balancing transformations, the intervening Lyapunov and Stein equations, satisfied by the gramians, have been solved directly for the Cholesky factors of the gramians using the mex-function `sl_glme`.

Example 26. Consider the discrete-time improper TFM

$$G(z) = \begin{bmatrix} z^2+z+1 & 4\,z^2+3\,z+2 & 2\,z^2-2 \\ z & 4\,z-1 & 2\,z-2 \\ z^2 & 4\,z^2-z & 2\,z^2-2\,z \end{bmatrix}, \tag{260}$$

which has two infinite poles (i.e., $G(z)$ is antistable and its McMillan-degree is equal to 2) and has a minimal descriptor state-space realization of order 4. The Hankel-norm of $G^{\sim}(z)$ is $\|G^{\sim}(z)\|_H = 8.6622$, and therefore the optimal stable Nehari approximation $X(z)$ must satisfy $\|G(z)-X(z)\|_\infty = 8.6622$. Indeed, the optimal stable Nehari approximation, computed with the function `gnehari`, achieves exactly this approximation error, with $X(z)$ having the McMillan degree equal to 1. In contrast, the suboptimal stable Nehari approximation computed for $\gamma = 10$, has McMillan degree 2 and the achieved approximation error is 9.8207.

The optimal and suboptimal Nehari approximations can be computed with the following sequence of commands:

```
z = tf('z');  % define the complex variable z
% enter G(z) and determine a minimal state-space realization
G = [z^2+z+1 4*z^2+3*z+2 2*z^2-2;
    z 4*z-1 2*z-2;
    z^2 4*z^2-z 2*z^2-2*z];
sys = gir(ss(G));
gpole(sys)                % the system is unstable and improper
ghanorm(sys')             % the achievable optimal approximation error
```

```
% compute the optimal Nehari approximation
[sysx,s1] = gnehari(sys);

% check the approximation error ||G(z)-X(z)||_inf = s1
norm(gminreal(sys-sysx,1.e-7),inf)-s1

% check the stability of poles of X(z)
gpole(sysx)  % X(z) is stable and has order 1

% compute the suboptimal Nehari approximation for gamma = 10
[sysxsub,s1] = gnehari(sys,10);

% compute the approximation error ||G(z)-Xsub(z)||_inf
norm(gminreal(sys-sysxsub,1.e-7),inf)

% check the stability of poles of Xsub(z)
gpole(sysxsub)  % Xsub(z) is stable and has order 2
```

$\Diamond$

3.7.2 glinfldp

Syntax

```
[SYSX,MINDIST] = glinfldp(SYS1,SYS2,OPTIONS)
[SYSX,MINDIST] = glinfldp(SYS,M2,OPTIONS)
```

Description

`glinfldp` solves the 2-block optimal least distance problem to find a stable $X(\lambda)$ such that

$$\|[\,G_1(\lambda) - X(\lambda)\;\; G_2(\lambda)\,]\|_\infty = \min\,, \tag{261}$$

or the 2-block suboptimal least distance problem to find a stable $X(\lambda)$ such that

$$\|[\,G_1(\lambda) - X(\lambda)\;\; G_2(\lambda)\,]\|_\infty < \gamma\,, \tag{262}$$

where $G_1(\lambda)$ and $G_2(\lambda)$ are the transfer function matrices of LTI descriptor state-space systems and $\gamma > \|G_2(\lambda)\|_\infty$.

Input data

For the usage with

```
[SYSX,MINDIST] = glinfldp(SYS1,SYS2,OPTIONS)
```

the input parameters `SYS1` and `SYS2` are as follows:

`SYS1` is a LTI system, whose transfer function matrix is $G_1(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_1\lambda x_1(t) &= A_1 x_1(t) + B_1 u(t), \\ y_1(t) &= C_1 x_1(t) + D_1 u(t), \end{aligned} \tag{263}$$

where $y_1(t) \in \mathbb{R}^p$ and $u(t) \in \mathbb{R}^{m_1}$. $G_1(\lambda)$ must not have poles in $\partial\mathbb{C}_s$.

`SYS2` is a LTI system, whose transfer function matrix is $G_2(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_2\lambda x_2(t) &= A_2 x_2(t) + B_2 v(t), \\ y_2(t) &= C_2 x_2(t) + D_2 v(t), \end{aligned} \tag{264}$$

where $y_2(t) \in \mathbb{R}^p$ and $v(t) \in \mathbb{R}^{m_2}$. If `SYS2` is empty, a 1-block least distance problem is solved. $G_2(\lambda)$ must not have poles in $\partial\mathbb{C}_s$.

For the usage with

```
[SYSX,MINDIST] = glinfldp(SYS,M2,OPTIONS)
```

the input parameters `SYS` and `M2` are as follows:

`SYS` is an input concatenated compound LTI system, `SYS = [ SYS1 SYS2 ]`, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_1 u(t) + B_2 v(t), \\ y(t) &= Cx(t) + D_1 u(t) + D_2 v(t), \end{aligned} \tag{265}$$

where `SYS1` has the transfer function matrix $G_1(\lambda)$ with the descriptor system realization $(A - \lambda E, B_1, C, D_1)$ and `SYS2` has the transfer function matrix $G_2(\lambda)$ with the descriptor system realization $(A - \lambda E, B_2, C, D_2)$. $[\, G_1(\lambda)\ G_2(\lambda)\,]$ must not have poles in $\partial\mathbb{C}_s$.

`M2` is the dimension $m_2 \geq 0$ of the input vector $v(t)$ of the system `SYS2`. If $m_2 = 0$, a 1-block least distance problem is solved.

For both usages:

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	Description
`tol`	tolerance for rank determinations (Default: internally computed)
`reltol`	specifies the relative tolerance $rtol$ for the desired accuracy of the gamma-iteration. The iterations are performed until the current estimates of the maximum distance γ_u and minimum distance γ_l, which bound the optimal distance γ_{opt} (i.e., $\gamma_l \leq \gamma_{opt} \leq \gamma_u$), satisfy $\gamma_u - \gamma_l < rtol(\|[\, G_1(\lambda)\ G_2(\lambda)\,]\|_\infty - \|G_2(\lambda)\|_\infty)$. (Default: $rtol = 10^{-4}$)
`gamma`	desired suboptimality level γ for the suboptimal least distance problem (262) (Default: `[ ]`, i.e., the optimal least distance problem (261) is solved)

Output data

`SYSX` contains the descriptor system state-space realization of the optimal or suboptimal solution $X(\lambda)$ in the form

$$\begin{aligned} E_X \lambda x_X(t) &= A_X x_X(t) + B_X u(t), \\ y_X(t) &= C_X x_X(t) + D_X u(t). \end{aligned} \tag{266}$$

`MINDIST` is the achieved distance by the computed (optimal or suboptimal) solution $X(\lambda)$.

Method

The solution approach is sketched in Section 2.16 and corresponds to extensions of the method proposed in [6] to descriptor system state-space representations.

Example

Example 27. The formulation of many so-called approximate model-matching problems can be done as error minimization problems, where approximate solutions of rational equations of the form $X(\lambda)G(\lambda) = F(\lambda)$ are determined by minimizing the $\mathcal{L}_\infty$-norm of the error $\mathcal{E}(\lambda) := F(\lambda) - X(\lambda)G(\lambda)$. For example, the standard formulation of the optimal $\mathcal{H}_\infty$ *model-matching problem* ($\mathcal{H}_\infty$-MMP) is: given $G(\lambda), F(\lambda) \in \mathcal{H}_\infty$, find $X(\lambda) \in \mathcal{H}_\infty$ which minimizes $\|\mathcal{E}(\lambda)\|_\infty$. The optimal solution is typically computed by solving a sequence of suboptimal $\mathcal{H}_\infty$ model-matching problems, such that $\|\mathcal{E}(\lambda)\|_\infty < \gamma$, for suitably chosen γ values.

For the solution of the $\mathcal{H}_\infty$-MMPs we can often assume that $G(\lambda)$ has full row rank for all $\lambda \in \partial\mathbb{C}_s$ and employ the (extended) co-outer–coinner factorization of $G(\lambda)$ to reduce this problem to a $\mathcal{H}_\infty$ least distance problem (LDP). Consider the extended factorization

$$G(\lambda) = \begin{bmatrix} G_o(\lambda) & 0 \end{bmatrix} G_i(\lambda) = \begin{bmatrix} G_o(\lambda) & 0 \end{bmatrix} \begin{bmatrix} G_{i,1}(\lambda) \\ G_{i,2}(\lambda) \end{bmatrix} = G_o(\lambda) G_{i,1}(\lambda),$$

where $G_i(\lambda) := \begin{bmatrix} G_{i,1}(\lambda) \\ G_{i,2}(\lambda) \end{bmatrix}$ is square and inner and $G_o(\lambda)$ is square and outer (therefore invertible in $\mathcal{H}_\infty$). This allows to write successively

$$\begin{aligned} \|\mathcal{E}(\lambda)\|_\infty &= \|F(\lambda) - X(\lambda)G(\lambda)\|_\infty \\ &= \left\| \left(F(\lambda) G_i^\sim(\lambda) - X(\lambda) \begin{bmatrix} G_o(\lambda) & 0 \end{bmatrix} \right) G_i(\lambda) \right\|_\infty \\ &= \left\| \begin{bmatrix} \widetilde{F}_1(\lambda) - Y(\lambda) & \widetilde{F}_2(\lambda) \end{bmatrix} \right\|_\infty , \end{aligned}$$

where $Y(\lambda) := X(\lambda)G_o(\lambda) \in \mathcal{H}_\infty$ and

$$F(\lambda) G_i^\sim(\lambda) = \left[\, F(\lambda) G_{i,1}^\sim(\lambda) \, \middle| \, F(\lambda) G_{i,2}^\sim(\lambda) \, \right] := \left[\, \widetilde{F}_1(\lambda) \, \middle| \, \widetilde{F}_2(\lambda) \, \right] .$$

Thus, the problem of computing a stable $X(\lambda)$ which minimizes the error norm $\|\mathcal{E}(\lambda)\|_\infty$ has been reduced to a LDP to compute the stable solution $Y(\lambda)$ which minimizes the distance $\left\| [\, \widetilde{F}_1(\lambda) - Y(\lambda) \;\; \widetilde{F}_2(\lambda) \,] \right\|_\infty$. A γ-iteration based approach is used for this purpose, as described in [12]. The solution of the original MMP is given by

$$X(\lambda) = Y(\lambda) G_o^{-1}(\lambda) .$$

We apply this approach to solve a MMP discussed in the book of Francis [12, Example 1, p. 112] with

$$G(s) = \begin{bmatrix} -\dfrac{s-1}{s^2+s+1} & \dfrac{s(s-1)}{s^2+s+1} \end{bmatrix} W(s), \quad F(s) = [\, W(s) \;\; 0 \,],$$

where $W(s) = (s+1)/(10s+1)$ is a suitable weighting factor. The optimal solution

$$X(s) = \frac{2.3144(s+0.4569)(s+2.189)(s^2+s+1)}{(s+3.095)(s+2.189)(s+1)(s+0.4569)}$$

leads to the optimal error norm $\|F(\lambda) - X(\lambda)G(\lambda)\|_\infty = 0.2521$, which fully agrees with the computed minimum distance `mindist` (see bellow). The solution in Francis' book [12, p. 114] corresponds to a suboptimal solution for $\gamma = 0.2729$ and is

$$X_{sub}(s) = \frac{2.2731(s+0.4732)(s+2.183)(s^2+s+1)}{(s+3.108)(s+2.189)(s+1)(s+0.4569)}.$$

The corresponding suboptimal error norm $\|F(\lambda) - X_{sub}(\lambda)G(\lambda)\|_\infty = 0.2536$. The solutions of the optimal and suboptimal $\mathcal{H}_\infty$-MMPs can be computed using the following MATLAB code:

```
s = tf('s');  % define the complex variable s
% enter W(s), G(s) and F(s)
W = (s+1)/(10*s+1);    % weighting function
G = [ -(s-1)/(s^2+s+1) (s^2-2*s)/(s^2+s+1)]*W;
F = [ W 0 ];

sys = ss(G);

% compute the extended outer-coinner factorization
[Gi,Go] = goifac(sys,struct('tol',1.e-7));

% define the LDP
Fbar = F*Gi'; m2 = 1;

% compute the optimal solution of the LDP
[Y,mindist] = glinfldp(Fbar,m2); mindist

% compute the optimal solution of the MMP
X = minreal(zpk(Y/Go))

% compute the error norm of the optimal solution
norm(X*G-F,inf)

% compute the suboptimal solution of the LDP
[Ysub,mindistsub] = glinfldp(Fbar,m2,struct('gamma',0.2729)); mindistsub
```

```
% compute the suboptimal solution of the MMP
Xsub = minreal(zpk(Ysub/Go))

% compute the error norm of the suboptimal solution
norm(Xsub*G-F,inf)
```

$\Diamond$

3.7.3 grasol

Syntax

```
[SYSX,INFO] = grasol(SYSG,SYSF,OPTIONS)
[SYSX,INFO] = grasol(SYSGF,MF,OPTIONS)
```

Description

`grasol` computes for the (rational) transfer function matrices $G(\lambda)$ and $F(\lambda)$ of LTI descriptor systems, an approximate solution $X(\lambda)$ of the linear rational matrix equation $G(\lambda)X(\lambda) = F(\lambda)$. The optimal solution $X(\lambda)$ minimizes the error norm such that

$$\|G(\lambda)X(\lambda) - F(\lambda)\|_{\infty/2} = \min. \tag{267}$$

Optionally, for a given suboptimality level γ, a suboptimal solution $X(\lambda)$ is determined such that

$$\|G(\lambda)X(\lambda) - F(\lambda)\|_{\infty} \leq \gamma. \tag{268}$$

The resulting $X(\lambda)$ has all poles stable or lying on the boundary of the stability domain.

Input data

For the usage with

```
[SYSX,INFO] = grasol(SYSG,SYSF,OPTIONS)
```

the input parameters are as follows:

`SYSG` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_G\lambda x_G(t) &= A_G x_G(t) + B_G u(t), \\ y_G(t) &= C_G x_G(t) + D_G u(t), \end{aligned} \tag{269}$$

where $y_G(t) \in \mathbb{R}^p$.

`SYSF` is a stable LTI system, whose transfer function matrix is $F(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_F\lambda x_F(t) &= A_F x_F(t) + B_F v(t), \\ y_F(t) &= C_F x_F(t) + D_F v(t), \end{aligned} \tag{270}$$

where $y_F(t) \in \mathbb{R}^p$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

OPTIONS fields	Description
tol	relative tolerance for rank computations (Default: internally computed);
offset	stability boundary offset β, to be used to assess the finite zeros which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite zeros having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite zeros having moduli in the interval $[1-\beta, 1+\beta]$ (Default: $\beta = 1.4901 \cdot 10^{-08}$).
sdeg	prescribed stability degree for the free poles of the solution $X(\lambda)$ (Default: [ ], in which case: -0.05, in the continuous-time case; 0.95, in the discrete-time case)
poles	a complex conjugated set of desired poles to be assigned for the free poles of the solution $X(\lambda)$ (Default: [ ]).
mindeg	option to compute a minimum degree solution: true – determine, if possible, a minimum order solution with all poles in $\mathbb{C}_s \cup \partial\mathbb{C}_s$; false – determine a particular solution which has possibly non-minimal order and all its poles in $\mathbb{C}_s \cup \partial\mathbb{C}_s$ (default).
H2sol	option to compute an $\mathcal{H}_2$-norm optimal solution : true – compute a $\mathcal{H}_2$-norm optimal solution; false – compute a $\mathcal{H}_\infty$-norm optimal or suboptimal solution (default).
reltol	specifies the relative tolerance $rtol$ for the desired accuracy of the γ-iteration to solve the $\mathcal{H}_\infty$ least distance problem (see **Method**). The iterations are performed until the current estimates of the maximum distance γ_u and minimum distance γ_l, which bound the optimal distance γ_{opt} (i.e., $\gamma_l \leq \gamma_{opt} \leq \gamma_u$), satisfy $\gamma_u - \gamma_l < rtol \cdot (\gamma_u^0 - \gamma_l^0)$, where γ_u^0 and γ_l^0 are the initial estimates of γ_u and γ_l, respectively. (Default: $rtol = 10^{-4}$)
gamma	desired suboptimality level γ for solving the suboptimal model-matching problem (268) (Default: [ ], i.e., the optimal model-matching problem (267) is solved)

For the usage with

```
[SYSX,INFO] = grasol(SYSGF,MF,OPTIONS)
```

the input parameters are as follows:

SYSGF is an input concatenated compound LTI system, SYSGF = [SYSG SYSF], in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_G u(t) + B_F v(t), \\ y(t) &= Cx(t) + D_G u(t) + D_F v(t), \end{aligned} \tag{271}$$

where SYSG has the transfer function matrix $G(\lambda)$, with the descriptor system realization $(A - \lambda E, B_G, C, D_G)$, and SYSF has the transfer function matrix $F(\lambda)$, with the descriptor system realization $(A - \lambda E, B_F, C, D_F)$.

MF is the dimension of the input vector $v(t)$ of the system SYSF.

`OPTIONS` is a MATLAB structure to specify user options and has the same fields as described previously.

Output data

`SYSX` contains the descriptor system state-space realization of the solution $X(\lambda)$ in the form

$$\begin{aligned} \widetilde{E}\lambda\widetilde{x}(t) &= \widetilde{A}\widetilde{x}(t) + \widetilde{B}v(t), \\ u(t) &= \widetilde{C}\widetilde{x}(t) + \widetilde{D}v(t). \end{aligned} \tag{272}$$

`INFO` is a MATLAB structure containing additional information, as follows:

`INFO` fields	Description
`nrank`	normal rank of the transfer function matrix $G(\lambda)$;
`mindist`	achieved approximation error norm $\|G(\lambda)X(\lambda) - F(\lambda)\|_{\infty/2}$;
`nonstandard`	logical value, which is set to `true` for a *non-standard problem* (when $G(\lambda)$ has zeros in $\partial\mathbb{C}_s$ within the stability offset `OPTIONS.offset` for finite zeros), and to `false` for a *standard problem* (when $G(\lambda)$ has no zeros in $\partial\mathbb{C}_s$);

Method

The employed solution approach is sketched in Section 2.17 for the *left* linear rational matrix equation $X(\lambda)G(\lambda) = F(\lambda)$ and extends the approach proposed in [12] to arbitrary $G(\lambda)$. In what follows, we give the three main steps of a similar approach to solve the *right* linear rational matrix equation $G(\lambda)X(\lambda) = F(\lambda)$:

1. Compute the (extended) inner–quasi-outer factorization of $G(\lambda)$ as

$$G(\lambda) = G_i(\lambda)\begin{bmatrix} G_o(\lambda) \\ 0 \end{bmatrix} = [\, G_{i,1}(\lambda)\ G_{i,2}(\lambda)\,]\begin{bmatrix} G_o(\lambda) \\ 0 \end{bmatrix} = G_{i,1}(\lambda)G_o(\lambda), \tag{273}$$

   where $G_i(\lambda) := [\, G_{i,1}(\lambda)\ G_{i,2}(\lambda)\,]$ is square and inner, and $G_o(\lambda)$ is full row rank (i.e., is right invertible), has the same poles as $G(\lambda)$ and has only zeros in $\mathbb{C}_s \cup \partial\mathbb{C}_s$.

2. Compute

$$G_i^{\sim}(\lambda)F(\lambda) = \begin{bmatrix} G_{i,1}^{\sim}(\lambda)F(\lambda) \\ \hline G_{i,2}^{\sim}(\lambda)F(\lambda) \end{bmatrix} := \begin{bmatrix} \widetilde{F}_1(\lambda) \\ \hline \widetilde{F}_2(\lambda) \end{bmatrix}$$

   and compute the stable solution $Y(\lambda)$ of the least distance problem

$$\gamma_{opt} := \left\| \begin{bmatrix} \widetilde{F}_1(\lambda) - Y(\lambda) \\ \widetilde{F}_2(\lambda) \end{bmatrix} \right\|_{\infty/2} = \min$$

   or of the γ-suboptimal distance problem

$$\left\| \begin{bmatrix} \widetilde{F}_1(\lambda) - Y(\lambda) \\ \widetilde{F}_2(\lambda) \end{bmatrix} \right\|_{\infty/2} \leq \gamma.$$

3. Compute an exact solution $X(\lambda)$ of the linear rational equation $G_o(\lambda)X(\lambda) = Y(\lambda)$, where $G_o(\lambda)$ is left invertible.

If `OPTIONS.mindeg = false`, the computed solution $X(\lambda)$ at Step 3 may not have the least achievable McMillan degree (excepting the case when $G_o(\lambda)$ is invertible), but its only unstable poles are (part of) the unstable zeros of $G_o(\lambda)$. These poles are also (part of) the zeros of $G(\lambda)$ lying in $\partial\mathbb{C}_s$. If $G_o(\lambda)$ is non-square, $X(\lambda)$ has a number of free poles, which are assigned to values specified via the option parameters `OPTIONS.sdeg` and `OPTIONS.poles`.

If `OPTIONS.mindeg = true`, the least order solution $X(\lambda)$ of the linear rational equation $G_o(\lambda)X(\lambda) = Y(\lambda)$ has the least achievable McMillan degree, but besides possible unstable poles which are the unstable zeros of $G_o(\lambda)$ lying in $\partial\mathbb{C}_s$, $X(\lambda)$ may also have additional unstable poles (e.g., in the case when $G_o(\lambda)$ has no least order stable right inverse). In this case, the solution corresponding to `OPTIONS.mindeg = false` is computed.

The resulting value of the achieved distance $\left\| \begin{bmatrix} \widetilde{F}_1(\lambda) - Y(\lambda) \\ \widetilde{F}_2(\lambda) \end{bmatrix} \right\|_{\infty/2}$ at Step 2, is returned in `INFO.mindist`. If `OPTIONS.H2sol = true`, an $\mathcal{H}_2$-LDP is solved (see Section 2.17), in which case the distance may be infinite if $\widetilde{F}_2(\lambda)$ is not strictly proper.

Example

Example 28. This example is taken from [14] and has been also considered in *Example* 6, where an exact model matching problem is solved for a stable solution of $G(s)X(s) = F(s)$, with

$$G(s) = \begin{bmatrix} \dfrac{s-1}{s\ (s+1)} & \dfrac{s-1}{s\ (s+2)} \end{bmatrix}, \quad F(s) = \begin{bmatrix} \dfrac{s-1}{(s+1)\ (s+3)} & \dfrac{s-1}{(s+1)\ (s+4)} \end{bmatrix}.$$

Both $G(s)$ and $[\,G(s)\ F(s)\,]$ have rank equal to 1 and zeros $\{1, \infty\}$. It follows, according to Lemma 9, that the linear rational matrix equation $G(s)X(s) = F(s)$ has a stable and proper solution. A fourth order stable and proper solution has been computed in [14]. A least order solution, with McMillan degree equal to 2, has been computed with `grasol` as

$$X(s) = \begin{bmatrix} \dfrac{0.75(s-0.3333)}{s+3} & \dfrac{0.75(s-0.3333)}{s+4} \\ \dfrac{0.25(s+2)}{s+3} & \dfrac{0.25(s+2)}{s+4} \end{bmatrix}.$$

Note that this solution is different (but of same order) from the exact solution computed with `grsol` in *Example* 6.

To compute a least order solution $X(s)$, the following sequence of commands can be used:

```
% Gao & Antsaklis (1989)
s = tf('s');
G = [(s-1)/(s*(s+1))  (s-1)/(s*(s+2))];
F = [(s-1)/((s+1)*(s+3))  (s-1)/((s+1)*(s+4))];

% solve G(s)*X(s) = F(s) for the least order solution
[X,info] = grasol(ss(G),ss(F),struct('mindeg',true,'tol',1.e-7)); info
minreal(zpk(X))
```

```
% check  solution
minreal(G*X-F)
```

$\Diamond$

Example 29. We solve the $\mathcal{H}_\infty$-MMP discussed in the book of Francis [12, Example 1, p. 112] with

$$G(s) = \begin{bmatrix} -\dfrac{s-1}{s^2+s+1} \\ \dfrac{s(s-1)}{s^2+s+1} \end{bmatrix} W(s), \quad F(s) = \begin{bmatrix} W(s) \\ 0 \end{bmatrix},$$

where $W(s) = (s+1)/(10s+1)$ is a suitable weighting factor. The optimal solution

$$X(s) = \frac{2.3144(s+0.4569)(s+2.189)(s^2+s+1)}{(s+3.095)(s+2.189)(s+1)(s+0.4569)}$$

leads to the optimal error norm $\|F(\lambda) - G(\lambda)X(\lambda)\|_\infty = 0.2521$. The solution in Francis' book [12, p. 114] corresponds to a suboptimal solution for $\gamma = 0.2729$ and is

$$X_{sub}(s) = \frac{2.2731(s+0.4732)(s+2.183)(s^2+s+1)}{(s+3.108)(s+2.189)(s+1)(s+0.4569)}.$$

The corresponding suboptimal error norm $\|F(\lambda) - G(\lambda)X_{sub}(\lambda)\|_\infty = 0.2536$.

The solutions of the optimal and suboptimal $\mathcal{H}_\infty$-MMPs can be computed using the following MATLAB code:

```
s = tf('s');  % define the complex variable s
% enter W(s), G(s) and F(s)
W = (s+1)/(10*s+1);    % weighting function
G = [ -(s-1)/(s^2+s+1); (s^2-2*s)/(s^2+s+1)]*W;
F = [ W; 0 ];

% solve G(s)*X(s) = F(s) for the least order solution
[X,info] = grasol(ss(G),ss(F),struct('mindeg',true,'tol',1.e-7)); info
minreal(zpk(X))

% compute the error norm of the optimal solution
norm(G*X-F,inf)


% compute the suboptimal solution for gamma = 0.2729
opts = struct('mindeg',true,'tol',1.e-7,'gamma',0.2729);
[Xsub,info] = grasol(ss(G),ss(F),opts); info
minreal(zpk(Xsub))

% compute the error norm of the suboptimal solution
norm(G*Xsub-F,inf)
```

$\Diamond$

3.7.4 glasol

Syntax

```
[SYSX,INFO] = glasol(SYSG,SYSF,OPTIONS)
[SYSX,INFO] = glasol(SYSGF,MF,OPTIONS)
```

Description

`glasol` computes for the (rational) transfer function matrices $G(\lambda)$ and $F(\lambda)$ of LTI descriptor systems, an approximate solution $X(\lambda)$ of the linear rational matrix equation $X(\lambda)G(\lambda) = F(\lambda)$. The optimal solution $X(\lambda)$ minimizes the error norm such that

$$\|X(\lambda)G(\lambda) - F(\lambda)\|_{\infty/2} = \min . \tag{274}$$

Optionally, for a given suboptimality level γ, a suboptimal solution $X(\lambda)$ is determined such that

$$\|X(\lambda)G(\lambda) - F(\lambda)\|_{\infty} \leq \gamma. \tag{275}$$

The resulting $X(\lambda)$ has all poles stable or lying on the boundary of the stability domain.

Input data

For the usage with

```
[SYSX,INFO] = glasol(SYSG,SYSF,OPTIONS)
```

the input parameters are as follows:

`SYSG` is a LTI system, whose transfer function matrix is $G(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_G\lambda x_G(t) &= A_G x_G(t) + B_G u(t), \\ y_G(t) &= C_G x_G(t) + D_G u(t), \end{aligned} \tag{276}$$

where $u(t) \in \mathbb{R}^m$.

`SYSF` is a stable LTI system, whose transfer function matrix is $F(\lambda)$, and is in a descriptor system state-space form

$$\begin{aligned} E_F\lambda x_F(t) &= A_F x_F(t) + B_F v(t), \\ y_F(t) &= C_F x_F(t) + D_F v(t), \end{aligned} \tag{277}$$

where $v(t) \in \mathbb{R}^m$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

OPTIONS fields	Description
tol	relative tolerance for rank computations (Default: internally computed);
offset	stability boundary offset β, to be used to assess the finite zeros which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite zeros having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite zeros having moduli in the interval $[1-\beta, 1+\beta]$ (Default: $\beta = 1.4901 \cdot 10^{-08}$).
sdeg	prescribed stability degree for the free poles of the solution $X(\lambda)$ (Default: [ ], in which case: -0.05, in the continuous-time case; 0.95, in the discrete-time case)
poles	a complex conjugated set of desired poles to be assigned for the free poles of the solution $X(\lambda)$ (Default: [ ]).
mindeg	option to compute a minimum degree solution: true – determine, if possible, a minimum order solution with all poles in $\mathbb{C}_s \cup \partial\mathbb{C}_s$; false – determine a particular solution which has possibly non-minimal order and all its poles in $\mathbb{C}_s \cup \partial\mathbb{C}_s$ (default).
H2sol	option to compute an $\mathcal{H}_2$-norm optimal solution : true – compute a $\mathcal{H}_2$-norm optimal solution; false – compute a $\mathcal{H}_\infty$-norm optimal or suboptimal solution (default).
reltol	specifies the relative tolerance $rtol$ for the desired accuracy of the γ-iteration to solve the $\mathcal{H}_\infty$ least distance problem (see **Method**). The iterations are performed until the current estimates of the maximum distance γ_u and minimum distance γ_l, which bound the optimal distance γ_{opt} (i.e., $\gamma_l \leq \gamma_{opt} \leq \gamma_u$), satisfy $\gamma_u - \gamma_l < rtol \cdot (\gamma_u^0 - \gamma_l^0)$, where γ_u^0 and γ_l^0 are the initial estimates of γ_u and γ_l, respectively. (Default: $rtol = 10^{-4}$)
gamma	desired suboptimality level γ for solving the suboptimal model-matching problem (275) (Default: [ ], i.e., the optimal model-matching problem (274) is solved)

For the usage with

```
[SYSX,INFO] = glasol(SYSGF,MF,OPTIONS)
```

the input parameters are as follows:

SYSGF is an output concatenated compound LTI system, SYSGF = [SYSG; SYSF], in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y_G(t) &= C_G x(t) + D_G u(t), \\ y_F(t) &= C_F x(t) + D_F u(t), \end{aligned} \tag{278}$$

where SYSG has the transfer function matrix $G(\lambda)$, with the descriptor system realization $(A - \lambda E, B, C_G, D_G)$, and SYSF has the transfer function matrix $F(\lambda)$, with the descriptor system realization $(A - \lambda E, B, CF, D_F)$.

`MF` is the dimension of the output vector $y_F(t)$ of the system `SYSF`.

`OPTIONS` is a MATLAB structure to specify user options and has the same fields as described previously.

Output data

`SYSX` contains the descriptor system state-space realization of the solution $X(\lambda)$ in the form

$$\begin{aligned} \widetilde{E}\lambda\widetilde{x}(t) &= \widetilde{A}\widetilde{x}(t) + \widetilde{B}y_G(t), \\ y(t) &= \widetilde{C}\widetilde{x}(t) + \widetilde{D}y_G(t). \end{aligned} \tag{279}$$

`INFO` is a MATLAB structure containing additional information, as follows:

`INFO` fields	Description
`nrank`	normal rank of the transfer function matrix $G(\lambda)$;
`mindist`	achieved approximation error norm $\|X(\lambda)G(\lambda) - F(\lambda)\|_{\infty/2}$;
`nonstandard`	logical value, which is set to `true` for a *non-standard problem* (when $G(\lambda)$ has zeros in $\partial\mathbb{C}_s$ within the stability offset `OPTIONS.offset` for finite zeros), and to `false` for a *standard problem* (when $G(\lambda)$ has no zeros in $\partial\mathbb{C}_s$);

Method

The employed solution approach to determine the approximate solution of the *left* linear rational matrix equation $X(\lambda)G(\lambda) = F(\lambda)$ is sketched in Section 2.17 and extends the approach proposed in [12] to arbitrary $G(\lambda)$. The three main steps of this approach are:

1. Compute the (extended) quasi-co-outer–coinner factorization of $G(\lambda)$ as

$$G(\lambda) = \begin{bmatrix} G_o(\lambda) & 0 \end{bmatrix} G_i(\lambda) = \begin{bmatrix} G_o(\lambda) & 0 \end{bmatrix} \begin{bmatrix} G_{i,1}(\lambda) \\ G_{i,2}(\lambda) \end{bmatrix} = G_o(\lambda)G_{i,1}(\lambda), \tag{280}$$

   where $G_i(\lambda) := \left[\begin{smallmatrix} G_{i,1}(\lambda) \\ G_{i,2}(\lambda) \end{smallmatrix}\right]$ is square and inner, and $G_o(\lambda)$ is full column rank (i.e., is left invertible), has the same poles as $G(\lambda)$ and has only zeros in $\mathbb{C}_s \cup \partial\mathbb{C}_s$.

2. Compute

$$F(\lambda)G_i^{\sim}(\lambda) = \left[\, F(\lambda)G_{i,1}^{\sim}(\lambda) \,\middle|\, F(\lambda)G_{i,2}^{\sim}(\lambda) \,\right] := \left[\, \widetilde{F}_1(\lambda) \,\middle|\, \widetilde{F}_2(\lambda) \,\right].$$

   and compute the stable solution $Y(\lambda)$ of the least distance problem

$$\gamma_{opt} := \left\| \left[\, \widetilde{F}_1(\lambda) - Y(\lambda)\ \widetilde{F}_2(\lambda) \,\right] \right\|_{\infty/2} = \min$$

   or of the γ-suboptimal distance problem

$$\left\| \left[\, \widetilde{F}_1(\lambda) - Y(\lambda)\ \widetilde{F}_2(\lambda) \,\right] \right\|_{\infty/2} \leq \gamma.$$

3. Compute an exact solution $X(\lambda)$ of the linear rational equation $X(\lambda)G_o(\lambda) = Y(\lambda)$, where $G_o(\lambda)$ is left invertible.

If `OPTIONS.mindeg = false`, the computed solution $X(\lambda)$ at Step 3 may not have the least achievable McMillan degree (excepting the case when $G_o(\lambda)$ is invertible), but its only unstable poles are (part of) the unstable zeros of $G_o(\lambda)$. These poles are also (part of) the zeros of $G(\lambda)$ lying in $\partial\mathbb{C}_s$. If $G_o(\lambda)$ is non-square, $X(\lambda)$ has a number of free poles, which are assigned to values specified via the option parameters `OPTIONS.sdeg` and `OPTIONS.poles`.

If `OPTIONS.mindeg = true`, the least order solution $X(\lambda)$ of the linear rational equation $X(\lambda)G_o(\lambda) = Y(\lambda)$ has the least achievable McMillan degree, but besides possible unstable poles which are the unstable zeros of $G_o(\lambda)$ lying in $\partial\mathbb{C}_s$, $X(\lambda)$ may also have additional unstable poles (e.g., in the case when $G_o(\lambda)$ has no least order stable left inverse). In this case, the solution corresponding to `OPTIONS.mindeg = false` is computed.

The resulting value of the achieved distance $\left\| \left[\widetilde{F}_1(\lambda) - Y(\lambda) \ \widetilde{F}_2(\lambda) \right] \right\|_{\infty/2}$ at Step 2, is returned in `INFO.mindist`. If `OPTIONS.H2sol = true`, an $\mathcal{H}_2$-LDP is solved (see Section 2.17), in which case the distance may be infinite if $\widetilde{F}_2(\lambda)$ is not strictly proper.

Example

Example 30. This example illustrates the computation of a stable left inverse for the example used in [67] (see also *Example* 7). The transfer function matrices $G(s)$ and $F(s)$ are

$$G(s) = \frac{1}{s^2+3s+2}\begin{bmatrix} s+1 & s+2 \\ s+3 & s^2+2s \\ s^2+3s & 0 \end{bmatrix}, \quad F(s) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The solution $X(s)$ of the rational equation $X(s)G(s) = I$ is a left inverse of $G(s)$. Since $G(s)$ has no zeros, a stable left inverse of $G(s)$ of McMillan degree equal to three exists. However, as illustrated in *Example* 7, any left inverse of least McMillan order equal to two is always unstable. The solution computed with `glasol` results always stable, even if the option for least order is selected. For the three free poles of the left inverse we can assign the poles to $\{-1,-2,-3\}$. For the computation of a stable left inverse with poles assigned to $\{-1,-2,-3\}$, the following MATLAB commands can be used:

```
% Wang and Davison Example (1973)
s = tf('s');
g = [ s+1 s+2; s+3 s^2+2*s; s^2+3*s 0 ]/(s^2+3*s+2); f = eye(2);
sysg = gir(ss(g)); sysf = ss(f);

% compute a stable solution of X(s)G(s) = I
[sysx,info] = glasol(sysg,sysf,struct('mindeg',true,'poles',[-1 -2 -3])); info
gpole(sysx)   % the left inverse is stable

% check solution
gir(sysx*sysg-sysf,1.e-7)
```

The computed solution is stable and the resulting minimal realization of the left inverse `sysx` has the poles equal to $\{-1,-2,-3\}$. ◊

Example 31. We solve the $\mathcal{H}_\infty$-MMP discussed in the book of Francis [12, Example 1, p. 112] with

$$G(s) = \left[\begin{array}{cc} -\dfrac{s-1}{s^2+s+1} & \dfrac{s(s-1)}{s^2+s+1} \end{array} \right] W(s), \quad F(s) = [\, W(s) \;\; 0\,],$$

where $W(s) = (s+1)/(10s+1)$ is a suitable weighting factor. The optimal solution

$$X(s) = \frac{2.3144(s+0.4569)(s+2.189)(s^2+s+1)}{(s+3.095)(s+2.189)(s+1)(s+0.4569)}$$

leads to the optimal error norm $\|F(\lambda) - X(\lambda)G(\lambda)\|_\infty = 0.2521$. The solution in Francis' book [12, p. 114] corresponds to a suboptimal solution for $\gamma = 0.2729$ and is

$$X_{sub}(s) = \frac{2.2731(s+0.4732)(s+2.183)(s^2+s+1)}{(s+3.108)(s+2.189)(s+1)(s+0.4569)}.$$

The corresponding suboptimal error norm $\|F(\lambda) - X_{sub}(\lambda)G(\lambda)\|_\infty = 0.2536$.

The solutions of the optimal and suboptimal $\mathcal{H}_\infty$-MMPs can be computed using the following MATLAB code:

```
s = tf('s');  % define the complex variable s
% enter W(s), G(s) and F(s)
W = (s+1)/(10*s+1);    % weighting function
G = [ -(s-1)/(s^2+s+1) (s^2-2*s)/(s^2+s+1)]*W;
F = [ W 0 ];

% solve G(s)*X(s) = F(s) for the least order solution
[X,info] = glasol(ss(G),ss(F),struct('mindeg',true,'tol',1.e-7)); info
minreal(zpk(X))

% compute the error norm of the optimal solution
norm(X*G-F,inf)


% compute the suboptimal solution for gamma = 0.2729
opts = struct('mindeg',true,'tol',1.e-7,'gamma',0.2729);
[Xsub,info] = glasol(ss(G),ss(F),opts); info
minreal(zpk(Xsub))

% compute the error norm of the suboptimal solution
norm(Xsub*G-F,inf)
```

♢

3.8 Functions for Matrix Pencils and Stabilization

These functions cover the reduction of linear matrix pencils to several Kronecker-like forms, the computation of specially ordered generalized real Schur forms, and the stabilization using state feedback.

3.8.1 gklf

Syntax

```
[AT,ET,INFO,Q,Z] = gklf(A,E,TOL,JOBOPT)
[AT,ET,INFO,Q,Z] = gklf(A,E,TOL,...,QOPT)
```

Description

`gklf` computes several Kronecker-like forms $\widetilde{A} - \lambda\widetilde{E}$ of a linear matrix pencil $A - \lambda E$.

Input data

`A,E` are the $m \times n$ real matrices A and E, which define the linear matrix pencil $A - \lambda E$.

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

`JOBOPT` is a character option variable to specify various options to compute Kronecker-like forms. The valid options are:

- `'standard'` – compute the standard Kronecker-like form (282), with infinite-finite ordering of the blocks of the regular part (see **Method**) (default);
- `'reverse'` – compute the standard Kronecker-like form (283), with reverse ordering of the blocks of the regular part (see **Method**);
- `'right'` – compute the Kronecker-like form (284), which exhibits the right and infinite Kronecker structures (see **Method**);
- `'left'` – compute the Kronecker-like form (285), which exhibits the left and infinite Kronecker structures (see **Method**).

`QOPT` is a character option variable to specify the options to accumulate or not the left orthogonal transformations in Q:

- `'Q'` – accumulate Q (default);
- `'noQ'` – do not accumulate Q.

Output data

`AT,ET` contain the matrices $\widetilde{A}$ and $\widetilde{E}$ which define the resulting pencil $\widetilde{A}-\lambda\widetilde{E}$ in a Kronecker-like form, satisfying

$$\widetilde{A} - \lambda\widetilde{E} = Q^T(A - \lambda E)Z, \tag{281}$$

where Q and Z are orthogonal transformation matrices.

`INFO` is a MATLAB structure, which provides information on the structure of the pencil $\widetilde{A}-\lambda\widetilde{E}$ (see **Method**) as follows:

`INFO.mr,INFO.nr` – the dimensions of the full row rank diagonal blocks of $[\, B_r \;\; A_r - \lambda E_r \,]$, which characterize the right Kronecker structure (see **Method**); `INFO.mr` and `INFO.nr` are empty if no right structure exists; if `JOBOPT = 'left'`, `INFO.mr` and `INFO.nr` are set to the negative values of the row and column dimensions of the pencil $A_{r,f} - \lambda E_{r,f}$ in (285);

`INFO.minf` – the dimensions of the square diagonal blocks of $A_\infty - \lambda E_\infty$, which characterize the infinite Kronecker structure; `INFO.minf` is empty if no infinite structure exists;

`INFO.mf` – the dimension of the square regular pencil $A_f - \lambda E_f$, which contains the finite eigenvalues;

`INFO.ml,INFO.nl` – the dimensions of the full column rank diagonal blocks of $\left[\begin{smallmatrix} A_l - \lambda E_l \\ C_l \end{smallmatrix}\right]$, which characterize the left Kronecker structure (see **Method**); `INFO.ml` and `INFO.nl` are empty if no left structure exists; if `JOBOPT = 'right'`, `INFO.ml` and `INFO.nl` are set to the negative values of the row and column dimensions of the pencil $A_{f,l} - \lambda E_{f,l}$ in (284).

`Q,Z` contain the orthogonal matrices Q and Z used to compute the Kronecker-like form $\widetilde{A} - \lambda \widetilde{E}$ in (281). Q and Z are not accumulated if both output variables `Q` and `Z` are not specified. If `QOPT = 'noQ'`, Q is not accumulated and, if specified, set to `Q = [ ]`.

Method

The Kronecker-like forms, obtainable by using orthogonal transformations, contains most of the relevant structural information provided by the potentially highly sensitive Kronecker canonical form (see Section 2.3). The computation of Kronecker-like forms is based on the method proposed in [3], which underlies the implementation of the mex-function `sl_klf`, called by `gklf`.

The (standard) Kronecker-like form has the following structure

$$\widetilde{A} - \lambda \widetilde{E} = \begin{bmatrix} B_r & A_r - \lambda E_r & * & * & * \\ 0 & 0 & A_\infty - \lambda E_\infty & * & * \\ 0 & 0 & 0 & A_f - \lambda E_f & * \\ 0 & 0 & 0 & 0 & A_l - \lambda E_l \\ 0 & 0 & 0 & 0 & C_l \end{bmatrix}, \tag{282}$$

where

(1) $[\, B_r \;\; A_r - \lambda E_r \,]$, with E_r invertible and upper triangular, contains the right Kronecker structure and is in a controllability staircase form with $[\, B_r \;\; A_r \,]$ as in (22) and E_r as in (23); the dimensions of the full row rank diagonal blocks of $[\, B_r \;\; A_r - \lambda E_r \,]$ are provided in `INFO.mr` and `INFO.nr`;

(2) $A_\infty - \lambda E_\infty$, with A_∞ invertible and upper triangular, and E_∞ nilpotent and upper triangular, contains the infinite structure and is in a block upper triangular form; the dimensions of the square diagonal blocks of $A_\infty - \lambda E_\infty$ are provided in `INFO.minf`;

(3) $A_f - \lambda E_f$, with E_f invertible, contains the finite structure; the dimension of the square regular pencil $A_f - \lambda E_f$ is provided in `INFO.mf`.

(4) $\left[\begin{smallmatrix} A_l-\lambda E_l \\ C_l \end{smallmatrix}\right]$, with E_l invertible and upper triangular, contains the left Kronecker structure and is in an observability staircase form with $\left[\begin{smallmatrix} A_l \\ C_l \end{smallmatrix}\right]$ as in (24) and E_l as in (25); the dimensions of the full column rank subdiagonal blocks of $\left[\begin{smallmatrix} A_l-\lambda E_l \\ C_l \end{smallmatrix}\right]$ are provided in `INFO.ml` and `INFO.nl`.

Depending on the option selected via the option parameter `JOBOPT`, several Kronecker-like forms can be determined, which contains basically the same blocks, however ordered differently, or contains only a part of the main structural blocks, which are relevant for particular applications (e.g., nullspace computation).

The following Kronecker-like forms can be computed:

1. `JOBOPT = 'standard'`: to determine the (standard) Kronecker-like form (282).

2. `JOBOPT = 'reverse'`: to determine a Kronecker-like form with a reversed order of the regular blocks

$$\widetilde{A}-\lambda\widetilde{E} = \begin{bmatrix} B_r & A_r-\lambda E_r & * & * & * \\ 0 & 0 & A_f-\lambda E_f & * & * \\ 0 & 0 & 0 & A_\infty-\lambda E_\infty & * \\ 0 & 0 & 0 & 0 & A_l-\lambda E_l \\ 0 & 0 & 0 & 0 & C_l \end{bmatrix}. \tag{283}$$

3. `JOBOPT = 'right'`: to determine a Kronecker-like form emphasizing the right and infinite structures

$$\widetilde{A}-\lambda\widetilde{E} = \begin{bmatrix} B_r & A_r-\lambda E_r & * & * \\ 0 & 0 & A_\infty-\lambda E_\infty & * \\ 0 & 0 & 0 & A_{f,l}-\lambda E_{f,l} \end{bmatrix}, \tag{284}$$

where $A_{f,l}-\lambda E_{f,l}$ contains the finite and left Kronecker structures. In this case, `INFO.ml` and `INFO.nl` are set to the negative values of the row and column dimensions of the pencil $A_{f,l}-\lambda E_{f,l}$, respectively, and `INFO.mf` $= 0$.

4. `JOBOPT = 'left'`: to determine a Kronecker-like form emphasizing the left and infinite structures

$$\widetilde{A}-\lambda\widetilde{E} = \begin{bmatrix} A_{r,f}-\lambda E_{r,f} & * & * \\ 0 & A_\infty-\lambda E_\infty & * \\ 0 & 0 & A_l-\lambda E_l \\ 0 & 0 & C_l \end{bmatrix}, \tag{285}$$

where $A_{r,f}-\lambda E_{r,f}$ contains the right and finite Kronecker structures. In this case, `INFO.mr` and `INFO.nr` are set to the negative values of the row and column dimensions of the pencil $A_{r,f}-\lambda E_{r,f}$, respectively, and `INFO.mf` $= 0$.

3.8.2 `gsklf`

Syntax

```
[AT,ET,DIMSC,Q,Z] = gsklf(SYS,TOL,ZEROSEL)
[AT,ET,DIMSC,Q,Z] = gsklf(SYS,TOL,ZEROSEL,OFFSET)
[AT,ET,DIMSC,Q,Z] = gsklf(SYS,TOL,...,QOPT)
```

Description

`gsklf` computes several special Kronecker-like forms of the system matrix pencil of a LTI descriptor system.

Input data

`SYS` is a LTI system, in a descriptor system state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{286}$$

`TOL` is a relative tolerance used for rank determinations. If `TOL` is not specified as input or if `TOL` $= 0$, an internally computed default value is used.

`ZEROSEL` is a character option variable to specify various zero selection options for the computation of the special Kronecker-like form (289) using various partitions of the form (288) of the extended complex plane (see **Method**). The valid options are:

`'none'`	–	use $\mathbb{C}_g = \mathbb{C} \cup \{\infty\}$ and $\mathbb{C}_b = \emptyset$ (default);
`'unstable'`	–	use $\mathbb{C}_g = \overline{\mathbb{C}}_s$ and $\mathbb{C}_b = \mathbb{C}_u$;
`'s-unstable'`	–	use $\mathbb{C}_g = \mathbb{C}_s$ and $\mathbb{C}_b = \overline{\mathbb{C}}_u$;
`'stable'`	–	use $\mathbb{C}_g = \overline{\mathbb{C}}_u$ and $\mathbb{C}_b = \mathbb{C}_s$;
`'all'`	–	use $\mathbb{C}_g = \emptyset$ and $\mathbb{C}_b = \mathbb{C} \cup \{\infty\}$;
`'finite'`	–	use $\mathbb{C}_g = \{\infty\}$ and $\mathbb{C}_b = \mathbb{C}$;
`'infinite'`	–	use $\mathbb{C}_g = \mathbb{C}$ and $\mathbb{C}_b = \{\infty\}$.

`OFFSET` is the stability boundary offset β, to be used to assess the finite zeros which belong to $\partial\mathbb{C}_s$ (the boundary of the stability domain) as follows: in the continuous-time case these are the finite zeros having real parts in the interval $[-\beta, \beta]$, while in the discrete-time case these are the finite zeros having moduli in the interval $[1-\beta, 1+\beta]$.
(Default: $\beta = 1.4901 \cdot 10^{-08}$).

`QOPT` is a character option variable to specify the options to accumulate or not the left orthogonal transformations in Q:

`'Q'`	–	accumulate Q (default);
`'noQ'`	–	do not accumulate Q.

Output data

`AT,ET` contain the matrices $\widetilde{A}$ and $\widetilde{E}$ which define the resulting system matrix pencil $\widetilde{A} - \lambda\widetilde{E}$ in a special Kronecker-like form (289), satisfying

$$\widetilde{A} - \lambda\widetilde{E} = \begin{bmatrix} Q^T & 0 \\ 0 & I_p \end{bmatrix} \left[\begin{array}{cc} A - \lambda E & B \\ \hline C & D \end{array}\right] Z, \tag{287}$$

where Q and Z are orthogonal transformation matrices.

`DIMSC` is a five-dimensional (integer) vector. `DIMSC(1:4)` contain the column dimensions of the matrices A_{rg}, $A_{b\ell}$, $B_{b\ell}$ and B_n, respectively, in the special Kronecker-like form (289). The column dimension `DIMSC(3)` of $B_{b\ell}$ represents the normal rank of the transfer function matrix $G(\lambda)$ of the system (286). `DIMSC(5)`, if non-negative, contains the number of finite zeros of `SYS` on the boundary of the stability region $\partial\mathbb{C}_s$, within the offset β specified by `OFFSET`. `DIMSC(6)`, if nonnegative, contains the number of infinite zeros of `SYS` in the continuous-time case and is set to 0 in the discrete-time case. Nonnegative values of `DIMSC(5)` and `DIMSC(6)` may only result for `OPTIONS.ZEROSEL` set to `'unstable'`, `'stable'` or `'s-unstable'`.

`Q,Z` contain the orthogonal matrices Q and Z used to compute the special Kronecker-like form $\widetilde{A}-\lambda\widetilde{E}$ in (289). Q and Z are not accumulated if both outputs `Q` and `Z` are not specified. If `QOPT = 'noQ'`, Q is not accumulated and, if specified, set to `Q = [ ]`.

Method

Consider a disjunct partition of the extended complex plane $\mathbb{C}_e = \mathbb{C} \cup \{\infty\}$ as

$$\mathbb{C}_e = \mathbb{C}_g \cup \mathbb{C}_b, \quad \mathbb{C}_g \cap \mathbb{C}_b = \emptyset\,, \tag{288}$$

where $\mathbb{C}_g$ and $\mathbb{C}_b$ are symmetric with respect to the real axis. Assume that the descriptor system realization (286) is infinite controllable (but $\mathbb{C}_b$-stabilizability is not assumed) and let $G(\lambda)$ be the $p \times m$ transfer function matrix of the system (286) having normal rank r. Then, there exist two orthogonal matrices Q and Z such that

$$\left[\begin{array}{cc} Q^T & 0 \\ 0 & I \end{array}\right] \left[\begin{array}{c|c} A-\lambda E & B \\ \hline C & D \end{array}\right] Z = \left[\begin{array}{cccc} A_{rg}-\lambda E_{rg} & * & * & * \\ 0 & A_{b\ell}-\lambda E_{b\ell} & B_{b\ell} & * \\ 0 & 0 & 0 & B_n \\ \hline 0 & & C_{b\ell} & D_{b\ell} \quad * \end{array}\right], \tag{289}$$

where

(a) The pencil $A_{rg}-\lambda E_{rg}$ has full row rank for $\lambda \in \mathbb{C}_g$ and E_{rg} has full row rank.

(b) $E_{b\ell}$ and B_n are invertible, the pencil

$$S_{bl}(\lambda) := \left[\begin{array}{cc} A_{b\ell}-\lambda E_{b\ell} & B_{b\ell} \\ C_{b\ell} & D_{b\ell} \end{array}\right] \tag{290}$$

has full column rank $n_{b\ell}+r$ in $\mathbb{C}_g$ (the pair $(A_{b\ell}-\lambda E_{b\ell}, B_{b\ell})$ is $\mathbb{C}_b$-stabilizable if the descriptor system realization (286) is $\mathbb{C}_b$-stabilizable).

The pencil $S_{bl}(\lambda)$ is the system matrix pencil of a descriptor system $(A_{b\ell}-\lambda E_{b\ell}, B_{b\ell}, C_{b\ell}, D_{b\ell})$, whose proper transfer function matrix $R(\lambda)$ is a range space basis matrix of $G(\lambda)$ [57]. By construction, the zeros of $R(\lambda)$ are those zeros of $G(\lambda)$ which are contained in $\mathbb{C}_b$, and can be selected via the option parameter `ZEROSEL`.

The computation of the special Kronecker-like form (289) is based on the method proposed in [29]. The function `gsklf` calls the more general function `gklf` to compute standard Kronecker-like forms, which is based on the mex-function `sl_klf`.

3.8.3 gsorsf

Syntax

```
[AT,ET,Q,Z,DIMS,NI] = gsorsf(A,E,OPTIONS)
```

Description

`gsorsf` computes, for a pair of square matrices (A, E), an orthogonally similar pair $(\widetilde{A}, \widetilde{E})$ in a specially ordered GRSF.

Input data

`A,E` are $n \times n$ real matrices A and E, which define a regular linear matrix pencil $A - \lambda E$.

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	**Description**
`tol`	tolerance for the singular values based rank determination of E (Default: $n^2\|E\|_1$`eps`)
`disc`	disc option for the "good" region $\mathbb{C}_g$: `true` – $\mathbb{C}_g$ is a disc centered in the origin; `false` – $\mathbb{C}_g$ is a left half complex plane (default).
`smarg`	stability margin β, which defines the stability region $\mathbb{C}_g$ of the eigenvalues of $A - \lambda E$, as follows: if `OPTIONS.disc = false`, the stable eigenvalues have real parts less than or equal to β, and if `OPTIONS.disc = true`, the stable eigenvalues have moduli less than or equal to β. (Default: `-sqrt(eps)` if `OPTIONS.disc = false`; `1-sqrt(eps)` if `OPTIONS.disc = true`.)
`reverse`	option for reverse ordering of diagonal blocks: `true` – the diagonal blocks are as in (294) (see **Method**); `false` – the diagonal blocks are as in (293) (see **Method**) (default).
`sepinf`	option for the separation of higher order infinite eigenvalues: `true` – separate higher order generalized infinite eigenvalues in the trailing positions if `OPTIONS.reverse = false`, or in the leading positions if `OPTIONS.reverse = true` (default); `false` – no separation of higher order infinite eigenvalues.
`fast`	option for fast separation of higher order infinite eigenvalues (to be used in conjuntion with the `OPTIONS.sepinf = true`): `true` – fast separation of higher order infinite eigenvalues using orthogonal pencil manipulation techniques with QR decompositon based rank determinations (default); `false` – separation of higher order infinite eigenvalues, by using SVD-based rank determinations (potentially more reliable, but slower).

Output data

`AT,ET` contain the transformed matrices $\widetilde{A}$ and $\widetilde{E}$, obtained as

$$\widetilde{A} = Q^T A Z, \qquad \widetilde{E} = Q^T E Z, \tag{291}$$

where Q and Z are orthogonal transformation matrices. The pair $(\widetilde{A}, \widetilde{E})$ is in a specially ordered GRSF, with the regular pencil $\widetilde{A} - \lambda\widetilde{E}$ in the form (293), if `OPTIONS.reverse = false`, or in the form (294), if `OPTIONS.reverse = true`.

`Q,Z` contain the orthogonal matrices Q and Z used to compute the pair $(\widetilde{A}, \widetilde{E})$ in (291), in a specially ordered GRSF.

`DIMS(1:4)` contain the orders of the diagonal blocks $(A_\infty, A_g, A_{b,f}, A_{b,\infty})$, if `OPTIONS.reverse = false`, or of the diagonal blocks $(A_{b,\infty}, A_{b,f}, A_g, A_\infty)$, if `OPTIONS.reverse = true` (see **Method**).

`NI` contains the dimensions of the square diagonal blocks of $A_{b,\infty} - \lambda E_{b,\infty}$, which characterize the higher order infinite eigenvalues of the pair (A, E). `NI` is empty if no higher order infinite eigenvalues exist.

Method

Consider a disjunct partition of the complex plane $\mathbb{C}$ as

$$\mathbb{C} = \mathbb{C}_g \cup \mathbb{C}_b, \quad \mathbb{C}_g \cap \mathbb{C}_b = \emptyset\,, \tag{292}$$

where both $\mathbb{C}_g$ and $\mathbb{C}_b$ are symmetrically located with respect to the real axis, and $\infty \in \mathbb{C}_b$. The complex domains $\mathbb{C}_g$ and $\mathbb{C}_b$ are typically associated with the "good" and "bad" generalized eigenvalues of the matrix pair (A, E), respectively. Using orthogonal similarity transformations as in (291), the pair (A, E) can be reduced to the specially ordered GRSF

$$\widetilde{A} - \lambda\widetilde{E} = \begin{bmatrix} A_\infty & * & * & * \\ 0 & A_g - \lambda E_g & * & * \\ 0 & 0 & A_{b,f} - \lambda E_{b,f} & * \\ 0 & 0 & 0 & A_{b,\infty} - \lambda E_{b,\infty} \end{bmatrix}, \tag{293}$$

where: (i) A_∞ is an $(n-r) \times (n-r)$ invertible (upper triangular) matrix, with $r = \operatorname{rank} E$; the leading pair $(A_\infty, 0)$ contains all infinite eigenvalues of $A - \lambda E$ corresponding to first-order eigenvectors; (ii) A_g and E_g are $n_g \times n_g$ matrices, such that the pair (A_g, E_g), with E_g invertible, is in a GRSF (i.e., A_g upper quasi-triangular and E_g upper triangular) and $\Lambda(A_g - \lambda E_g)$ are the finite eigenvalues lying in $\mathbb{C}_g$; (iii) $A_{b,f}$ and $E_{b,f}$ are $n_b^f \times n_b^f$ matrices, such that the pair $(A_{b,f}, E_{b,f})$, with $E_{b,f}$ invertible, is in a GRSF and $\Lambda(A_{b,f} - \lambda E_{b,f})$ are the finite eigenvalues lying in $\mathbb{C}_b$; and (iv) $A_{b,\infty}$ and $E_{b,\infty}$ are $n_b^\infty \times n_b^\infty$ upper triangular matrices, with $A_{b,\infty}$ invertible and $E_{b,\infty}$ nilpotent, and $\Lambda(A_{b,\infty} - \lambda E_{b,\infty})$ are the higher order infinite eigenvalues. The orders $[n-r, n_g, n_b^f, n_b^\infty]$ of the diagonal blocks $(A_\infty, A_g, A_{b,f}, A_{b,\infty})$, respectively, are provided in `DIMS(1:4)`. $A_{b,\infty} - \lambda E_{b,\infty}$ has a block upper triangular form with k diagonal blocks, whose increasingly ordered dimensions are provided in `NI`$(1{:}k)$. The dimensions `NI`$(1{:}k)$ define the multiplicities of the higher order infinite eigenvalues as follows: for $i = 1, \ldots, k$, there are

$\texttt{NI}(k-i+1)-\texttt{NI}(k-i)$ infinite eigenvalues of multiplicity i, where $\texttt{NI}(0) := 0$. The multiplicity of simple infinite eigenvalues is $\texttt{DIMS}(1)-\texttt{NI}(k)$.

To obtain the specially ordered GRSF (293), the **Procedure GSORSF**, described in [58], can be used. For the separation of the higher order infinite generalized eigenvalues, the mex-function `sl_klf` is called by `gsorsf`, if `OPTIONS.fast = true`. This function employs rank determinations based on QR-decompositions with column pivoting and, therefore, is more efficient than an alternative, potentially more reliable approach, based on SVD-based rank determination. This latter approach is performed if `OPTIONS.fast = false`.

The same procedure, applied to the transposed pair (A^T, E^T), is used to obtain the specially ordered GRSF with a reverse ordering of the blocks

$$\widetilde{A}-\lambda\widetilde{E} = \begin{bmatrix} A_{b,\infty}-\lambda E_{b,\infty} & * & * & * \\ 0 & A_{b,f}-\lambda E_{b,f} & * & * \\ 0 & 0 & A_g-\lambda E_g & * \\ 0 & 0 & 0 & A_\infty \end{bmatrix}. \tag{294}$$

The orders $[n_b^\infty, n_b^f, n_g, n-r]$ of the diagonal blocks $(A_{b,\infty}, A_{b,f}, A_g, A_\infty)$, respectively, are provided in `DIMS(1:4)`. $A_{b,\infty}-\lambda E_{b,\infty}$ has a block upper triangular form with k diagonal blocks, whose decreasingly ordered dimensions are provided in $\texttt{NI}(1{:}k)$. The dimensions $\texttt{NI}(1{:}k)$ define the multiplicities of the higher order infinite eigenvalues as follows: for $i = 1, \ldots, k$, there are $\texttt{NI}(i)-\texttt{NI}(i+1)$ infinite eigenvalues of multiplicity i, where $\texttt{NI}(k+1) := 0$. The multiplicity of simple infinite eigenvalues is $\texttt{DIMS}(4)-\texttt{NI}(1)$.

3.8.4 `gsfstab`

Syntax

```
[F,INFO] = gsfstab(A,E,B,POLES,SDEG,OPTIONS)
```

Description

`gsfstab` computes, for a descriptor system pair $(A-\lambda E, B)$, a state-feedback matrix F such that the controllable finite generalized eigenvalues of the closed-loop pair $(A+BF, E)$ lie in the stability domain $\mathbb{C}_s$.

Input data

`A,E` are $n \times n$ real matrices A and E, which define a regular linear pencil $A-\lambda E$. $E = I$ is assumed, for `E = [ ]`.

`B` is a $n \times m$ real matrix B.

`POLES` specifies a complex conjugated set of desired eigenvalues to be assigned for the pair $(A+BF, E)$. The specified values are intended to replace the finite generalized eigenvalues of (A, E) lying outside of the specified stability domain $\mathbb{C}_s$. If the number of specified eigenvalues in `POLES` is less than the number of controllable generalized eigenvalues of (A, E) outside of $\mathbb{C}_s$, then the rest of generalized eigenvalues of $(A+BF, E)$ are assigned to the nearest values on the boundary of $\mathbb{C}_s$ (defined by the parameter `SDEG`, see below) or are kept unmodified if `SDEG = [ ]`. (Default: `POLES = [ ]`)

`SDEG` specifies a prescribed stability degree for the eigenvalues of the pair $(A + BF, E)$. For a continuous-time setting, with `SDEG` < 0, the stability domain $\mathbb{C}_s$ is the set of complex numbers with real parts at most `SDEG`, while for a discrete-time setting, with $0 \leq$ `SDEG` < 1, $\mathbb{C}_s$ is the set of complex numbers with moduli at most `SDEG`
(Default: -0.2, used if `SDEG = [ ]` and `POLES = [ ]`).

`OPTIONS` is a MATLAB structure to specify user options and has the following fields:

`OPTIONS` fields	Description
`tol`	tolerance for rank determinations (Default: internally computed)
`sepinf`	option for a preliminary separation of the infinite eigenvalues: `true` – perform preliminary separation of the infinite generalized eigenvalues from the finite ones ; `false` – no separation of infinite generalized eigenvalues (default)

Output data

`F` contains the resulting $m \times n$ state-feedback gain F.

`INFO` is a MATLAB structure, which provides additional information on the closed-loop matrices $A_{cl} := Q(A + BF)Z$, $E_{cl} := QEZ$, and $B_{cl} := QB$, where Q and Z are the orthogonal matrices used to obtain the pair (A_{cl}, E_{cl}) in a GRSF. The fields of the INFO structure are:

`INFO` fields	Description
`Acl`	contains A_{cl} in a quasi-upper triangular form;
`Ecl`	contains E_{cl} in an upper triangular form;
`Bcl`	contains B_{cl} ;
`Q,Z`	contains the orthogonal transformation matrices Q and Z.
`ninf`	number of infinite generalized eigenvalues of (A, E);
`nfg`	number of finite generalized eigenvalues of (A, E) lying in the stability domain $\mathbb{C}_s$;
`naf`	number of assigned finite generalized eigenvalues in $\mathbb{C}_s$;
`nuf`	number of uncontrollable finite generalized eigenvalues lying outside $\mathbb{C}_s$.

Method

For a standard system pair (A, I) (for `E = [ ]`), the Schur method of [43] is used, while for a generalized system pair (A, E) the generalized Schur method of [47] is used. The resulting closed-loop matrices $A_{cl} := Q(A + BF)Z$, $E_{cl} := QEZ$, and $B_{cl} := QB$, where Q and Z are the orthogonal matrices used to obtain the pair (A_{cl}, E_{cl}) in a GRSF, have the forms

$$A_{cl} = \begin{bmatrix} A_\infty & * & * & * \\ 0 & A_{f,g} & * & * \\ 0 & 0 & A_{f,a} & * \\ 0 & 0 & 0 & A_{f,u} \end{bmatrix}, \quad E_{cl} = \begin{bmatrix} E_\infty & * & * & * \\ 0 & E_{f,g} & * & * \\ 0 & 0 & E_{f,a} & * \\ 0 & 0 & 0 & E_{f,u} \end{bmatrix}, \quad B_{cl} = \begin{bmatrix} * \\ * \\ * \\ 0 \end{bmatrix}, \tag{295}$$

where: (i) the pair (A_∞, E_∞), with A_∞ upper triangular and invertible and E_∞ upper triangular and nilpotent, contains the `INFO.ninf` infinite generalized eigenvalues of (A, E); (ii) the pair $(A_{f,g}, E_{f,g})$, in GRSF, contains the `INFO.nfg` finite generalized eigenvalues of (A, E) in $\mathbb{C}_s$; (iii) the pair $(A_{f,a}, E_{f,a})$, in GRSF, contains the `INFO.naf` assigned finite generalized eigenvalues in $\mathbb{C}_s$; and, (iv) the pair $(A_{f,u}, E_{f,u})$, in GRSF, contains the uncontrollable finite generalized eigenvalues of (A, E) lying outside $\mathbb{C}_s$. For the separation of the infinite generalized eigenvalues, the mex-function `sl_klf` is called by `gsfstab`.

A Installing DSTOOLS

DSTOOLS runs with MATLAB R2015b (or later versions) under 64-bit Windows 7 (or later). Additionally, the *Control System Toolbox* (Version 9.10 or later) is necessary to be installed. To install **DSTOOLS**, perform the following steps:

- download **DSTOOLS** as a zip file from Bitbucket[3]
- create on your computer the directory `dstools`
- extract, using any unzip utility, the functions of the **DSTOOLS** collection in the corresponding directory `dstools`
- start MATLAB and put the directory `dstools` on the MATLAB path, by using the `pathtool` command; for repeated use, save the new MATLAB search path, or alternatively, use the `addpath` command to set new path entries in `startup.m`
- try out the installation by running the demonstration script `DSToolsdemo.m`

[3] `https://bitbucket.org/DSVarga/dstools`

B Current `Contents.m` File

The M-functions available in the current version of **FDITOOLS** are listed in the current version of the `Contents.m` file, given below:

```
% DSTOOLS - Descriptor System Tools.
% Version 0.74            30-July-2019
% Copyright (c) 2016-2019 by A. Varga
%
% Demonstration.
%   DSToolsdemo - Demonstration of DSTOOLS.
%
% System analysis.
%   gpole     - Poles of a LTI descriptor system.
%   gzero     - Zeros of a LTI descriptor system.
%   gnrank    - Normal rank of the transfer function matrix of a LTI system.
%   ghanorm   - Hankel norm of a proper and stable LTI descriptor system.
%   gnugap    - Nu-gap distance between two LTI systems.
%
% System order reduction.
%   gir       - Reduced order realizations of LTI descriptor systems.
%   gminreal  - Minimal realization of a LTI descriptor system.
%   gbalmr    - Balancing-based model reduction of a LTI descriptor system.
%   gss2ss    - Conversions to SVD-like forms without non-dynamic modes.
%
% Operations on transfer function matrices.
%   grnull    - Right nullspace basis of a transfer function matrix.
%   glnull    - Left nullspace basis of a transfer function matrix.
%   grange    - Range space basis of a transfer function matrix.
%   gcrange   - Coimage space basis of a transfer function matrix.
%   grsol     - Solution of the linear rational matrix equation G*X = F.
%   glsol     - Solution of the linear rational matrix equation X*G = F.
%   ginv      - Generalized inverses.
%   gsdec     - Generalized additive spectral decompositions.
%   grmcover1 - Right minimum dynamic cover of Type 1 based order reduction.
%   glmcover1 - Left minimum dynamic cover of Type 1 based order reduction.
%   grmcover2 - Right minimum dynamic cover of Type 2 based order reduction.
%   glmcover2 - Left minimum dynamic cover of Type 2 based order reduction.
%   gbilin    - Generalized bilinear transformation.
%   gbilin1   - Transfer functions of commonly used bilinear transformations.
%
% Factorizations of transfer function matrices.
%   grcf      - Right coprime factorization with proper and stable factors.
%   glcf      - Left coprime factorization with proper and stable factors.
%   grcfid    - Right coprime factorization with inner denominator.
%   glcfid    - Left coprime factorization with inner denominator.
%   gnrcf     - Normalized right coprime factorization.
%   gnlcf     - Normalized left coprime factorization.
%   giofac    - Inner-outer/QR-like factorization.
%   goifac    - Co-outer-coinner/RQ-like factorization.
%   grsfg     - Right spectral factorization of gamma^2*I-G'*G.
%   glsfg     - Left spectral factorization of gamma^2*I-G*G'.
```

```
%
% Model-matching problem.
%   grasol    - Approximate solution of the linear rational matrix equation G*X = F.
%   glasol    - Approximate solution of the linear rational matrix equation X*G = F.
%   glinfldp  - Solution of the least distance problem min||G1-X G2||_inf.
%   gnehari   - Generalized Nehari approximation.
%
% Feedback stabilization.
%   gsfstab    - Generalized state-feedback stabilization.
%   eigselect1 - Selection of a real eigenvalue to be assigned.
%   eigselect2 - Selection of a pair of eigenvalues to be assigned.
%   galoc2     - Generalized pole allocation for second order systems.
%
% Pencil similarity transformations
%   gklf      - Kronecker-like staircase forms of a linear matrix pencil.
%   gsklf     - Special Kronecker-like form of a system matrix pencil.
%   gsorsf    - Specially ordered generalized real Schur form.
%
% SLICOT-based mex-functions.
%   sl_gstra  - Descriptor system coordinate transformations.
%   sl_gminr  - Minimal realization of descriptor systems.
%   sl_gsep   - Descriptor system additive spectral decompositions.
%   sl_gzero  - Computation of system zeros and Kronecker structure.
%   sl_klf    - Pencil reduction to Kronecker-like forms.
%   sl_glme   - Solution of generalized linear matrix equations.
```

C DSTOOLS Release Notes

The **DSTOOLS** Release Notes describe the changes introduced in the successive versions of the **DSTOOLS** collection, as new features, enhancements to functions, or major bug fixes. The following versions of **DSTOOLS** have been released:

Version	Release date	Comments
V0.5	December 31, 2016	Initial version accompanying the book [59].
V0.6	July 31, 2017	New function for range computation, many enhancements of existing functions.
V0.61	January 29, 2018	New function for coimage computation
V0.64	July 31, 2018	New functions for approximate solutions of linear rational equations.
V0.7	August 23, 2018	New functions for normalized coprime factorizations and bilinear transformation, and several enhancements of existing functions.
V0.71	September 30, 2018	New function for the computation of ν-gap metric, and several enhancements of existing functions related to reliably detecting poles and zeros on the stability domain boundary.
V0.74	July 30, 2019	New function for the computation of generalized inverses and enhancements in the evaluation of the normal rank.

C.1 Release Notes V0.5

This is the initial version of the **DSTOOLS** collection of M- and MEX-functions, which accompanies the book [59]. All numerical results presented in this book have been obtained using this version of **DSTOOLS**.

C.1.1 New Features

The M-functions and MEX-functions available in the Version 0.5 of **DSTOOLS** are listed below, where we kept the originally employed descriptions of the functions:

```
% DSTOOLS - Descriptor System Tools.
% Version 0.5             31-Dec-2016
% Copyright (c) 2016 by A. Varga
%
% Demonstration.
%   DSToolsdemo - Demonstration of DSTOOLS.
%
% System analysis.
%   gpole     - Poles of a LTI descriptor system.
%   gzero     - Invariant zeros and Kronecker structure of a system pencil.
%   nrank     - Normal rank of a transfer function matrix of a LTI system.
%   ghanorm   - Hankel norm of a proper and stable LTI descriptor system.
%
% Order reduction.
```

```
%   gir       - Irreducible realizations of LTI descriptor systems.
%   gminreal  - Minimal realization of a LTI descriptor system.
%   gbalmr    - Balancing-based model reduction of a stable descriptor system.
%   gss2ss    - Conversions to SVD-like forms without non-dynamic modes.
%
% Operations on generalized LTI systems.
%   glnull    - Left nullspace basis of a rational matrix.
%   grnull    - Right nullspace basis of a rational matrix.
%   glsol     - Solution of linear rational equation X(s)*G(s)=F(s).
%   grsol     - Solution of linear rational equation G(s)*X(s)=F(s).
%   gsdec     - Generalized additive spectral decompositions.
%   glmcover1 - Left minimum dynamic cover of Type 1 based order reduction.
%   grmcover1 - Right minimum dynamic cover of Type 1 based order reduction.
%   glmcover2 - Left minimum dynamic cover of Type 2 based order reduction.
%   grmcover2 - Right minimum dynamic cover of Type 2 based order reduction.
%

% Factorizations.
%   giofac    - Generalized inner-outer factorization of descriptor systems.
%   goifac    - Generalized co-outer-inner factorization of descriptor systems.
%   glcf      - Generalized left coprime factorization.
%   grcf      - Generalized right coprime factorization.
%   glcfid    - Generalized left coprime factorization with inner denominator.
%   grcfid    - Generalized right coprime factorization with inner denominator.
%   glsfg     - Generalized left spectral factorization of g^2-G*G'.
%   grsfg     - Generalized right spectral factorization of g^2-G'*G.
%
% Approximations.
%   gnehari   - Generalized Nehari approximation.
%   glinfldp  - Solution of the least distance problem min||F1-X F2||_inf.
%
% Feedback stabilization.
%   gsfstab    - Generalized state-feedback stabilization.
%   eigselect1 - Selection of a real eigenvalue to be assigned.
%   eigselect2 - Selection of a pair of eigenvalues to be assigned.
%   galoc2     - Generalized pole allocation for second order systems.
%
% Pencil similarity transformations
%   gklf      - Generalized Kronecker-like staircase form of a linear pencil.
%   gsorsf    - Specially ordered generalized real Schur form.
%
% SLICOT-based mex-functions.
%   sl_gstra  - Descriptor system coordinate transformations.
%   sl_gminr  - Minimal realization of descriptor systems.
%   sl_gsep   - Descriptor system spectral separations.
%   sl_gzero  - Computation of system zeros and Kronecker structure.
```

```
%   sl_klf    - Pencil reduction to Kronecker-like forms.
%   sl_glme   - Solution of generalized linear matrix equations.
```

C.2 Release Notes V0.6

This version of the **DSTOOLS** includes minor revisions of most functions, by adding exhaustive input parameter checks and performing several simplifications in the codes. Besides this, two new functions have been implemented and the underlying mex-functions have been replaced with new ones, which use an enhanced rank determination strategy.

C.2.1 New Features

Two new functions have been added to **DSTOOLS**:

```
% Operations on generalized LTI systems.
%   grange    - Range space basis of a transfer function matrix.
% Pencil similarity transformations
%   gsklf     - Special Kronecker-like form of a system matrix pencil.
```

The function `grange` to compute a range space basis of a transfer function matrix is based on a special Kronecker-like form of the system matrix pencil, which is computed by the function `gsklf`.

Several additional changes have been performed in the following functions:

`gpole`: the computations for standard state-space realizations have been separated from those for descriptor system realizations;

`nrank`: functionality extended to handle all LTI system objects (i.e., `ss`, `tf`, `zpk`);

`gir`:
- explicit handling of standard state-space representations added;
- option structure removed and replaced with a character string option parameter;

`gminreal`:
- explicit handling of standard state-space representations added;
- logical option parameter replaced with a character string option parameter;

`gbalmr`:
- original realization preserved if no order reduction and no balancing take place;
- checking the invertibility of E added;

`gss2ss`:
- relies entirely on `sl_gstra` to compute the SVD-like coordinate forms;
- exploits the generalized Hessenberg form of the pair (A, E) if E is invertible;

`grnull`: stabilization and pole assignment options added;

`glnull`: stabilization and pole assignment options added;

`grsol`: `INFO` provides always full structural information in `INFO.nr`, `INFO.nf` and `INFO.ninf`;

`glsol`: `INFO` provides always full structural information in `INFO.ninf`, `INFO.nf` and `INFO.nl`;

`gsdec`: explicit handling of standard state-space representations added;

`giofac`:
- – `OPTIONS` structure input parameter introduced to specify several user options;
- – new functionality added to compute the QR-like factorization of rational matrices;
- – the code sequence to compute a special Kronecker-like form replaced by a call of `gsklf`;

`goifac`:
- – `OPTIONS` structure input parameter introduced to specify several user options;
- – new functionality added to compute the RQ-like factorization of rational matrices;

`gnehari`: functionality restricted to systems without poles on the stability domain boundary;

`gsorsf`:
- – the output parameter `DIMS` contains now the orders of all four diagonal blocks;
- – the default option for `OPTIONS.sepinf` has been redefined;

C.2.2 Bug Fixes

Several minor bug fixes have been performed in the following functions:

`grnull`:
- – nullspace computation fixed for zero input dimensions;

`glnull`:
- – nullspace computation fixed for zero output dimensions;

C.3 Release Notes V0.61

This version of the **DSTOOLS** includes a new implemented function (`gcrange`), minor enhancements of two functions (`grange` and `gsklf`), and a few bug fixes.

C.3.1 New Features

A new function `gcrange` has been implemented for the computation of coimage space bases of transfer function matrices. This function relies on the new, enhanced versions of the functions `grange` and `gsklf`, with new options to handle "strictly unstable" system zeros.

C.3.2 Bug Fixes

Several minor bug fixes have been performed in the following functions:

`glinfldp`:
- – bug fixes in initializing the γ-iteration;

`gbalmr`:
- – bug fix to handle non-dynamic reduced systems;

`gnehari`:
- – bug fix to handle non-dynamic systems.

C.4 Release Notes V0.64

This version of the **DSTOOLS** includes two new functions `grasol` and `glasol`, to compute approximate solutions of linear rational equations, minor enhancements of the function `gir` and a few bug fixes.

C.4.1 New Features

Two new functions have been added to **DSTOOLS**:

```
% Model-matching problem.
%  grasol  - Approximate solution of the linear rational matrix equation G*X = F.
%  glasol  - Approximate solution of the linear rational matrix equation X*G = F.
```

The function `gir` has been extended to handle arbitrary two-dimensional arrays of state-space systems.

C.4.2 Bug Fixes

Several minor bug fixes have been performed in the following functions:

`gnehari`:
- extra argument added to specify a tolerance for rank computations.

`glinfldp`:
- bug fix when early terminating the γ-iteration without performing any iteration;
- an explicit minimum value for a suboptimal γ value is provided in the error message;
- tolerance added in several calls to the function `gnehari`

C.5 Release Notes V0.7

This version of the **DSTOOLS** includes four new functions, several enhancements of existing functions and a few bug fixes.

C.5.1 New Features

The following new functions have been added to **DSTOOLS**:

`gnrcf`	–	generalized normalized right coprime factorization;
`gnlcf`	–	generalized normalized left coprime factorization;
`gbilin`	–	generalized bilinear transformations;
`gbilin1`	–	generation of first order transfer functions which describe several commonly used bilinear transformations.

The functions `grnull` and `glnull` have been enhanced to optionally compute inner, respectively, coinner nullspace bases.

The functions `grange` and `gcrange` provide additional information in an INFO structure, as the normal rank of the transfer function matrix, the number of its zeros lying on the boundary of stability region, and the accuracy of the solution of the involved Riccati equation. `INFO.nrank`

is now provided in both functions instead `INFO.rankG`.

In the functions `grsol` and `glsol`, `INFO.nrank` is now provided instead `INFO.rankG`.

The functions `giofac` and `goifac` provide additional information in an INFO structure, as the normal rank of the transfer function matrix, the number of its zeros lying on the boundary of stability region, and the accuracy of the solution of the involved Riccati equation.

The functions `grasol` and `glasol` use the new information provided by `giofac` and `goifac`, to detect nonstandard problems without performing additional computations. `INFO.nrank` is now provided in both functions instead `INFO.rankG`.

The function `gsklf` provides additionally the number of finite zeros on the boundary of the stability region and the number of infinite zeros in the continuous-time case.

C.5.2 Bug Fixes

Several minor bug fixes have been performed in the following functions:

`gcrange`:
- – option `inner` changed in `coinner`;

`glasol`:
- – minor enhancements to properly work with empty systems;

`grasol`:
- – minor enhancements to properly work with empty systems;

`glinfldp`:
- – minor bug fixes to enhance termination issues of the γ-iteration;

C.6 Release Notes V0.71

This version of the **DSTOOLS** includes a new function to evaluate the ν-gap metric, several enhancements of existing functions related to reliably detecting poles and zeros on the stability domain boundary and a few bug fixes.

C.6.1 New Features

A new function `gnugap` has been implemented to compute the ν-gap metric for LTI systems. This function is applicable to arbitrary (even improper) systems and relies on the new enhancements of the functions `gpole` and `gzero`.

The functions `gpole` and `gzero` have been substantially enhanced, by providing extensive additional information on the structure of the underlying pencils and properties of computed poles and zeros, respectively. An additional optional entry can be used to specify an offset for the stability domain boundary.

The function `gnrank` replaces `nrank`. For compatibility purposes, the obsolete function `nrank` can be still used, but it will be removed in a future version of `DSTOOLS`.

The functions `grasol` and `glasol` have an additional option entry, to specify an offset for the stability domain boundary.

The functions `grange` and `gcrange` have an additional option entry, to specify an offset for the stability domain boundary.

The functions `giofac` and `goifac` have an additional option entry, to specify an offset for the stability domain boundary.

The function `gsklf` has an additional input to specify an offset for the stability domain boundary. For compatibility purposes, the old calling sequence used up to version V0.7 can be still used.

C.6.2 Bug Fixes

Several bug fixes and minor improvements have been performed in the following functions:

`gpole`:
- bug fixed in determining the number of eigenvalues to be set to `NaN` values;

`gir`:
- improved performance in handling standard systems;

`grnull`:
- stabilizability test added when computing inner basis;

`glnull`:
- detectability test added when computing coinner basis;

`grange`:
- controllability is enforced by eliminating all uncontrollable eigenvalues;

`giofac`:
- controllability is enforced by eliminating all uncontrollable eigenvalues;

`gsklf`:
- bug fixed in evaluating the number of unstable finite zeros;
- bug fixed in detecting the lack of impulse controllability;

`grasol`:
- minor bug fixed in handling sampling-time compatibility related issues;

`glasol`:
- minor bug fixed in handling sampling-time compatibility related issues;

C.7 Release Notes V0.74

This version of the **DSTOOLS** includes a new function to compute several generalized inverses of LTI systems, an enhanced version of the function for normal rank evaluation, as well as a few bug fixes.

C.7.1 New Features

A new function `ginv` has been implemented to compute several system theory relevant generalized inverses of LTI systems.

An enhancement of the function `gnrank` allows to reliably check null normal rank conditions on transfer function matrices using the techniques described in [60].

C.7.2 Bug Fixes

The handling of errors issued by the Riccati equation solvers have been enhanced in the functions `giofac`, `grange`, `glnull`, and `grnull`.

The usage of the Frobenius-norm for condition number estimation and evaluation of state-feedback/feedforward gains has been enforced in the functions `grmcover1`, `grmcover2`, `glmcover1`, `glmcover2`, `grsol` and `glsol`.

A minor bug has been fixed in `grsol`.

References

[1] A. C. Antoulas. New results on the algebraic theory of linear systems: the solution of the cover problems. *Linear Algebra Appl.*, 505:1–43, 1983.

[2] T. Beelen. *New algorithms for computing the Kronecker structure of a pencil with applications to systems and control theory.* Ph. D. Thesis, Eindhoven University of Technology, 1987.

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

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

[5] S. L. Campbell. *Singular Systems of Differential Equations.* Pitman, London, 1980.

[6] C.-C. Chu, J. C. Doyle, and E. B. Lee. The general distance problem in H_∞ optimal control theory. *Int. J. Control*, 44:565–596, 1986.

[7] L. Dai. *Singular Control Systems*, volume 118 of *Lecture Notes in Control and Information Sciences.* Springer Verlag, New York, 1989.

[8] J. Demmel and B. Kågström. The generalized Schur decomposition of an arbitrary pencil $A-\lambda B$: robust software with error bounds and applications. Part I: Theory and algorithms. Part II: Software and applications. *ACM Trans. Math. Software*, 19:160–174, 175–201, 1993.

[9] G.-R. Duan. *Analysis and Design of Descriptor Linear Systems*, volume 23 of *Advances in Mechanics and Mathematics.* Springer, New York, 2010.

[10] A. Emami-Naeini and P. M. Van Dooren. Computation of zeros of linear multivariable systems. *Automatica*, 18:415–430, 1982.

[11] E. Emre, L. M. Silverman, and K. Glover. Generalized dynamic covers for linear systems with applications to deterministic identification and realization problems. *IEEE Trans. Automat. Control*, 22:26–35, 1977.

[12] B. A. Francis. *A Course in H^∞ Theory*, volume 88 of *Lecture Notes in Control and Information Sciences.* Springer-Verlag, New York, 1987.

[13] F. R. Gantmacher. *Theory of Matrices*, volume 2. Chelsea, New York, 1959.

[14] Z. Gao and P. J. Antsaklis. On stable solutions of the one- and two-sided model matching problems. *IEEE Trans. Automat. Control*, 34:978–982, 1989.

[15] K. Glover. All optimal Hankel-norm approximations of linear multivariable systems and their L^∞-error bounds. *Int. J. Control*, 39:1115–1193, 1984.

[16] G. H. Golub and C. F. Van Loan. *Matrix Computations, 4th Edition.* John Hopkins University Press, Baltimore, 2013.

[17] D. W. Gu, M. C. Tsai, S. D. O'Young, and I. Postlethwaite. State-space formulae for discrete-time H^∞ optimization. *Int. J. Control*, 49:1683–1723, 1989.

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

[19] N. J. Higham. *Functions of Matrices Theory and Computation.* SIAM, Philadelphia, 2008.

[20] R. A. Horn and C. R. Johnson. *Matrix Analysis, 2nd Edition.* Cambridge University Press, New York, 2013.

[21] B. Kågström and P. Van Dooren. Additive decomposition of a transfer function with respect to a specified region. In *Signal Processing, Scattering and Operator theory, and Numerical Methods, Proceedings of the International Symposium on Mathematical Theory of Networks and Systems 1989, Amsterdam, The Netherlands*, M. A. Kaashoek, J. H. van Schuppen, and A. C. M. Ran, editors, volume 3, Birhhäuser, Boston, 1990, pages 469–477.

[22] T. Kailath. *Linear Systems.* Prentice Hall, Englewood Cliffs, 1980.

[23] G. Kimura. Geometric structure of observers for linear feedback control laws. *IEEE Trans. Automat. Control*, 22:846–855, 1977.

[24] P. Kunkel and V. Mehrmann. *Differential-Algebraic Equations. Analysis and Numerical Solution.* EMS Publishing House, Zurich, 2006.

[25] MathWorks. *Control System Toolbox (R2015b), User's Guide.* The MathWorks Inc., Natick, MA, 2015.

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

[27] A. S. Morse. Minimal solutions to transfer matrix equations. *IEEE Trans. Automat. Control*, 21:131–133, 1976.

[28] C. Oară. On computing normalized-coprime factorizations of general rational matrices. *IEEE Trans. Automat. Control*, 46:286–290, 2001.

[29] C. Oară. Constructive solutions to spectral and inner-outer factorizations with respect to the disk. *Automatica*, 41:1855–1866, 2005.

[30] C. Oară and P. Van Dooren. An improved algorithm for the computation of structural invariants of a system pencil and related geometric aspects. *Syst. Control Lett.*, 30:39–48, 1997.

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

[32] C. Oară and A. Varga. Computation of general inner-outer and spectral factorizations. *IEEE Trans. Automat. Control*, 45:2307–2325, 2000.

[33] T. Penzl. Numerical solution of generalized Lyapunov equations. *Adv. Comput. Math.*, 8: 33–48, 1998.

[34] M. G. Safonov, R. Y. Chiang, and D. J. N. Limebeer. Optimal Hankel model reduction for nonminimal systems. *IEEE Trans. Automat. Control*, 35:496–502, 1990.

[35] T. Stykel. Gramian based model reduction for descriptor systems. *Math. Control Signals Syst.*, 16:297–319, 2004.

[36] F. Svaricek. Computation of the structural invariants of linear multivariable systems with an extended version of the program ZEROS. *Syst. Control Lett.*, 6:261–266, 1985.

[37] M. S. Tombs and I. Postlethwaite. Truncated balanced realization of a stable non-minimal state-space system. *Int. J. Control*, 46:1319–1330, 1987.

[38] P. Van Dooren. The computation of Kronecker's canonical form of a singular pencil. *Linear Algebra Appl.*, 27:103–141, 1979.

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

[40] P. Van Dooren. Rational and polynomial matrix factorizations via recursive pole-zero cancellation. *Linear Algebra Appl.*, 137/138:663–697, 1990.

[41] P. Van Dooren. Numerical Linear Algebra for Signals, Systems and Control. University of Louvain, Louvain la Neuve, Belgium, Draft notes prepared for the Graduate School in Systems and Control, 2003. `http://perso.uclouvain.be/paul.vandooren/PVDnotes.pdf`.

[42] A. I. G. Vardulakis and N. Karcanias. Proper and stable, minimal MacMillan degrees bases of rational vector spaces. *IEEE Trans. Automat. Control*, 29:1118–1120, 1984.

[43] A. Varga. A Schur method for pole assignment. *IEEE Trans. Automat. Control*, 26:517–519, 1981.

[44] A. Varga. Computation of irreducible generalized state-space realizations. *Kybernetika*, 26: 89–106, 1990.

[45] A. Varga. Efficient minimal realization procedure based on balancing. In *Prepr. of IMACS Symp. on Modelling and Control of Technological Systems*, A. El Moudni, P. Borne, and S. G. Tzafestas, editors, volume 2, 1991, pages 42–47.

[46] A. Varga. Minimal realization procedures based on balancing and related techniques. In *Computer Aided Systems Theory – EUROCAST'91, A Selection of Papers from the Proceedings of the Second International Workshop on Computer Aided System Theory, Krems, Austria*, F. Pichler and R. Moreno Díaz, editors, volume 585 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, 1992, pages 733–761.

[47] A. Varga. On stabilization of descriptor systems. *Syst. Control Lett.*, 24:133–138, 1995.

[48] A. Varga. Computation of Kronecker-like forms of a system pencil: Applications, algorithms and software. In *Proceedings of the IEEE International Symposium on Computer-Aided Control System Design, Dearborn, MI, USA*, 1996, pages 77–82.

[49] A. Varga. Computation of coprime factorizations of rational matrices. *Linear Algebra Appl.*, 271:83–115, 1998.

[50] A. Varga. Computing generalized inverse systems using matrix pencil methods. *Int. J. of Applied Mathematics and Computer Science*, 11:1055–1068, 2001.

[51] A. Varga. On computing least order fault detectors using rational nullspace bases. In *Proceedings of the IFAC Symposium SAFEPROCESS, Washington D.C., USA*, 2003.

[52] A. Varga. Reliable algorithms for computing minimal dynamic covers. In *Proceedings of the IEEE Conference on Decision and Control, Maui, HI, USA*, 2003, pages 1873–1878.

[53] A. Varga. Computation of least order solutions of linear rational equations. In *Proceedings of the International Symposium on Mathematical Theory of Networks and Systems, Leuven, Belgium*, 2004.

[54] A. Varga. Reliable algorithms for computing minimal dynamic covers for descriptor systems. In *Proceedings of the International Symposium on Mathematical Theory of Networks and Systems, Leuven, Belgium*, 2004.

[55] A. Varga. On computing nullspace bases – a fault detection perspective. In *Proceedings of the IFAC World Congress, Seoul, Korea*, 2008, pages 6295–6300.

[56] A. Varga. On computing minimal proper nullspace bases with applications in fault detection. In *Numerical Linear Algebra in Signals, Systems and Control*, P. Van Dooren, S. P. Bhattacharyya, R. H. Chan, V. Olshevsky, and A. Routray, editors, volume 80 of *Lecture Notes in Electrical Engineering*, Springer Verlag, Berlin, 2011, pages 433–465.

[57] A. Varga. A note on computing range space bases of rational matrices, 2017. `https://arxiv.org/abs/1707.00489`.

[58] A. Varga. On recursive computation of coprime factorizations of rational matrices, 2017. `https://arxiv.org/abs/1703.07307`.

[59] A. Varga. *Solving Fault Diagnosis Problems – Linear Synthesis Techniques*, volume 84 of *Studies in Systems, Decision and Control*. Springer International Publishing, 2017.

[60] A. Varga. On checking null rank conditions of rational matrices , 2018. `https://arxiv.org/abs/1812.11396`.

[61] G. Verghese, B. Lévy, and T. Kailath. A generalized state-space for singular systems. *IEEE Trans. Automat. Control*, 26:811–831, 1981.

[62] 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.

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

[64] M. Vidyasagar. *Control System Synthesis: A Factorization Approach*. "Morgan & Claypool", 2011.

[65] G. Vinnicombe. Frequency domain uncertainty and the graph topology. *IEEE Trans. Automat. Control*, 38:1371–1383, 1993.

[66] G. Vinnicombe. *Uncertainty and Feedback: $\mathcal{H}_\infty$ Loop-shaping and the ν-gap Metric.* Imperial College Press, London, 2001.

[67] S.-H. Wang and E. J. Davison. A minimization algorithm for the design of linear multivariable systems. *IEEE Trans. Automat. Control*, 18:220–225, 1973.

[68] J. H. Wilkinson. The perfidious polynomial. In *Studies in Numerical Analysis*, G. H. Golub, editor, volume 24 of *Studies in Mathematics*, Mathematical Association of America, 1984, pages 1–28.

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

Index