Fault Detection and Isolation Tools (FDITOOLS)

User's Guide

Andreas Varga[*]

July 7, 2019

Abstract

The Fault Detection and Isolation Tools (**FDITOOLS**) is a collection of MATLAB functions for the analysis and solution of fault detection and model detection problems. The implemented functions are based on the computational procedures described in the Chapters 5, 6 and 7 of the book: "A. Varga, Solving Fault Diagnosis Problems – Linear Synthesis Techniques, Springer, 2017". This document is the User's Guide for the version V1.0.5 of **FDITOOLS**. First, we present the mathematical background for solving several basic exact and approximate synthesis problems of fault detection filters and model detection filters. Then, we give in-depth information on the command syntax of the main analysis and synthesis functions. Several examples illustrate the use of the main functions of **FDITOOLS**.

[*]Andreas Varga lives in Gilching, Germany. *E-mail address*: `varga.andreas@gmail.com`, *URL*: `https://sites.google.com/site/andreasvargacontact/`

Contents

Notations and Symbols

General notations

$\mathbb{C}$	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
$\mathbb{R}(\lambda)$	set of rational matrices in indeterminate λ with real coefficients
$\mathbb{R}(\lambda)^{p\times m}$	set of $p \times m$ rational 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
$\mathcal{L}_\infty$	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 $G(\lambda)$ or 2-norm of a matrix G
$\|G\|_\infty$	$\mathcal{H}_\infty$- 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 $G(\lambda)$
$\|G\|_{\infty-}$	$\mathcal{H}_{\infty-}$-index of the transfer function matrix $G(\lambda)$
$\|G\|_{\Omega-}$	$\mathcal{H}_{-}$-index over a frequency domain Ω 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 $G_2(\lambda)$
M^T	transpose of the matrix M
M^{-1}	inverse of the matrix M
M^{-L}	left 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
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

Fault diagnosis related notations

$y(t)$	measured output vector: $y(t) \in \mathbb{R}^p$
$\mathbf{y}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed measured output vector
$u(t)$	control input vector: $u(t) \in \mathbb{R}^{m_u}$
$\mathbf{u}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed control input vector
$d(t)$	disturbance input vector: $d(t) \in \mathbb{R}^{m_d}$
$\mathbf{d}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed disturbance input vector
$w(t)$	noise input vector: $w(t) \in \mathbb{R}^{m_w}$
$\mathbf{w}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed noise input vector
$f(t)$	fault input vector: $f(t) \in \mathbb{R}^{m_f}$
$\mathbf{f}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed fault input vector
$x(t)$	state vector: $x(t) \in \mathbb{R}^n$
$G_u(\lambda)$	transfer function matrix from u to y
$G_d(\lambda)$	transfer function matrix from d to y
$G_w(\lambda)$	transfer function matrix from w to y
$G_f(\lambda)$	transfer function matrix from f to y
$G_{f_j}(\lambda)$	transfer function matrix from the j-th fault input f_j to y
A	system state matrix
E	system descriptor matrix
B_u, B_d, B_w, B_f	system input matrices from u, d, w, f
C	system output matrix
D_u, D_d, D_w, D_f	system feedthrough matrices from u, d, w, f
$r(t)$	residual vector: $r(t) \in \mathbb{R}^q$
$\mathbf{r}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed residual vector
n_b	number of components of residual vector r
$r^{(i)}(t)$	i-th residual vector component: $r^{(i)}(t) \in \mathbb{R}^{q_i}$
$\mathbf{r}^{(i)}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed i-th residual vector component
$Q(\lambda)$	transfer function matrix of the implementation form of the residual generator from y and u to r
$Q_y(\lambda)$	transfer function matrix of residual generator from y to r
$Q_u(\lambda)$	transfer function matrix of residual generator from u to r
$Q^{(i)}(\lambda)$	transfer function matrix of the implementation form of the i-th residual generator from y and u to $r^{(i)}$
$R(\lambda)$	transfer function matrix of the internal form of the residual generator from u, d, w and f to r
$R_u(\lambda)$	transfer function matrix from u to r
$R_d(\lambda)$	transfer function matrix from d to r
$R_w(\lambda)$	transfer function matrix from w to r
$R_f(\lambda)$	transfer function matrix from f to r
$R_{f_j}(\lambda)$	transfer function matrix from the j-th fault input f_j to r
$R^{(i)}_{f_j}(\lambda)$	transfer function matrix from the j-th fault input f_j to $r^{(i)}$
S	binary structure matrix
S_{R_f}	binary structure matrix corresponding to $R_f(\lambda)$

$M_r(\lambda)$	transfer function matrix of a reference model from f to r
$\theta(t)$	residual evaluation vector
$\iota(t)$	binary decision vector
τ, τ_i	decision thresholds

Model detection related notations

N	number of component models of the multiple model
$y(t)$	measured output vector: $y(t) \in \mathbb{R}^p$
$\mathbf{y}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed measured output vector
$u(t)$	control input vector: $u(t) \in \mathbb{R}^{m_u}$
$\mathbf{u}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed control input vector
$u^{(j)}(t)$	control input vector of j-th model: $u^{(j)}(t) := u(t) \in \mathbb{R}^{m_u}$
$\mathbf{u}^{(j)}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed control input vector of j-th model
$d^{(j)}(t)$	disturbance input vector of j-th model: $d^{(j)}(t) \in \mathbb{R}^{m_d^{(j)}}$
$\mathbf{d}^{(j)}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed disturbance input vector of j-th model
$w^{(j)}(t)$	noise input vector of j-th model: $w^{(j)}(t) \in \mathbb{R}^{m_w^{(j)}}$
$\mathbf{w}^{(j)}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed noise input vector of j-th model
$y^{(j)}(t)$	output vector of j-th model: $y^{(j)}(t) \in \mathbb{R}^p$
$\mathbf{y}^{(j)}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed output vector of j-th model
$x^{(j)}(t)$	state vector of j-th model: $x^{(j)}(t) \in \mathbb{R}^{n_i}$
$G_u^{(j)}(\lambda)$	transfer function matrix of j-th model from $u^{(j)}$ to $y^{(j)}$
$G_d^{(j)}(\lambda)$	transfer function matrix of j-th model from $d^{(j)}$ to $y^{(j)}$
$G_w^{(j)}(\lambda)$	transfer function matrix of j-th model from $w^{(j)}$ to $y^{(j)}$
$A^{(j)}$	system state matrix of j-th model
$E^{(j)}$	system descriptor matrix of j-th model
$B_u^{(j)}$, $B_d^{(j)}$, $B_w^{(j)}$	system input matrices of j-th model from $u^{(j)}$, $d^{(j)}$, $w^{(j)}$
$C^{(j)}$	system output matrix of j-th model
$D_u^{(j)}$, $D_d^{(j)}$, $D_w^{(j)}$	system feedthrough matrices of j-th model from $u^{(j)}$, $d^{(j)}$, $w^{(j)}$
$r^{(i)}(t)$	i-th residual vector component: $r^{(i)}(t) \in \mathbb{R}^{q_i}$
$\mathbf{r}^{(i)}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed i-th residual vector component
$r(t)$	overall residual vector: $r(t) \in \mathbb{R}^q$, $q = \sum_{i=1}^N q_i$
$\mathbf{r}(\lambda)$	Laplace- or $\mathcal{Z}$-transformed overall residual vector
$Q^{(i)}(\lambda)$	transfer function matrix of the implementation form of the i-th residual generator from y and u to $r^{(i)}$
$Q_y^{(i)}(\lambda)$	transfer function matrix of residual generator from y to $r^{(i)}$
$Q_u^{(i)}(\lambda)$	transfer function matrix of residual generator from u to $r^{(i)}$
$Q(\lambda)$	transfer function matrix of the implementation form of the overall residual generator from y and u to r
$R^{(i,j)}(\lambda)$	the transfer function matrix of the internal form of the overall residual generator from $(u^{(j)}, d^{(j)}, w^{(j)})$ to $r^{(i)}$

$R_u^{(i,j)}(\lambda)$	the transfer function matrix of the internal form of the overall residual generator from $u^{(j)}$ to $r^{(i)}$
$R_d^{(i,j)}(\lambda)$	the transfer function matrix of the internal form of the overall residual generator from $d^{(j)}$ to $r^{(i)}$
$R_w^{(i,j)}(\lambda)$	the transfer function matrix of the internal form of the overall residual generator from $w^{(j)}$ to $r^{(i)}$
$\theta(t)$	N-dimensional residual evaluation vector
$\iota(t)$	N-dimensional binary decision vector
τ_i	decision threshold for i-th component of the residual vector

Acronyms

AFDP	Approximate fault detection problem
AFDIP	Approximate fault detection and isolation problem
AMDP	Approximate model detection problem
AMMP	Approximate model matching problem
EFDP	Exact fault detection problem
EFEP	Exact fault estimation problem
EFDIP	Exact fault detection and isolation problem
EMDP	Exact model detection problem
EMMP	Exact model matching problem
FDD	Fault detection and diagnosis
FDI	Fault detection and isolation
LTI	Linear time-invariant
LFT	Linear fractional transformation
LPV	Linear parameter-varying
MIMO	Multiple-input multiple-output
MMP	Model-matching problem
TFM	Transfer function matrix

1 Introduction

The Fault Detection and Isolation Tools (**FDITOOLS**) is a collection of MATLAB functions for the analysis and solution of fault detection problems. **FDITOOLS** supports various synthesis approaches of linear residual generation filters for continuous- or discrete-time linear systems. The underlying synthesis techniques rely on reliable numerical algorithms developed by the author and described in the Chapters 5, 6 and 7 of the author's book [16]:

> 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 **FDITOOLS** collection rely on the *Control System Toolbox* [2] and the *Descriptor System Tools* (**DSTOOLS**) V0.71 [4]. The current release of **FDITOOLS** is version V1.0.5, dated July 7, 2019. **FDITOOLS** 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 2018b. To use the functions of **FDITOOLS**, the *Control System Toolbox* and the **DSTOOLS** collection must be installed in MATLAB running under 64-bit Windows 7, 8, 8.1 or 10.

This document describes version V1.0.5 of the **FDITOOLS** collection. This version covers all synthesis procedures described in the book [16] and, additionally, includes a comprehensive collection of analysis functions, as well as functions for an easy setup of synthesis models. The book [16] represents an important complementary documentation for the **FDITOOLS** collection: it describes the mathematical background of solving synthesis problems of fault detection and model detection filters and gives detailed descriptions of the underlying synthesis 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 **FDITOOLS** as follows:

> A. Varga. FDITOOLS – The Fault Detection and Isolation Tools for MATLAB, 2018. `https://sites.google.com/site/andreasvargacontact/home/software/fditools`.

The implementation of the functions included in the **FDITOOLS** collection follows several principles, which have been consequently enforced when implementing these functions. These principles are listed below and partly consists of the requirements for robust software implementation, but also include several requirements which are specific to the field of fault detection:

- *Using general, numerically reliable and computationally efficient numerical approaches as basis for the implementation of all computational functions, to guarantee the solvability of problems under the most general existence conditions of the solutions.* Consequently, the implemented methods provide a solution whenever a solution exists. These methods are extensively described in the book [16], which forms the methodological and computational basis of all implemented analysis and synthesis functions.

- *Support for the most general model representation of linear time-invariant systems in form of generalized state-space representation, also known as descriptor systems.* All analysis

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

and synthesis functions are applicable to both continuous- and discrete-time systems. The basis for implementation of all functions is the *Descriptor System Tools* (**DSTOOLS**) [4], a collection of functions to handle rational transfer function matrices (proper or improper), via their equivalent descriptor system representations. The initial version of this collection has been implemented in conjunction with the book [16].

- *Providing simple user interface to all synthesis functions.* All functions rely on default settings of problem parameters and synthesis options, which allow to easily obtain preliminary synthesis results. Also, all functions to solve a class of problems (e.g., fault detection), are applicable to the same input models. Therefore, the synthesis functions to solve approximate synthesis problems are applicable to solve the exact synthesis problems as well. On the other side, the solution of an exact problem for a system with noise inputs, represents a first approximation to the solution of the approximate synthesis problem.

- *Providing an exhaustive set of options to ensure the complete freedom in choosing problem specific parameter and synthesis options.* Among the frequently used synthesis options are: the number of residual signal outputs or the numbers of outputs of the components of structured residual signals; stability degree for the poles of the resulting filters or the location of their poles; frequency values to enforce strong fault detectability; type of the employed nullspace basis (e.g., proper, proper and simple, full-order observer); performing least-order synthesis, etc.

- *Guaranteeing the reproducibility of results.* This feature is enforced by employing the so-called *design matrices.* These matrices are internally used to build linear combinations of left nullspace basis vectors and are frequently randomly generated (if not explicitly provided). The values of the employed design matrices are returned as additional information by all synthesis functions. The use of design matrices also represents a convenient mean to perform an optimization-based tuning of these matrices to achieve specific performance characteristics for the resulting filters.

2 Fault Detection Basics

In this section we describe first the basic fault monitoring tasks, such as fault detection and fault isolation, and then introduce and characterize the concepts of fault detectability and fault isolability. Six "canonical" fault detection problems are formulated in the book [16] for the class of *linear time-invariant* (LTI) systems with additive faults. Of the formulated six problems, three involve the exact synthesis and three involve the approximate synthesis of fault detection filters. The current release of **FDITOOLS** covers all synthesis techniques described in [16]. Jointly with the formulation of the fault detection problems, general solvability conditions are given for each problem in terms of ranks of certain transfer function matrices. More details and the proofs of the results are available in Chapters 2 and 3 of [16].

2.1 Basic Fault Monitoring Tasks

A fault represents a deviation from the normal behaviour of a system due to an unexpected event (e.g., physical component failure or supply breakdown). The occurrence of faults must be detected as early as possible to prevent any serious consequence. For this purpose, fault diagnosis techniques are used to allow the detection of occurrence of faults (fault detection) and the localization of detected faults (fault isolation). The term *fault detection and diagnosis* (FDD) includes the requirements for *fault detection and isolation* (FDI).

A FDD system is a device (usually based on a collection of real-time processing algorithms) suitably set-up to fulfill the above tasks. The minimal functionality of any FDD system is illustrated in Fig. 1.

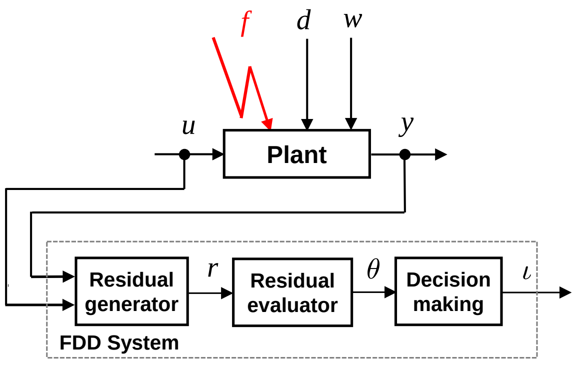

Figure 1: Basic fault diagnosis setup.

The main plant variables are the control inputs u, the unknown disturbance inputs d, the noise inputs w, and the output measurements y. The output y and control input u are the only measurable signals which can be used for fault monitoring purposes. The disturbance inputs d and noise inputs w are non-measurable "unknown" input signals, which act adversely on the system performance. For example, the unknown disturbance inputs d may represent physical disturbance inputs, as for example, wind turbulence acting on an aircraft or external loads acting on a plant. Typical noise inputs are sensor noise signals as well as process input noise. However, fictive noise inputs can also account for the cumulative effects of unmodelled system dynamics or for the effects of parametric uncertainties. In general, there is no clear-cut separation between disturbances and noise, and therefore, the appropriate definition of the disturbance and noise inputs is a challenging aspect when modelling systems for solving fault detection problems. A *fault* is any unexpected variation of some physical parameters or variables of a plant causing an unacceptable violation of certain specification limits for normal operation. Frequently, a fault input f is defined to account for any anomalous behaviour of the plant.

The main component of any FDD system (as that in Fig. 1) is the *residual generator* (or *fault detection filter*, or simply *fault detector*), which produces residual signals grouped in a q-dimensional vector r by processing the available measurements y and the known values of control inputs u. The role of the residual signals is to indicate the presence or absence of faults, and therefore the residual r must be equal (or close) to zero in the absence of faults and significantly different from zero after a fault occurs. For decision-making, suitable measures of the residual magnitudes (e.g., signal norms) are generated in a vector θ, which is then used to produce the corresponding decision vector ι. In what follows, two basic fault monitoring tasks are formulated and discussed.

Fault detection is simply a binary decision on the presence of any fault ($f \neq 0$) or the absence of all faults ($f = 0$). Typically, $\theta(t)$ is scalar evaluation signal, which approximates $\|r\|_2$, the $\mathcal{L}_2$- or ℓ_2-norms of signal r, while $\iota(t)$ is a scalar decision making signal defined as $\iota(t) = 1$ if $\theta(t) > \tau$ (fault occurrence) or $\iota(t) = 0$ if $\theta(t) \leq \tau$ (no fault), where τ is a suitable threshold quantifying the gap between the "small" and "large" magnitudes of the residual. The decision on the occurrence or absence of faults must be done in the presence of arbitrary control inputs u, disturbance inputs d, and noise inputs w acting simultaneously on the system. The effects of the control inputs on the residual can be always decoupled by a suitable choice of the residual generation filter. In the ideal case, when no noise inputs are present ($w \equiv 0$), the residual generation filter must additionally be able to *exactly* decouple the effects of the disturbances inputs in the residual and ensure, simultaneously, the sensitivity of the residual to all faults (i.e., *complete fault detectability*, see Section 2.4). In this case, $\tau = 0$ can be (ideally) used. However, in the general case when $w \not\equiv 0$, only an *approximate* decoupling of w can be achieved (at best) and a sufficient gap must exist between the magnitudes of residuals in fault-free and faulty situations. Therefore, an appropriate choice of $\tau > 0$ must avoid false alarms and missed detections.

Fault isolation concerns with the exact localization of occurred faults and involves for each component f_j of the fault vector f the decision on the presence of j-th fault ($f_j \neq 0$) or its absence ($f_j = 0$). Ideally, this must be achieved regardless the faults occur one at a time or several faults occur simultaneously. Therefore, the fault isolation task is significantly more difficult than the simpler fault detection. For fault isolation purposes, we will assume a partitioning of the q-dimensional residual vector r in n_b stacked q_i-dimensional subvectors $r^{(i)}$, $i = 1, \ldots, n_b$, in the form

$$r = \begin{bmatrix} r^{(1)} \\ \vdots \\ r^{(n_b)} \end{bmatrix}, \tag{1}$$

where $q = \sum_{i=1}^{n_b} q_i$. A typical fault evaluation setup used for fault isolation is to define $\theta_i(t)$, the

i-th component of $\theta(t)$, as a real-time computable approximation of $\|r^{(i)}\|_2$. The i-th component of $\iota(t)$ is set to $\iota_i(t) = 1$ if $\theta_i(t) > \tau_i$ (i-th residual fired) or $\iota_i(t) = 0$ if $\theta_i(t) \leq \tau_i$ (i-th residual not fired), where τ_i is a suitable threshold for the i-th subvector $r^{(i)}(t)$. If a sufficiently large number of measurements are available, then it can be aimed that $r^{(i)}$ is influenced only by the i-th fault signal f_i. This setting, with n_b chosen equal to the actual number of fault components, allows *strong fault isolation*, where an arbitrary number of simultaneous faults can be isolated. The isolation of the i-th fault is achieved if $\iota_i(t) = 1$, while for $\iota_i(t) = 0$ the i-th fault is not present. In many practical applications, the lack of a sufficiently large number of measurements impedes strong isolation of simultaneous faults. Therefore, often only *weak fault isolation* can be performed under simplifying assumptions as, for example, that the faults occur one at a time or no more than two faults may occur simultaneously. The fault isolation schemes providing weak fault isolation compare the resulting n_b-dimensional binary decision vector $\iota(t)$, with a predefined set of binary fault signatures. If each individual fault f_j has associated a distinct signature s_j, then the j-th fault can be isolated by simply checking that $\iota(t)$ matches the associated signature s_j. Similarly to fault detection, besides the decoupling of the control inputs u from the residual r (always possible), the exact decoupling of the disturbance inputs d from r can be strived in the case when $w \equiv 0$. However, in the general case when $w \not\equiv 0$, only approximate decoupling of w can be achieved (at best) and a careful selection of tolerances τ_i is necessary to perform fault isolation without false alarms and missed detections.

2.2 Plant Models with Additive Faults

The following input-output representation is used to describe LTI systems with additive faults

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_f(\lambda)\mathbf{f}(\lambda) + G_w(\lambda)\mathbf{w}(\lambda), \tag{2}$$

where $\mathbf{y}(\lambda)$, $\mathbf{u}(\lambda)$, $\mathbf{d}(\lambda)$, $\mathbf{f}(\lambda)$, and $\mathbf{w}(\lambda)$, with boldface notation, denote the Laplace-transformed (in the continuous-time case) or Z-transformed (in the discrete-time case) time-dependent vectors, namely, the p-dimensional system output vector $y(t)$, m_u-dimensional control input vector $u(t)$, m_d-dimensional disturbance vector $d(t)$, m_f-dimensional fault vector $f(t)$, and m_w-dimensional noise vector $w(t)$ respectively. $G_u(\lambda)$, $G_d(\lambda)$, $G_f(\lambda)$ and $G_w(\lambda)$ are the transfer-function matrices (TFMs) from the control inputs u, disturbance inputs d, fault inputs f, and noise inputs w to the outputs y, respectively. According to the system type, $\lambda = s$, the complex variable in the Laplace-transform in the case of a continuous-time system or $\lambda = z$, the complex variable in the Z-transform in the case of a discrete-time system. For most of practical applications, the TFMs $G_u(\lambda)$, $G_d(\lambda)$, $G_f(\lambda)$, and $G_w(\lambda)$ are proper rational matrices. However, for complete generality of our problem settings, we will allow that these TFMs are general improper rational matrices for which we will not *a priori* assume any further properties (e.g., stability, full rank, etc.).

The main difference between the disturbance input $d(t)$ and noise input $w(t)$ arises from the formulation of the fault monitoring goals. In this respect, when synthesizing devices to serve for fault diagnosis purposes, we will generally target the *exact* decoupling of the effects of disturbance inputs. Since generally the exact decoupling of effects of noise inputs is not achievable, we will simultaneously try to attenuate their effects, to achieve an *approximate* decoupling. Consequently, we will try to solve synthesis problems exactly or approximately, in accordance with the absence or presence of noise inputs in the underlying plant model, respectively.

An equivalent *descriptor* state-space realization of the input-output model (2) has the form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_u u(t) + B_d d(t) + B_f f(t) + B_w w(t)\,, \\ y(t) &= Cx(t) + D_u u(t) + D_d d(t) + D_f f(t) + D_w w(t)\,, \end{aligned} \tag{3}$$

with the n-dimensional state vector $x(t)$, where $\lambda x(t) = \dot{x}(t)$ or $\lambda x(t) = x(t+1)$ depending on the type of the system, continuous- or discrete-time, respectively. In general, the square matrix E can be singular, but we will assume that the linear pencil $A - \lambda E$ is regular. For systems with proper TFMs in (2), we can always choose a *standard* state-space realization where $E = I$. In general, it is advantageous to choose the representation (3) minimal, with the pair $(A - \lambda E, C)$ *observable* and the pair $(A - \lambda E, [\, B_u \; B_d \; B_f \; B_w \,])$ *controllable*. The corresponding TFMs of the model in (2) are

$$\begin{aligned} G_u(\lambda) &= C(\lambda E - A)^{-1}B_u + D_u, \\ G_d(\lambda) &= C(\lambda E - A)^{-1}B_d + D_d, \\ G_f(\lambda) &= C(\lambda E - A)^{-1}B_f + D_f, \\ G_w(\lambda) &= C(\lambda E - A)^{-1}B_w + D_w \end{aligned} \tag{4}$$

or in an equivalent notation

$$\left[\; G_u(\lambda) \quad G_d(\lambda) \quad G_f(\lambda) \quad G_w(\lambda) \;\right] := \left[\begin{array}{c|cccc} A - \lambda E & B_u & B_d & B_f & B_w \\ \hline C & D_u & D_d & D_f & D_w \end{array}\right].$$

2.3 Residual Generation

A linear residual generator (or fault detection filter) processes the measurable system outputs $y(t)$ and known control inputs $u(t)$ and generates the residual signals $r(t)$ which serve for decision-making on the presence or absence of faults. The input-output form of this filter is

$$\mathbf{r}(\lambda) = Q(\lambda)\left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array}\right] = Q_y(\lambda)\mathbf{y}(\lambda) + Q_u(\lambda)\mathbf{u}(\lambda)\,, \tag{5}$$

with $Q(\lambda) = [\, Q_y(\lambda) \;\; Q_u(\lambda) \,]$, and is called the *implementation form*. The TFM $Q(\lambda)$ for a physically realizable filter must be *proper* (i.e., only with finite poles) and *stable* (i.e., only with poles having negative real parts for a continuous-time system or magnitudes less than one for a discrete-time system). The dimension q of the residual vector $r(t)$ depends on the fault detection problem to be addressed.

The residual signal $r(t)$ in (5) generally depends on all system inputs $u(t)$, $d(t)$, $f(t)$ and $w(t)$ via the system output $y(t)$. The *internal form* of the filter is obtained by replacing in (5) $\mathbf{y}(\lambda)$ by its expression in (2), and is given by

$$\mathbf{r}(\lambda) = R(\lambda)\begin{bmatrix} \mathbf{u}(\lambda) \\ \mathbf{d}(\lambda) \\ \mathbf{f}(\lambda) \\ \mathbf{w}(\lambda) \end{bmatrix} = R_u(\lambda)\mathbf{u}(\lambda) + R_d(\lambda)\mathbf{d}(\lambda) + R_f(\lambda)\mathbf{f}(\lambda) + R_w(\lambda)\mathbf{w}(\lambda)\,, \tag{6}$$

with $R(\lambda) = [\, R_u(\lambda)\; R_d(\lambda)\; R_f(\lambda)\; R_w(\lambda\,]$ defined as

$$\left[\; R_u(\lambda) \;\middle|\; R_d(\lambda) \;\middle|\; R_f(\lambda) \;\middle|\; R_w(\lambda) \;\right] := Q(\lambda) \left[\begin{array}{c|c|c|c} G_u(\lambda) & G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ I_{m_u} & 0 & 0 & 0 \end{array} \right] . \qquad (7)$$

For a properly designed filter $Q(\lambda)$, the corresponding internal representation $R(\lambda)$ is also a proper and stable system, and additionally fulfills specific fault detection and isolation requirements.

2.4 Fault Detectability

The concepts of fault detectability and complete fault detectability deal with the sensitivity of the residual to an individual fault and to all faults, respectively. For the discussion of these concepts we will assume that no noise input is present in the system model (2) ($w \equiv 0$).

Definition 1. For the system (2), the j-th fault f_j is *detectable* if there exists a fault detection filter $Q(\lambda)$ such that for all control inputs u and all disturbance inputs d, the residual $r \neq 0$ if $f_j \neq 0$ and $f_k = 0$ for all $k \neq j$.

Definition 2. The system (2) is *completely fault detectable* if there exists a fault detection filter $Q(\lambda)$ such that for each j, $j = 1, \ldots, m_f$, all control inputs u and all disturbance inputs d, the residual $r \neq 0$ if $f_j \neq 0$ and $f_k = 0$ for all $k \neq j$.

We have the following results, proven in [16], which characterize the fault detectability and the complete fault detectability properties.

Proposition 1. For the system (2) the j-th fault is detectable if and only if

$$\operatorname{rank} \left[\, G_d(\lambda) \;\; G_{f_j}(\lambda) \,\right] > \operatorname{rank} G_d(\lambda), \qquad (8)$$

where $G_{f_j}(\lambda)$ is the j-th column of $G_f(\lambda)$ and rank $(\cdot)$ is the normal rank (i.e., over rational functions) of a rational matrix.

Theorem 1. The system (2) is completely fault detectable if and only if

$$\operatorname{rank} \left[\, G_d(\lambda) \;\; G_{f_j}(\lambda) \,\right] > \operatorname{rank} G_d(\lambda), \; j = 1, \ldots, m_f\, . \qquad (9)$$

Strong fault detectability is a concept related to the reliability and easiness of performing fault detection. The main idea behind this concept is the ability of the residual generators to produce persistent residual signals in the case of persistent fault excitation. For example, for reliable fault detection it is advantageous to have an asymptotically non-vanishing residual signal in the case of persistent faults as step or sinusoidal signals. On the contrary, the lack of strong fault detectability may make the detection of these type of faults more difficult, because their effects manifest in the residual only during possibly short transients, thus the effect disappears in the residual after an enough long time although the fault itself still persists.

The definitions of strong fault detectability and complete strong fault detectability cover several classes of persistent fault signals. Let $\partial\mathbb{C}_s$ denote the boundary of the stability domain, which, in the case of a continuous-time system, is the extended imaginary axis (including also the infinity), while in the case of a discrete-time system, is the unit circle centered in the origin. Let $\Omega \subset \partial\mathbb{C}_s$ be a set of complex frequencies, which characterize the classes of persistent fault signals in question. Common choices in a continuous-time setting are $\Omega = \{0\}$ for a step signal or $\Omega = \{\mathrm{i}\omega\}$ for a sinusoidal signal of frequency ω. However, Ω may contain several such frequency values or even a whole interval of frequency values, such as $\Omega = \{\mathrm{i}\omega \mid \omega \in [\omega_1, \omega_2]\}$. We denote by $\mathcal{F}_\Omega$ the class of persistent fault signals characterized by Ω.

Definition 3. For the system (2) and a given set of frequencies $\Omega \subset \partial\mathbb{C}_s$, the j-th fault f_j is *strong fault detectable* with respect to Ω if there exists a stable fault detection filter $Q(\lambda)$ such that for all control inputs u and all disturbance inputs d, the residual $r(t) \neq 0$ for $t \to \infty$ if $f_j \in \mathcal{F}_\Omega$ and $f_k = 0$ for all $k \neq j$.

Definition 4. The system (2) is *completely strong fault detectable* with respect to a given set of frequencies $\Omega \subset \partial\mathbb{C}_s$, if there exists a stable fault detection filter $Q(\lambda)$ such that for each $j = 1, \ldots, m_f$, all control inputs u and all disturbance inputs d, the residual $r(t) \neq 0$ for $t \to \infty$ if $f_j \in \mathcal{F}_\Omega$ and $f_k = 0$ for all $k \neq j$.

For a given stable filter $Q(\lambda)$ checking the strong detection property of the filter for the j-th fault f_j involves to check that $R_{f_j}(\lambda)$ has no zeros in Ω. A characterization of strong detectability as a system property is given in what follows.

Theorem 2. Let $\Omega \subset \partial\mathbb{C}_s$ be a given set of frequencies. For the system (2), f_j is strong fault detectable with respect to Ω if and only if f_j is fault detectable and the rational matrices $G_{e,j}(\lambda)$ and $\left[\begin{array}{c} G_{e,j}(\lambda) \\ F_e(\lambda) \end{array}\right]$ have the same zero structure for each $\lambda_z \in \Omega$, where

$$G_{e,j}(\lambda) := \left[\begin{array}{ccc} G_{f_j}(\lambda) & G_u(\lambda) & G_d(\lambda) \\ 0 & I_{m_u} & 0 \end{array}\right], \quad F_e(\lambda) := [\, 1 \;\; 0_{1\times m_u} \;\; 0_{1\times m_d} \,]. \tag{10}$$

Remark 1. Strong fault detectability implies fault detectability, which can be thus assimilated with a kind of *weak* fault detectability property. For the characterization of the strong fault detectability, we can impose a weaker condition, involving only the existence of a filter $Q(\lambda)$ without poles in Ω (instead imposing stability). For such a filter $Q(\lambda)$, the stability can always be achieved by replacing $Q(\lambda)$ by $M(\lambda)Q(\lambda)$, where $M(\lambda)$ is a stable and invertible TFM without zeros in Ω. Such an $M(\lambda)$ can be determined from a left coprime factorization with least order denominator of $[\, Q(\lambda)\; R_f\,(\lambda)\,]$. $\square$

For complete strong fault detectability the strong fault detectability of each individual fault is necessary, however, it is not a sufficient condition. The following theorem gives a general characterization of the complete strong fault detectability as a system property.

Theorem 3. Let Ω be the set of frequencies which characterize the persistent fault signals. The system (2) with $w \equiv 0$ is completely strong fault detectable with respect to Ω if and only if each fault f_j, for $j = 1, \ldots, m_f$, is strong fault detectable with respect to Ω and all $G_{f_j}(\lambda)$, for $j = 1, \ldots, m_f$, have the same pole structure in λ_p for all $\lambda_p \in \Omega$.

2.5 Fault Isolability

While the detectability of a fault can be individually defined and checked, for the definition of fault isolability, we need to deal with the interactions among all fault inputs. Therefore for fault isolation, we assume a structuring of the residual vector r into n_b subvectors as in (1), where each individual q_i-dimensional subvector $r^{(i)}$ is differently sensitive to faults. We assume that each fault f_j is characterized by a distinct pattern of zeros and ones in a n_b-dimensional vector s_j called the *signature* of the j-th fault. Then, fault isolation consists of recognizing which signature matches the resulting decision vector ι generated by the FDD system in Fig. 1 according to the partitioning of r in (1).

For the discussion of fault isolability, we will assume that no noise input is present in the model (2) ($w \equiv 0$). The structure of the residual vector in (1) corresponds to a $q \times m_f$ TFM $Q(\lambda)$ ($q = \sum_{i=1}^{n_b} q_i$) of the residual generation filter, built by stacking a bank of n_b filters $Q^{(1)}(\lambda)$, ..., $Q^{(n_b)}(\lambda)$ as

$$Q(\lambda) = \left[\begin{array}{c} Q^{(1)}(\lambda) \\ \vdots \\ Q^{(n_b)}(\lambda) \end{array} \right] . \qquad (11)$$

Thus, the i-th subvector $r^{(i)}$ is the output of the i-th filter with the $q_i \times m_f$ TFM $Q^{(i)}(\lambda)$

$$\mathbf{r}^{(i)}(\lambda) = Q^{(i)}(\lambda) \left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array} \right] . \qquad (12)$$

Let $R_f(\lambda)$ be the corresponding $q \times m_f$ fault-to-residual TFM in (6) and we denote $R^{(i)}_{f_j}(\lambda) := Q^{(i)}(\lambda) \left[\begin{array}{c} G_{f_j}(\lambda) \\ 0 \end{array} \right]$, the $q_i \times 1$ (i,j)-th block of $R_f(\lambda)$ which describes how the j-th fault f_j influences the i-th residual subvector $r^{(i)}$. Thus, $R_f(\lambda)$ is an $n_b \times m_f$ block-structured TFM of the form

$$R_f(\lambda) = \left[\begin{array}{ccc} R^{(1)}_{f_1}(\lambda) & \cdots & R^{(1)}_{f_{m_f}}(\lambda) \\ \vdots & \ddots & \vdots \\ R^{(n_b)}_{f_1}(\lambda) & \cdots & R^{(n_b)}_{f_{m_f}}(\lambda) \end{array} \right] . \qquad (13)$$

We associate to such a structured $R_f(\lambda)$ the $n_b \times m_f$ *structure matrix* S_{R_f} whose (i,j)-th element is defined as

$$\begin{array}{rcl} S_{R_f}(i,j) & = & 1 \quad \text{if} \quad R^{(i)}_{f_j}(\lambda) \neq 0 \, , \\ S_{R_f}(i,j) & = & 0 \quad \text{if} \quad R^{(i)}_{f_j}(\lambda) = 0 \, . \end{array} \qquad (14)$$

If $S_{R_f}(i,j) = 1$ then we say that the residual component $r^{(i)}$ is sensitive to the j-th fault f_j, while if $S_{R_f}(i,j) = 0$ then the j-th fault f_j is decoupled from $r^{(i)}$.

Fault isolability is a property which involves all faults and this is reflected in the following definition, which relates the fault isolability property to a certain structure matrix S. For a given structure matrix S, we refer to the i-th row of S as the *specification* associated with the i-th residual component $r^{(i)}$, while the j-th column of S is called the *signature* (or *code*) associated with the j-th fault f_j.

Definition 5. For a given $n_b \times m_f$ structure matrix S, the model (2) is *S-fault isolable* if there exists a fault detection filter $Q(\lambda)$ such that $S_{R_f} = S$.

When solving fault isolation problems, the choice of a suitable structure matrix S is an important aspect. This choice is, in general, not unique and several choices may lead to satisfactory synthesis results. In this context, the availability of the maximally achievable structure matrix is of paramount importance, because it allows to construct any S by simply selecting a (minimal) number of achievable specifications (i.e., rows of this matrix). The M-function `genspec`, allows to compute the maximally achievable structure matrix for a given system.

The choice of S should usually reflect the fact that complete fault detectability must be a necessary condition for the S-fault isolability. This requirement is fulfilled if S is chosen without zero columns. Also, for the unequivocal isolation of the j-th fault, the corresponding j-th column of S must be different from all other columns. Structure matrices having all columns pairwise distinct are called *weakly isolating*. Fault signatures which results as (logical OR) combinations of two or more columns of the structure matrix, can be occasionally employed to isolate simultaneous faults, provided they are distinct from all columns of S. In this sense, a structure matrix S which allows the isolation of an arbitrary number of simultaneously occurring faults is called *strongly isolating*. It is important to mention in this context that a system which is not fault isolable for a given S may still be fault isolable for another choice of the structure matrix.

To characterize the fault isolability property, we observe that each block row $Q^{(i)}(\lambda)$ of the TFM $Q(\lambda)$ is itself a fault detection filter which must achieve the specification contained in the i-th row of S. Thus, the isolability conditions will consist of a set of n_b independent conditions, each of them characterizing the complete detectability of particular subsets of faults. We have the following straightforward characterization of fault isolability.

Theorem 4. For a given $n_b \times m_f$ structure matrix S, the model (2) is S-fault isolable if and only if for $i = 1, \ldots, n_b$

$$\operatorname{rank}\,[\, G_d(\lambda)\ \widehat{G}_d^{(i)}(\lambda)\ G_{f_j}(\lambda)\,] > \operatorname{rank}[\, G_d(\lambda)\ \widehat{G}_d^{(i)}(\lambda)\,], \quad \forall j, \ \ S_{ij} \neq 0\,, \tag{15}$$

where $\widehat{G}_d^{(i)}(\lambda)$ is formed from the columns $G_{f_j}(\lambda)$ of $G_f(\lambda)$ for which $S_{ij} = 0$.

The conditions (15) of Theorem 4 give a very general characterization of isolability of faults. An important particular case is *strong fault isolability*, in which case $S = I_{m_f}$, and thus diagonal. The following result characterizes the strong isolability.

Theorem 5. The model (2) is strongly fault isolable if and only if

$$\operatorname{rank}\,[\, G_d(\lambda)\ G_f(\lambda)\,] = \operatorname{rank} G_d(\lambda) + m_f\,. \tag{16}$$

Remark 2. In the case $m_d = 0$, the strong fault isolability condition reduces to the left invertibility condition

$$\operatorname{rank} G_f(\lambda) = m_f\,. \tag{17}$$

This condition is a necessary condition even in the case $m_d \neq 0$ (otherwise $R_f(\lambda)$ would not have full column rank). $\square$

Remark 3. The definition of the structure matrix S_{R_f} associated with a given TFM $R_f(\lambda)$ can be extended to cover the strong fault detectability requirement defined by $\Omega \subset \partial\mathbb{C}_s$, where Ω is the set of relevant frequencies. For each $\lambda_z \in \Omega$, we can define the strong structure matrix at the complex frequency λ_z as

$$\begin{array}{rcl} S_{R_f}(i,j) & = & 1 \;\;\text{if}\;\; R^{(i)}_{f_j}(\lambda_z) \neq 0\,, \\ S_{R_f}(i,j) & = & 0 \;\;\text{if}\;\; R^{(i)}_{f_j}(\lambda_z) = 0\,. \end{array} \tag{18}$$

□

2.6 Fault Detection and Isolation Problems

In this section we formulate several synthesis problems of fault detection and isolation filters for LTI systems. These problems can be considered as a minimal (canonical) set to cover the needs of most practical applications. For the solution of these problems we seek linear residual generators (or fault detection filters) of the form (5), which process the measurable system outputs $y(t)$ and known control inputs $u(t)$ and generate the residual signals $r(t)$, which serve for decision-making on the presence or absence of faults. The standard requirements on all TFMs appearing in the implementation form (5) and internal form (6) of the fault detection filter are *properness* and *stability*, to ensure physical realizability of the filter $Q(\lambda)$ and to guarantee a stable behaviour of the FDD system. The *order* of the filter $Q(\lambda)$ is its *McMillan degree*, that is, the dimension of the state vector of a minimal state-space realization of $Q(\lambda)$. For practical purposes, lower order filters are preferable to larger order ones, and therefore, determining *least order residual generators* is also a desirable synthesis goal. Finally, while the dimension q of the residual vector $r(t)$ depends on the fault detection problem to be solved, filters with the *least number of outputs*, are always of interest for practical usage.

For the solution of fault detection and isolation problems it is always possible to completely decouple the control input $u(t)$ from the residual $r(t)$ by requiring $R_u(\lambda) = 0$. Regarding the disturbance input $d(t)$ and noise input $w(t)$ we aim to impose a similar condition on the disturbance input $d(t)$ by requiring $R_d(\lambda) = 0$, while minimizing simultaneously the effect of noise input $w(t)$ on the residual (e.g., by minimizing the norm of $R_w(\lambda)$). Thus, from a practical synthesis point of view, the distinction between $d(t)$ and $w(t)$ lies solely in the way these signals are treated when solving the residual generator synthesis problem.

In all fault detection problems formulated in what follows, we require that by a suitable choice of a stable fault detection filter $Q(\lambda)$, we achieve that the residual signal $r(t)$ is fully decoupled from the control input $u(t)$ and disturbance input $d(t)$. Thus, the following *decoupling conditions* must be fulfilled for the filter synthesis

$$\begin{array}{ll} (i) & R_u(\lambda) = 0\,, \\ (ii) & R_d(\lambda) = 0\,. \end{array} \tag{19}$$

In the case when condition (ii) can not be fulfilled (e.g., due to lack of sufficient number of measurements), we can redefine some (or even all) components of $d(t)$ as noise inputs and include them in $w(t)$.

For each fault detection problem formulated in what follows, specific requirements have to be fulfilled, which are formulated as additional synthesis conditions. For all formulated problems we also give the existence conditions of the solutions of these problems. For the proofs of the results consult [16].

2.6.1 EFDP – Exact Fault Detection Problem

For the *exact fault detection problem* (EFDP) the basic additional requirement is simply to achieve by a suitable choice of a stable and proper fault detection filter $Q(\lambda)$ that, in the absence of noise input (i.e., $w \equiv 0$), the residual $r(t)$ is sensitive to all fault components $f_j(t)$, $j = 1, \ldots, m_f$. If a noise input $w(t)$ is present, then we assume the TFM $G_w(s)$ is stable (thus $R_w(\lambda)$ is stable too). Thus, the following *detection condition* has to be fulfilled:

$$(iii) \;\; R_{f_j}(\lambda) \neq 0, \; j = 1, \ldots, m_f \;\text{ with }\; R_f(\lambda) \;\text{ stable.} \tag{20}$$

This is precisely the complete fault detectability requirement (without the stability condition) and leads to the following solvability condition:

Theorem 6. For the system (2), the EFDP is solvable if and only if the system (2) is completely fault detectable.

Let $\Omega \subset \partial\mathbb{C}_s$ be a given set of frequencies which characterize the relevant persistent faults. We can give a similar result in the case when the EFDP is solved with a *strong detection condition*:

$$(iii)' \;\; R_{f_j}(\lambda_z) \neq 0, \;\; \forall \lambda_z \in \Omega, \; j = 1, \ldots, m_f \;\text{ with }\; R_f(\lambda) \;\text{ stable.} \tag{21}$$

The solvability condition of the EFDP with the strong detection condition above is precisely the complete strong fault detectability requirement as stated by the following theorem.

Theorem 7. Let Ω be the set of frequencies which characterize the persistent fault signals. For the system (2), the EFDP with the strong detection condition (21) is solvable if and only if the system (2) is completely strong fault detectable with respect to Ω.

2.6.2 AFDP – Approximate Fault Detection Problem

The effects of the noise input $w(t)$ can usually not be fully decoupled from the residual $r(t)$. In this case, the basic requirements for the choice of $Q(\lambda)$ can be expressed to achieve that the residual $r(t)$ is influenced by all fault components $f_j(t)$ and the influence of the noise signal $w(t)$ is negligible. For the *approximate fault detection problem* (AFDP) the following two additional conditions have to be fulfilled:

$$\begin{array}{ll} (iii) & R_{f_j}(\lambda) \neq 0, \; j = 1, \ldots, m_f \;\text{ with }\; R_f(\lambda) \;\text{ stable;} \\ (iv) & R_w(\lambda) \approx 0, \;\text{ with }\; R_w(\lambda) \;\text{ stable.} \end{array} \tag{22}$$

Here, (iii) is the *detection condition* of all faults employed also in the EFDP, while (iv) is the *attenuation condition* for the noise input. The condition $R_w(\lambda) \approx 0$ expresses the requirement that the transfer gain $\|R_w(\lambda)\|$ (measured by any suitable norm) can be made arbitrarily small.

The solvability conditions of the formulated AFDP can be easily established:

Theorem 8. For the system (2) the AFDP is solvable if and only if the EFDP is solvable.

Remark 4. The above theorem is a pure mathematical result. The resulting filter $Q(\lambda)$, which makes $\|R_w(\lambda)\|$ "small", may simultaneously reduce $\|R_f(\lambda)\|$, such that while the fault detectability property is preserved, the filter has very limited practical use. In practice, the usefulness of a solution $Q(\lambda)$ of the AFDP must be judged by taking into account the maximum size of the noise signal and the desired minimum detectable sizes of faults. $\square$

2.6.3 EFDIP – Exact Fault Detection and Isolation Problem

For a row-block structured fault detection filter $Q(\lambda)$ as in (11), let $R_f(\lambda)$ be the corresponding block-structured fault-to-residual TFM as defined in (13) with $n_b \times m_f$ blocks, and let S_{R_f} be the corresponding $n_b \times m_f$ structure matrix defined in (14) (see Section 2.5). Let s_j, $j = 1, \ldots, m_f$ be a set of n_b-dimensional binary signature vectors associated to the faults f_j, $j = 1, \ldots, m_f$, which form the desired structure matrix $S := [\, s_1 \quad \ldots s_{m_f} \,]$. The *exact fault detection and isolation problem* (EFDIP) requires to determine for a given $n_b \times m_f$ structure matrix S, a stable and proper filter $Q(\lambda)$ of the form (11) such that the following condition is additionally fulfilled:

$$(iii) \quad S_{R_f} = S, \quad \text{with } R_f(\lambda) \text{ stable.} \tag{23}$$

We have the following straightforward solvability condition:

Theorem 9. For the system (2) with $w \equiv 0$ and a given structure matrix S, the EFDIP is solvable if and only if the system (2) is S-fault isolable.

A similar result can be established for the case when S is the m_f-th order identity matrix $S = I_{m_f}$. We call the associated synthesis problem the *strong* EFDIP. The proof is similar to that of Theorem 9.

Theorem 10. For the system (2) with $w \equiv 0$ and $S = I_{m_f}$, the EFDIP is solvable if and only if the system (2) is strongly fault isolable.

2.6.4 AFDIP – Approximate Fault Detection and Isolation Problem

Let S be a desired $n_b \times m_f$ structure matrix targeted to be achieved by using a structured fault detection filter $Q(\lambda)$ with n_b row blocks as in (11). The $n_b \times m_f$ block structured fault-to-residual TFM $R_f(\lambda)$, corresponding to $Q(\lambda)$ is defined in (13), can be additively decomposed as $R_f(\lambda) = \widetilde{R}_f(\lambda) + \overline{R}_f(\lambda)$, where $\widetilde{R}_f(\lambda)$ and $\overline{R}_f(\lambda)$ have the same block structure as $R_f(\lambda)$ and have their (i,j)-th blocks defined as

$$\widetilde{R}^{(i)}_{f_j}(\lambda) = S_{ij} R^{(i)}_{f_j}(\lambda), \quad \overline{R}^{(i)}_{f_j}(\lambda) = (1 - S_{ij}) R^{(i)}_{f_j}(\lambda)\,. \tag{24}$$

To address the approximate fault detection and isolation problem, we will target to enforce for the part $\widetilde{R}_f(\lambda)$ of $R_f(\lambda)$ the desired structure matrix S, while the part $\overline{R}_f(\lambda)$ must be (ideally) negligible. The *soft approximate fault detection and isolation problem* (*soft* AFDIP) can be formulated as follows. For a given $n_b \times m_f$ structure matrix S, determine a stable and proper filter $Q(\lambda)$ in the form (11) such that the following conditions are additionally fulfilled:

$$\begin{array}{ll} (iii) & S_{\widetilde{R}_f} = S,\ \overline{R}_f(\lambda) \approx 0, \text{ with } R_f(\lambda) \text{ stable}, \\ (iv) & R_w(\lambda) \approx 0, \text{ with } R_w(\lambda) \text{ stable}. \end{array} \tag{25}$$

The necessary and sufficient condition for the solvability of the *soft* AFDIP is the solvability of the EFDP.

Theorem 11. For the system (2) and a given structure matrix S without zero columns, the *soft* AFDIP is solvable if and only if the EFDP is solvable.

Remark 5. If the given structure matrix S has zero columns, then all faults corresponding to the zero columns of S can be redefined as additional noise inputs. In this case, the Theorem 11 can be applied to a modified system with a reduced set of faults and increased set of noise inputs. $\square$

The solvability of the EFDIP is clearly a sufficient condition for the solvability of the *soft* AFDIP, but is not, in general, also a necessary condition, unless we impose in the formulation of the AFDIP the stronger condition $\overline{R}_f(\lambda) = 0$ (instead $\overline{R}_f(\lambda) \approx 0$). This is equivalent to require $S_{R_f} = S$. Therefore, we can alternatively formulate the *strict* AFDIP to fulfill the conditions:

$$\begin{array}{ll} (iii)' & S_{R_f} = S, \text{ with } R_f(\lambda) \text{ stable}, \\ (iv)' & R_w(\lambda) \approx 0, \text{ with } R_w(\lambda) \text{ stable}. \end{array} \tag{26}$$

In this case we have the following result:

Theorem 12. For the system (2) and a given structure matrix S, the *strict* AFDIP is solvable with $S_{R_f} = S$ if and only if the EFDIP is solvable.

2.6.5 EMMP – Exact Model-Matching Problem

Let $M_{rf}(\lambda)$ be a given $q \times m_f$ TFM of a stable and proper reference model specifying the desired input-output behaviour from the faults to residuals as $\mathbf{r}(\lambda) = M_{rf}(\lambda)\mathbf{f}(\lambda)$. Thus, we want to achieve by a suitable choice of a stable and proper $Q(\lambda)$ satisfying (i) and (ii) in (19), that we have additionally $R_f(\lambda) = M_{rf}(\lambda)$. For example, a typical choice for $M_{rf}(\lambda)$ is an $m_f \times m_f$ diagonal and invertible TFM, which ensures that each residual $r_i(t)$ is influenced only by the fault $f_i(t)$. The choice $M_{rf}(\lambda) = I_{m_f}$ targets the solution of an *exact fault estimation problem* (EFEP).

To determine $Q(\lambda)$, we have to solve the linear rational equation (7), with the settings $R_u(\lambda) = 0$, $R_d(\lambda) = 0$, and $R_f(\lambda) = M_{rf}(\lambda)$ ($R_w(\lambda)$ and $G_w(\lambda)$ are assumed empty matrices). The choice of $M_{rf}(\lambda)$ may lead to a solution $Q(\lambda)$ which is not proper or is unstable or has both these undesirable properties. Therefore, besides determining $Q(\lambda)$, we also consider the determination of a suitable updating factor $M(\lambda)$ of $M_{rf}(\lambda)$ to ensure the stability and properness of the solution $Q(\lambda)$ for $R_f(\lambda) = M(\lambda)M_{rf}(\lambda)$. Obviously, $M(\lambda)$ must be chosen a proper, stable and invertible TFM. Additionally, by choosing $M(\lambda)$ diagonal, the zero and nonzero entries of $M_{rf}(\lambda)$ can be also preserved in $R_f(\lambda)$ (see also Section 2.6.3).

The *exact model-matching problem* (EMMP) can be formulated as follows: given a stable and proper $M_{rf}(\lambda)$, it is required to determine a stable and proper filter $Q(\lambda)$ and a diagonal, proper, stable and invertible TFM $M(\lambda)$ such that, additionally to (19), the following condition is fulfilled:

$$(iii) \quad R_f(\lambda) = M(\lambda)M_{rf}(\lambda)\,. \tag{27}$$

The solvability condition of the EMMP is the standard solvability condition of systems of linear equations:

Theorem 13. For the system (2) with $w \equiv 0$ and a given $M_{rf}(\lambda)$, the EMMP is solvable if and only if the following condition is fulfilled

$$\text{rank}\,[\,G_d(\lambda)\;G_f(\lambda)\,] = \text{rank}\left[\begin{array}{cc} G_d(\lambda) & G_f(\lambda) \\ 0 & M_{rf}(\lambda) \end{array}\right]. \tag{28}$$

Remark 6. When $M_{rf}(\lambda)$ has full column rank m_f, the solvability condition (28) of the EMMP reduces to the strong isolability condition (16) (see also Theorem 10). □

Remark 7. It is possible to solve a slightly more general EMMP, to determine $Q(\lambda)$ and $M(\lambda)$ as before, such that, for given $M_r(\lambda) = [\,M_{ru}(\lambda)\;M_{rd}(\lambda)\;M_{rf}(\lambda)\;M_{rw}(\lambda\,]$, they satisfy

$$Q(\lambda)\left[\begin{array}{c|c|c|c} G_u(\lambda) & G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ I_{m_u} & 0 & 0 & 0 \end{array}\right] = M(\lambda)\left[\begin{array}{c|c|c|c} M_{ru}(\lambda) & M_{rd}(\lambda) & M_{rf}(\lambda) & M_{rw}(\lambda) \end{array}\right]. \tag{29}$$

This formulation may arise, for example, if $M_r(\lambda)$ is the internal form resulted from an approximate synthesis, for which $R_u(\lambda) \approx 0$, $R_d(\lambda) \approx 0$ and $R_w(\lambda) \approx 0$.

The solvability condition is simply that for solving the linear system (29) for $M(\lambda) = I$

$$\text{rank}\,[\,G_d(\lambda)\;G_f(\lambda)\;G_w(\lambda)\,] = \text{rank}\left[\begin{array}{ccc} G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ M_{rd}(\lambda) & M_{rf}(\lambda) & M_{rw}(\lambda) \end{array}\right]. \tag{30}$$

□

The solvability conditions (see Theorem 13) become more involved if we strive for a stable proper solution $Q(\lambda)$ for a given reference model $M_{rf}(\lambda)$ without allowing its updating. For example, this is the case when solving the EFEP for $M_{rf}(\lambda) = I_{m_f}$. For a slightly more general case, we have the following result.

Theorem 14. For the system (2) with $w \equiv 0$ and a given stable and minimum-phase $M_{rf}(\lambda)$ of full column rank, the EMMP is solvable with $M(\lambda) = I$ if and only if the system is strongly fault isolable and $G_f(\lambda)$ is minimum phase.

Remark 8. If $G_f(\lambda)$ has unstable or infinite zeros, the solvability of the EMMP with $M(\lambda) = I$ is possible provided $M_{rf}(\lambda)$ is chosen such that

$$\left[\begin{array}{cc} G_f(\lambda) & G_d(\lambda) \end{array}\right] \text{ and } \left[\begin{array}{cc} G_f(\lambda) & G_d(\lambda) \\ M_{rf}(\lambda) & 0 \end{array}\right] \tag{31}$$

have the same unstable zero structure. For this it is necessary that $M_{rf}(\lambda)$ has the same unstable and infinity zeros structure as $G_f(\lambda)$. □

If we assume that that the reference model has the following row-block structured form

$$M_{rf}(\lambda) = \left[\begin{array}{c} M_{rf}^{(1)}(\lambda) \\ \vdots \\ M_{rf}^{(n_b)}(\lambda) \end{array} \right], \tag{32}$$

then we can obtain a row-block structured fault detection filter $Q(\lambda)$ as in (11), with the corresponding row-block structured $R_f(\lambda)$

$$R_f(\lambda) := \left[\begin{array}{c} Q^{(1)}(\lambda) \\ \vdots \\ Q^{(n_b)}(\lambda) \end{array} \right] \left[\begin{array}{c} G_f(\lambda) \\ 0 \end{array} \right] = \left[\begin{array}{c} R_f^{(1)}(\lambda) \\ \vdots \\ R_f^{(n_b)}(\lambda) \end{array} \right], \tag{33}$$

where the following conditions are fulfilled:

$$(iii)' \;\; R_f^{(i)}(\lambda) = M^{(i)}(\lambda) M_{rf}^{(i)}(\lambda), \quad i = 1, \ldots, n_b. \tag{34}$$

The solvability condition of the overall EMMP with the structured reference model $M_{rf}(\lambda)$ in (32) is the union of the standard solvability conditions of n_b systems of linear equations:

Theorem 15. For the system (2) with $w \equiv 0$ and a given $M_{rf}(\lambda)$ in (32), the EMMP with the conditions in (34) is solvable for $Q(\lambda)$ of the form (11) if and only if the following conditions are fulfilled

$$\operatorname{rank}\,[\, G_d(\lambda)\; G_f(\lambda)\,] = \operatorname{rank} \left[\begin{array}{cc} G_d(\lambda) & G_f(\lambda) \\ 0 & M_{rf}^{(i)}(\lambda) \end{array} \right], \quad i = 1, \ldots, n_b. \tag{35}$$

Remark 9. It is possible to solve a slightly more general EMMP, to determine, as before, a bank of n_b filters $Q^{(i)}(\lambda)$ and $M^{(i)}(\lambda)$, for $i = 1, \ldots, n_b$, such that, $Q^{(i)}(\lambda)$ satisfies, for the given $M_r^{(i)}(\lambda) = [\, M_{ru}^{(i)}(\lambda)\; M_{rd}^{(i)}(\lambda)\; M_{rf}^{(i)}(\lambda)\; M_{rw}^{(i)}(\lambda\,]$, the linear rational equation

$$Q^{(i)}(\lambda) \left[\begin{array}{c|c|c|c} G_u(\lambda) & G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ I_{m_u} & 0 & 0 & 0 \end{array} \right] = M^{(i)}(\lambda) [\, M_{ru}^{(i)}(\lambda) \mid M_{rd}^{(i)}(\lambda) \mid M_{rf}^{(i)}(\lambda) \mid M_{rw}^{(i)}(\lambda)\,]\,. \tag{36}$$

This formulation may arise, for example, if $M_r^{(i)}(\lambda)$ is the internal form resulted from an approximate synthesis, for which the corresponding components of the internal form $R_u^{(i)}(\lambda) \approx 0$, $R_d^{(i)}(\lambda) \approx 0$ and $R_w^{(i)}(\lambda) \approx 0$.

The solvability conditions are simply those for solving n_b linear systems (36) for $M^{(i)}(\lambda) = I$

$$\operatorname{rank}\,[\, G_d(\lambda)\; G_f(\lambda)\; G_w(\lambda)\,] = \operatorname{rank} \left[\begin{array}{ccc} G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ M_{rd}^{(i)}(\lambda) & M_{rf}^{(i)}(\lambda) & M_{rw}^{(i)}(\lambda) \end{array} \right], \quad i = 1, \ldots, n_b. \tag{37}$$

□

2.6.6 AMMP – Approximate Model-Matching Problem

Similarly to the formulation of the EMMP, we include the determination of an updating factor of the reference model in the standard formulation of the *approximate model-matching problem* (AMMP). Specifically, for a given stable and proper TFM $M_{rf}(\lambda)$, it is required to determine a stable and proper filter $Q(\lambda)$ and a diagonal, proper, stable and invertible TFM $M(\lambda)$ such that the following conditions are additionally fulfilled:

$$\begin{array}{ll} (iii) & R_f(\lambda) \approx M(\lambda)M_{rf}(\lambda), \;\text{ with }\; R_f(\lambda) \;\text{ stable;} \\ (iv) & R_w(\lambda) \approx 0, \;\text{ with }\; R_w(\lambda) \;\text{ stable.} \end{array} \tag{38}$$

The conditions (iii) and (iv) mean to simultaneously achieve that $\|R_f(\lambda) - M(\lambda)M_{rf}(\lambda)\| \approx 0$ and $\|R_w(\lambda)\| \approx 0$ (in some suitable norm).

A sufficient condition for the solvability of AMMP is the solvability of the EMMP.

Proposition 2. For the system (2) and a given $M_{rf}(\lambda)$, the AMMP is solvable if the EMMP is solvable.

Remark 10. It is possible to formulate a more general AMMP, to determine $Q(\lambda)$ and $M(\lambda)$ as before, such that, for given $M_r(\lambda) = [\, M_{ru}(\lambda)\; M_{rd}(\lambda)\; M_{rf}(\lambda)\; M_{rw}(\lambda)\,]$, they satisfy

$$Q(\lambda)\left[\begin{array}{c|c|c|c} G_u(\lambda) & G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ I_{m_u} & 0 & 0 & 0 \end{array}\right] \approx M(\lambda)\left[\begin{array}{c|c|c|c} M_{ru}(\lambda) & M_{rd}(\lambda) & M_{rf}(\lambda) & M_{rw}(\lambda) \end{array}\right]. \tag{39}$$

□

The AMMP can be also formulated, analogously to the EMMP, for a block-row structured reference model $M_{rf}(\lambda)$ as in (32) to obtain the row-block structured fault detection filter $Q(\lambda)$ as in (11) and diagonal, stable and invertible TFM $M^{(i)}(\lambda)$, $i = 1, \ldots, n_b$, such that for the corresponding $R_f(\lambda)$ in (33) the following conditions are additionally fulfilled:

$$\begin{array}{ll} (iii)' & R_f^{(i)}(\lambda) \approx M^{(i)}(\lambda)M_{rf}^{(i)}(\lambda), i = 1, \ldots, n_b, \;\text{ with }\; R_f(\lambda) \;\text{ stable;} \\ (iv)' & R_w(\lambda) \approx 0, \;\text{ with }\; R_w(\lambda) \;\text{ stable.} \end{array} \tag{40}$$

The Proposition 2 provides sufficient solvability conditions also for this problem formulation.

Remark 11. It is possible to formulate a more general AMMP, to determine as before, a bank of n_b filters $Q^{(i)}(\lambda)$ and $M^{(i)}(\lambda)$, for $i = 1, \ldots, n_b$, such that, $Q^{(i)}(\lambda)$ satisfies, for the given $M_r^{(i)}(\lambda) = [\, M_{ru}^{(i)}(\lambda)\; M_{rd}^{(i)}(\lambda)\; M_{rf}^{(i)}(\lambda)\; M_{rw}^{(i)}(\lambda\,]$

$$Q^{(i)}(\lambda)\left[\begin{array}{c|c|c|c} G_u(\lambda) & G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ I_{m_u} & 0 & 0 & 0 \end{array}\right] \approx M^{(i)}(\lambda)\left[\, M_{ru}^{(i)}(\lambda) \mid M_{rd}^{(i)}(\lambda) \mid M_{rf}^{(i)}(\lambda) \mid M_{rw}^{(i)}(\lambda)\,\right]. \tag{41}$$

□

2.7 Performance Evaluation of FDI Filters

Let $Q(\lambda)$ be a FDI filter of the form (5), which solves one of the six formulated FDI problems in Section 2.6. Accordingly, in the internal form (6) of the filter, the transfer function matrices $R_u(\lambda)$ and $R_d(\lambda)$ are zero to fulfill the decoupling conditions (19), $R_f(\lambda)$ is a stable transfer function matrix with m_f columns, whose zero/nonzero structure characterizes the fault detection and isolation properties, while $R_w(\lambda)$ will be generally assumed stable and nonzero. When solving fault detection and isolation problems with a targeted $n_b \times m_f$ structure matrix S, the filters $Q(\lambda)$, $R_f(\lambda)$ and $R_w(\lambda)$ have a row partitioned structure, resulted by stacking banks of n_b filters as follows

$$Q(\lambda) = \left[\begin{array}{c} Q^{(1)}(\lambda) \\ \vdots \\ Q^{(n_b)}(\lambda) \end{array} \right], \quad R_f(\lambda) = \left[\begin{array}{c} R_f^{(1)}(\lambda) \\ \vdots \\ R_f^{(n_b)}(\lambda) \end{array} \right], \quad R_w(\lambda) = \left[\begin{array}{c} R_w^{(1)}(\lambda) \\ \vdots \\ R_w^{(n_b)}(\lambda) \end{array} \right]. \tag{42}$$

The transfer function matrix $R_f(\lambda)$ has a block structure as in (13), which allows to define the associated binary structure matrix S_{R_f}, whose (i,j)-th element is 1 if $R_{f_j}^{(i)}(\lambda) \neq 0$ and 0 if $R_{f_j}^{(i)}(\lambda) \neq 0$. If S_{R_f} is the achieved structure matrix, then ideally $S_{R_f} = S$, but S_{R_f} may also differ from S, as in the case of solving a *soft* AFDIP (see Section 2.6.4).

The performance of the fault diagnosis system can be assessed using specific performance criteria, which can also serve for optimization-based tuning of various free parameters which intervene in the synthesis of FDI filters. In what follows we discuss three categories of performance criteria of which, the first one can be used to assess the fault detectability properties of the diagnosis system, the second one characterizes the noise attenuation properties and the third one characterizes the model-matching performance. In the case of block structured filters as in (42), specific performance measures are defined, taking into account the assumed "ideal" structure matrix associated with the zero and nonzero columns of $R_f^{(i)}(\lambda)$, which is provided in the i-th row of the targeted structure matrix S.

2.7.1 Fault Sensitivity Condition

When solving fault detection problems, it is important to assess the sensitivity of the residual signal to individual fault components. The *complete fault detectability* can be assessed by checking $R_{f_j}(\lambda) \neq 0$, for $j = 1, \ldots, m_f$. Alternatively, the assessment of complete fault detectability can be done by checking $\|R_f(\lambda)\|_{\infty -} > 0$, where

$$\|R_f(\lambda)\|_{\infty -} := \min_j \|R_{f_j}(\lambda)\|_\infty$$

is the $\mathcal{H}_{\infty -}$-index defined in [16], as a measure of the degree of complete fault detectability. If $\|R_f(\lambda)\|_{\infty -} = 0$, then an least one fault component is not detectable in the residual signal r. The assessment of the *strong complete fault detectability* with respect to a set of frequencies contained in a set Ω comes down to check $R_{f_j}(\lambda_s) \neq 0$, for $\forall \lambda_s \in \Omega$ and for $j = 1, \ldots, m_f$. Alternatively, the assessment of strong complete fault detectability can be done by checking $\|R_f(\lambda)\|_{\Omega -} > 0$, where

$$\|R_f(\lambda)\|_{\Omega -} := \min_j \{ \inf_{\lambda_s \in \Omega} \|R_{f_j}(\lambda_s)\|_2 \}$$

is the (modified) $\mathcal{H}_{\infty-}$-index defined over the frequencies contained in Ω (see [16]). Since nonzero values of $\|R_f(\lambda)\|_{\infty-}$ or $\|R_f(\lambda)\|_{\Omega-}$ are not invariant to scaling (e.g., when replacing $Q(\lambda)$ by $\alpha Q(\lambda)$), these quantities are less appropriate to quantitatively assess the degrees of complete detectability.

A scaling independent measure of complete fault detectability is the *fault sensitivity condition* defined (over all frequencies) as

$$J_1 = \|R_f(\lambda)\|_{\infty-} / \max_j \|R_{f_j}(\lambda)\|_\infty .$$

Similarly, scaling independent measure of the strong complete fault detectability is the *fault sensitivity condition* defined (over the frequencies contained in Ω) as

$$\widetilde{J}_1 = \|R_f(\lambda)\|_{\Omega-} / \max_j \{ \sup_{\lambda_s \in \Omega} \|R_{f_j}(\lambda_s)\|_2 \}.$$

For a completely fault detectable system J_1 satisfies

$$0 < J_1 \leq 1$$

and for a strong completely fault detectable system $\widetilde{J}_1$ satisfies

$$0 < \widetilde{J}_1 \leq 1.$$

A value of J_1 (or of $\widetilde{J}_1$) near to 1, indicates nearly equal sensitivities of residual to all fault components, and makes easier the choice of suitable thresholds for fault detection. On contrary, a small value of J_1 (or of $\widetilde{J}_1$) indicates potential difficulties in detecting some components of the fault vector, due to a very low sensitivity of the residual to these fault components. In such cases, employing fault detection filters with several outputs ($q > 1$) could be advantageous.

When solving fault detection and isolation problems with a targeted structure matrix S, we obtain partitioned filters in the form (42) and we can define for each individual filter an associated fault condition number. Let $f^{(i)}$ be formed from the subset of faults corresponding to nonzero entries in the i-th row of S and let $R^{(i)}_{f^{(i)}}(\lambda)$ be formed from the corresponding columns of $R^{(i)}_f(\lambda)$. To characterize the complete fault detectability of the subset of faults corresponding to nonzero entries in the i-th row of S we can define the fault condition number of the i-th filter as

$$J_1^{(i)} = \big\|R^{(i)}_{f^{(i)}}(\lambda)\big\|_{\infty-} / \max_j \big\|R^{(i)}_{f_j}(\lambda)\big\|_\infty .$$

Similarly, to characterize the strong complete fault detectability of the subset of faults corresponding to nonzero entries in the i-th row of S, we define the fault condition number of the i-th filter as

$$\widetilde{J}_1^{(i)} = \big\|R^{(i)}_{f^{(i)}}(\lambda)\big\|_{\Omega-} / \max_j \{ \sup_{\lambda_s \in \Omega} \big\|R^{(i)}_{f_j}(\lambda_s)\big\|_2 \}.$$

2.7.2 Fault-to-Noise Gap

A performance criterion relevant to solve approximate fault detection problems is the *fault-to-noise gap* defined as

$$J_2 = \|R_f(\lambda)\|_{\infty-} / \|R_w(\lambda)\|_\infty ,$$

which represents a measure of the noise attenuation property of the designed filter. By convention, $J_2 = 0$ if $\|R_f(\lambda)\|_{\infty-} = 0$ and $J_2 = \infty$ if $\|R_f(\lambda)\|_{\infty-} > 0$ and $\|R_w(\lambda)\|_\infty = 0$ (e.g., when solving exact synthesis problems without noise inputs). A finite frequency variant of the above criterion, which allows to address strong fault detectability aspects for a given set Ω of relevant frequencies is

$$\widetilde{J}_2 = \|R_f(\lambda)\|_{\Omega-}/\|R_w(\lambda)\|_\infty .$$

The higher the value of J_2 (or $\widetilde{J}_2$), the easier is to choose suitable thresholds to be used for fault detection purposes in the presence of noise. Therefore, the maximization of the above gaps is a valuable goal in improving the fault detection capabilities of the fault diagnosis system in the presence of exogenous noise.

For a partitioned filter in the form (42) and a targeted structure matrix S, we can define for the i-th filter component the associated value of the fault-to-noise gap, which characterizes the noise attenuation properties of the i-th filter. Let $f^{(i)}$ be formed from the subset of faults corresponding to nonzero entries in the i-th row of S and let $\bar{f}^{(i)}$ be formed from the complementary subset of faults corresponding to zero entries in the i-th row of S. If $R^{(i)}_{f^{(i)}}(\lambda)$ and $R^{(i)}_{\bar{f}^{(i)}}(\lambda)$ are formed from the columns of $R^{(i)}_f(\lambda)$ corresponding to $f^{(i)}$ and $\bar{f}^{(i)}$, respectively, then the fault-to-noise gap of the i-th filter can be defined as

$$J_2^{(i)} = \big\|R^{(i)}_{f^{(i)}}(\lambda)\big\|_{\infty-}/\big\|\,[\, R^{(i)}_{\bar{f}^{(i)}}(\lambda)\;\; R^{(i)}_w(\lambda)\,]\,\big\|_\infty .$$

This definition covers both the case of a *soft* AFDIP as well as of a *strict* AFDIP (see Section 2.6.4). For a similar characterization of the strong complete fault detectability of the subset of faults corresponding to nonzero entries in the i-th row of S, we have

$$\widetilde{J}_2^{(i)} = \big\|R^{(i)}_{f^{(i)}}(\lambda)\big\|_{\Omega-}/\big\|\,[\, R^{(i)}_{\bar{f}^{(i)}}(\lambda)\;\; R^{(i)}_w(\lambda)\,]\,\big\|_\infty .$$

2.7.3 Model-matching performance

A criterion suitable to characterize the solution of model-matching based syntheses is the residual error norm

$$J_3 = \big\|R(\lambda) - M(\lambda)M_r(\lambda)\big\|_{\infty/2},$$

where $R(\lambda) = [\, R_u(\lambda)\; R_d(\lambda)\; R_f(\lambda)\; R_w(\lambda)\,]$ is the resulting internal form (7), $M_r(\lambda)$ is a desired reference model $M_r(\lambda) = [\, M_{ru}(\lambda)\; M_{rd}(\lambda)\; M_{rf}(\lambda)\; M_{rw}(\lambda)\,]$ and $M(\lambda)$ is an updating factor. When applied to the results computed by other synthesis approaches (e.g., to solve the EFDP, AFDP, EFDIP, the *strict* AFDIP or EMMP), this criterion can be formulated as

$$\widetilde{J}_3 = \big\|R_w(\lambda)\big\|_{\infty/2},$$

which corresponds to assume that $M(\lambda) = I$ and $M_r(\lambda) = [\, R_u(\lambda)\; R_d(\lambda)\; R_f(\lambda)\; 0\,]$ (i.e., a perfect matching of control, disturbance and fault channels is always achieved).

In the case of solving an EFDIP or a *strict* AFDIP, $R_w(\lambda)$ has the partitioned form in (42). For this case, we can define for the i-th filter component the associated model-matching performance $J_3^{(i)}$, characterizing the noise attenuation property of the i-th filter. $J_3^{(i)}$ is defined simply as

$$J_3^{(i)} := \big\| R_w^{(i)}(\lambda) \big\|_{\infty/2}.$$

When solving a *soft* AFDIP, we can use a more general definition, which also accounts for possibly no exact matching of a targeted structure matrix S in the fault channel. Assuming the partitioned filter in the form (42) and a targeted structure matrix S, we build $\overline{R}_f^{(i)}(\lambda)$, with its j-th column defined as $\overline{R}_{f_j}^{(i)}(\lambda) := (1 - S_{ij}) R_{f_j}^{(i)}(\lambda)$ (see also (24)). We can define the model matching performance criterion of the i-th component filter as

$$\widetilde{J}_3^{(i)} = \big\| \big[\, \overline{R}_f^{(i)}(\lambda)\; R_w^{(i)}(\lambda) \,\big] \big\|_{\infty/2}.$$

In the case of solving an EFDIP or a *strict* AFDIP, $\overline{R}_{f_j}^{(i)}(\lambda) = 0$, and therefore $\widetilde{J}_3^{(i)} = J_3^{(i)}$.

3 Model Detection Basics

In this section we describe first the basic model detection task and introduce and characterize the concept of model detectability. Two model detection problems are formulated in the book [16] relying on LTI multiple models. The formulated synthesis problems, involve the exact synthesis and the approximate synthesis of model detection filters. Jointly with the formulation of the model detection problems, general solvability conditions are given in terms of ranks of certain transfer function matrices. More details and the proofs of the results are available in Chapters 2 and 4 of [16].

3.1 Basic Model Detection Task

Multiple models which describe various fault situations have been frequently used for fault detection purposes. In such applications, the detection of the occurrence of a fault comes down to identifying, using the available measurements from the measurable outputs and control inputs, that model (from a collection of models) which best matches the dynamical behaviour of the faulty plant. The term *model detection* describes the model identification task consisting of the selection of a model from a collection of N models, which best matches the current dynamical behaviour of a plant.

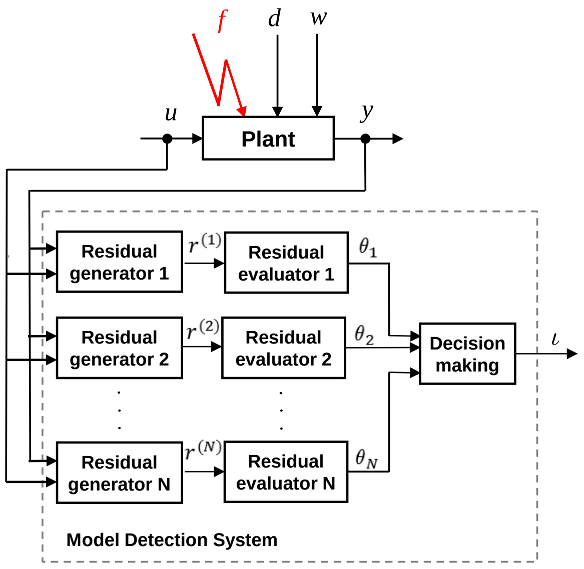

Figure 2: Basic model detection setup.

A typical model detection setting is shown in Fig. 2. A bank of N residual generation filters (or residual generators) is used, with $r^{(i)}(t)$ being the output of the i-th residual generator. The i-th component θ_i of the N-dimensional evaluation vector θ usually represents an approximation of $\|r^{(i)}\|_2$, the $\mathcal{L}_2$- or ℓ_2-norm of $r^{(i)}$. The i-th component of the N-dimensional decision vector ι is set to 0 if $\theta_i \leq \tau_i$ and 1 otherwise, where τ_i is a suitable threshold. The j-th model is "detected" if $\iota_j = 0$ and $\iota_i = 1$ for all $i \neq j$. It follows that model detection can be interpreted as a particular type of week fault isolation with N signature vectors, where the N-dimensional j-th signature vector has all elements set to one, excepting the j-th entry which is set to zero. An alternative decision scheme can also be devised if θ_i can be associated with a distance function from the current model to the i-th model. In this case, ι is a scalar, set to $\iota = j$, where j is the index for which $\theta_j = \min_{i=1:N} \theta_i$. Thus, the decision scheme selects that model j which best fits with the current model characterized by the measured input and output data.

The underlying synthesis techniques of model detection systems rely on multiple-model descriptions of physical fault cases. Since different degrees of performance degradations can be easily described via multiple models, model detection techniques have potentially the capability to address certain fault identification aspects too.

3.2 Multiple Physical Fault Models

For physically modelled faults, each fault mode leads to a distinct model. Assume that we have N LTI models describing the fault-free and faulty systems, and for $j = 1, \ldots, N$ the j-th model is specified in the input-output form

$$\mathbf{y}^{(j)}(\lambda) = G_u^{(j)}(\lambda)\mathbf{u}(\lambda) + G_d^{(j)}(\lambda)\mathbf{d}^{(j)}(\lambda) + G_w^{(j)}(\lambda)\mathbf{w}^{(j)}(\lambda), \tag{43}$$

where $y^{(j)}(t) \in \mathbb{R}^p$ is the output vector of the i-th system with control input $u(t) \in \mathbb{R}^{m_u}$, disturbance input $d^{(j)}(t) \in \mathbb{R}^{m_d^{(j)}}$ and noise input $w^{(j)}(t) \in \mathbb{R}^{m_w^{(j)}}$, and where $G_u^{(j)}(\lambda)$, $G_d^{(j)}(\lambda)$ and $G_w^{(j)}(\lambda)$ are the TFMs from the corresponding plant inputs to outputs. The significance of disturbance and noise inputs, and the basic difference between them, have already been discussed in Section 2.2. The state-space realizations corresponding to the multiple model (43) are for $j = 1, \ldots, N$ of the form

$$\begin{array}{rcl} E^{(j)}\lambda x^{(j)}(t) & = & A^{(j)}x^{(j)}(t) + B_u^{(j)}u(t) + B_d^{(j)}d^{(j)}(t) + B_w^{(j)}w^{(j)}(t)\,, \\ y^{(j)}(t) & = & C^{(j)}x^{(j)}(t) + D_u^{(j)}u(t) + D_d^{(j)}d^{(j)}(t) + D_w^{(j)}w^{(j)}(t)\,, \end{array} \tag{44}$$

where $x^{(j)}(t) \in \mathbb{R}^{n^{(j)}}$ is the state vector of the j-th system and, generally, can have different dimensions for different systems.

The multiple-model description represents a very general way to describe plant models with various faults. For example, extreme variations of parameters representing the so-called parametric faults, can be easily described by multiple models.

3.3 Residual Generation

Assume we have N LTI models of the form (43), for $j = 1, ..., N$, but the N models originate from a common underlying system with $y(t) \in \mathbb{R}^p$, the measurable output vector, and $u(t) \in \mathbb{R}^{m_u}$, the known control input. Therefore, $y^{(j)}(t) \in \mathbb{R}^p$ is the output vector of the j-th system with the control input $u(t) \in \mathbb{R}^{m_u}$, disturbance input $d^{(j)}(t) \in \mathbb{R}^{m_d^{(j)}}$ and noise input $w^{(j)}(t) \in \mathbb{R}^{m_w^{(j)}}$, respectively, and $G_u^{(j)}(\lambda)$, $G_d^{(j)}(\lambda)$ and $G_w^{(j)}(\lambda)$ are the TFMs from the corresponding plant inputs to outputs. We explicitly assumed that all models are controlled with the same control inputs

$u(t)$, but the disturbance and noise inputs $d^{(j)}(t)$ and $w^{(j)}(t)$, respectively, may differ for each component model. For complete generality of our problem formulations, we will allow that these TFMs are general rational matrices (proper or improper) for which we will not *a priori* assume any further properties.

Residual generation for model detection is performed using N linear residual generators, which process the measurable system outputs $y(t)$ and known control inputs $u(t)$ and generate N residual signals $r^{(i)}(t)$, $i = 1, \ldots, N$, which serve for decision making on which model best matches the current input-output measurement data. As already mentioned, model detection can be interpreted as a week fault isolation problem with an $N \times N$ structure matrix S having all its elements equal to one, excepting those on its diagonal which are zero. The task of model detection is thus to find out the model which best matches the measurements of outputs and inputs, by comparing the resulting decision vector ι with the set of signatures associated to each model and coded in the columns of S. The N residual generation filters in their implementation form are described for $i = 1, \ldots, N$, by the input-output relations

$$\mathbf{r}^{(i)}(\lambda) = Q^{(i)}(\lambda) \left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array} \right], \tag{45}$$

where y and u is the *actual* measured system output and control input, respectively. The TFMs $Q^{(i)}(\lambda)$, for $i = 1, \ldots, N$, must be proper and stable. The dimension q_i of the residual vector component $r^{(i)}(t)$ can be chosen always one, but occasionally values $q_i > 1$ may provide better sensitivity to model mismatches.

Assuming $y(t) = y^{(j)}(t)$, the residual signal component $r^{(i)}(t)$ in (45) generally depends on all system inputs $u(t)$, $d^{(j)}(t)$ and $w^{(j)}(t)$ via the system output $y^{(j)}(t)$. The *internal form* of the i-th filter driven by the j-th model is obtained by replacing in (45) $\mathbf{y}(\lambda)$ with $\mathbf{y}^{(j)}(\lambda)$ from (43). To make explicit the dependence of $r^{(i)}$ on the j-th model, we will use $\widetilde{r}^{(i,j)}$, to denote the i-th residual output for the j-th model. After replacing in (45) $\mathbf{y}(\lambda)$ with $\mathbf{y}^{(j)}(\lambda)$ from (43), we obtain

$$\begin{aligned} \widetilde{\mathbf{r}}^{(i,j)}(\lambda) \;&:=\; R^{(i,j)}(\lambda) \left[\begin{array}{c} \mathbf{u}(\lambda) \\ \mathbf{d}^{(j)}(\lambda) \\ \mathbf{w}^{(j)}(\lambda) \end{array} \right] \\ &=\; R_u^{(i,j)}(\lambda)\mathbf{u}(\lambda) + R_d^{(i,j)}(\lambda)\mathbf{d}^{(j)}(\lambda) + R_w^{(i,j)}(\lambda)\mathbf{w}^{(j)}(\lambda)\,, \end{aligned} \tag{46}$$

with $R^{(i,j)}(\lambda) := \left[\begin{array}{c|c|c} R_u^{(i,j)}(\lambda) & R_d^{(i,j)}(\lambda) & R_w^{(i,j)}(\lambda) \end{array} \right]$ defined as

$$\left[\begin{array}{c|c|c} R_u^{(i,j)}(\lambda) & R_d^{(i,j)}(\lambda) & R_w^{(i,j)}(\lambda) \end{array} \right] := Q^{(i)}(\lambda) \left[\begin{array}{c|c|c} G_u^{(j)}(\lambda) & G_d^{(j)}(\lambda) & G_w^{(j)}(\lambda) \\ I_{m_u} & 0 & 0 \end{array} \right]. \tag{47}$$

For a successfully designed set of filters $Q^{(i)}(\lambda)$, $i = 1, \ldots, N$, the corresponding internal representations $R^{(i,j)}(\lambda)$ in (46) are also a proper and stable.

3.4 Model Detectability

The concept of model detectability concerns with the sensitivity of the components of the residual vector to individual models from a given collection of models. Assume that we have N models, with the j-th model specified in the input-output form (43). For the discussion of the model detectability concept we will assume that no noise inputs are present in the models (43) (i.e., $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$). For model detection purposes, N filters of the form (45) are employed. It follows from (46) that the i-th component $r^{(i)}$ of the residual r is sensitive to the j-th model provided

$$R^{(i,j)}(\lambda) := \left[\begin{array}{cc} R_u^{(i,j)}(\lambda) & R_d^{(i,j)}(\lambda) \end{array}\right] \neq 0\,. \tag{48}$$

This condition involves the use of both control and disturbance inputs for model detection and can be useful even in the case of absence of control inputs.

For most of practical applications, it is however necessary to be able to perform model detection also in the (unlikely) case when the disturbance inputs are zero. Therefore, to achieve model detection independently of the presence or absence of disturbances, it is meaningful to impose instead (48), the stronger condition

$$R_u^{(i,j)}(\lambda) \neq 0\,. \tag{49}$$

This condition involves the use of only control inputs for model detection purposes and is especially relevant to active methods for model detection based on employing special inputs to help the discrimination between models.

Depending on which of the condition (48) or (49) are relevant for a particular model detection application, we define the following two concepts of model detectability.

Definition 6. *The multiple model defined by the N component systems (43) with $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$, is* extended model detectable *if there exist N filters of the form (45), such that $R^{(i,j)}(\lambda)$ defined in (48) fulfills $R^{(i,i)}(\lambda) = 0$ for $i = 1, \ldots, N$ and $R^{(i,j)}(\lambda) \neq 0$ for all $i, j = 1, \ldots, N$ such that $i \neq j$.*

Definition 7. *The multiple model defined by the N component systems (43) with $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$, is* model detectable *if there exist N filters of the form (45), such that $R^{(i,j)}(\lambda)$ defined in (48) fulfills $R^{(i,i)}(\lambda) = 0$ for $i = 1, \ldots, N$ and $R_u^{(i,j)}(\lambda) \neq 0$ for all $i, j = 1, \ldots, N$ such that $i \neq j$.*

The Definition 7 of model detectability involves the usage of only the control inputs for model detection purpose, and therefore implies the more general property of extended model detectability in Defintion 6. In the case of lack of disturbance inputs, the two definitions coincide.

The following result, proven in [16], characterizes the extended model detectability property.

Theorem 16. *The multiple model defined by the N component systems (43) with $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$, is* extended model detectable *if and only if for $i = 1, \ldots, N$*

$$\text{rank}\,[\, G_d^{(i)}(\lambda) \;\; G_d^{(j)}(\lambda) \;\; G_u^{(i)}(\lambda) - G_u^{(j)}(\lambda) \,] > \text{rank}\; G_d^{(i)}(\lambda) \;\; \forall j \neq i\,. \tag{50}$$

The characterization of model detectability (using only control inputs) can be simply established as a corollary of this theorem.

Theorem 17. The multiple model defined by the N component systems (43) with $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$, is model detectable if and only if for $i = 1, \ldots, N$

$$\operatorname{rank}\,[\, G_d^{(i)}(\lambda) \;\; G_u^{(i)}(\lambda) - G_u^{(j)}(\lambda) \,] > \operatorname{rank}\, G_d^{(i)}(\lambda) \;\; \forall j \neq i\,. \tag{51}$$

We can also define the concepts of strong model detectability and strong extended model detectability with respect to classes of persistent control inputs characterized by a set of complex frequencies $\Omega \subset \partial\mathbb{C}_s$. The following definitions formalize the aim that for each model j, there exists at least one excitation signal class characterized by a frequency $\lambda_z \in \Omega$ for which all residual components $r^{(i)}(t)$ for $i \neq j$ are asymptotically nonzero and $r^{(j)}(t)$ asymptotically vanishes.

Definition 8. The multiple model defined by the N component systems (43) with $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$, is *strong model detectable* with respect to a set of frequencies $\Omega \subset \partial\mathbb{C}_s$ if there exist N stable filters of the form (45), such that $R^{(i,j)}(\lambda)$ defined in (48) fulfills $R^{(i,i)}(\lambda) = 0$ for $i = 1, \ldots, N$ and for all $i, j = 1, \ldots, N$ with $i \neq j$, $\exists\, \lambda_z \in \Omega$ such that $R_u^{(i,j)}(\lambda_z) \neq 0$.

Definition 9. The multiple model defined by the N component systems (43) with $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$, is *strong extended model detectable* with respect to a set of frequencies $\Omega \subset \partial\mathbb{C}_s$ if there exist N stable filters of the form (45), such that $R^{(i,j)}(\lambda)$ defined in (48) fulfills $R^{(i,i)}(\lambda) = 0$ for $i = 1, \ldots, N$ and for all $i, j = 1, \ldots, N$ with $i \neq j$, $\exists\, \lambda_z \in \Omega$ such that $R^{(i,j)}(\lambda_z) \neq 0$.

The following results characterize the strong model detectability property and, respectively, the strong extended model detectability property.

Theorem 18. Let $\Omega \subset \partial\mathbb{C}_s$ be a given set of frequencies, such that none of $\lambda_z \in \Omega$ is a pole of any of the component system (43), for $j = 1, \ldots, N$. Then, the multiple model (43) with $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$, is strong model detectable with respect to Ω if and only if for $i = 1, \ldots, N$

$$\forall j \neq i, \exists \lambda_z \in \Omega \text{ such that } \operatorname{rank}\,[\, G_d^{(i)}(\lambda_z) \;\; G_u^{(i)}(\lambda_z) - G_u^{(j)}(\lambda_z) \,] > \operatorname{rank}\, G_d^{(i)}(\lambda_z)\,. \tag{52}$$

Theorem 19. Let $\Omega \subset \partial\mathbb{C}_s$ be a given set of frequencies, such that none of $\lambda_z \in \Omega$ is a pole of any of the component system (43), for $j = 1, \ldots, N$. Then, the multiple model (43) with $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$, is strong model detectable with respect to Ω if and only if for $i = 1, \ldots, N$

$$\forall j \neq i, \exists \lambda_z \in \Omega \text{ such that } \operatorname{rank}\,\left[\, G_d^{(i)}(\lambda_z) \;\; G_d^{(j)}(\lambda_z) \;\; G_u^{(i)}(\lambda_z) - G_u^{(j)}(\lambda_z) \,\right] > \operatorname{rank}\, G_d^{(i)}(\lambda_z)\,. \tag{53}$$

3.5 Model Detection Problems

In this section we formulate the exact and approximate synthesis problems of model detection filters for the collection of N LTI systems (43). As in the case of the EFDIP or AFDIP, we seek N linear residual generators (or model detection filters) of the form (45), which process the measurable system outputs $y(t)$ and known control inputs $u(t)$ and generate the N residual signals $r^{(i)}(t)$ for $i = 1, \ldots, N$. These signals serve for decision-making by comparing the pattern of fired and not fired residuals with the signatures coded in the columns of the associated standard $N \times N$ structure matrix S with zeros on the diagonal and ones elsewhere. The standard requirements for the TFMs of the filters $Q^{(i)}(\lambda)$ in (45) are *properness* and *stability*. For practical purposes, the orders of the filter $Q^{(i)}(\lambda)$ must be as small as possible. Least order filters $Q^{(i)}(\lambda)$ can be usually achieved by employing scalar output least order filters.

In analogy to the formulations of the EFDIP and AFDIP, we use the internal form of the i-th residual generator (46) to formulate the basic model detection requirements. Independently of the presence of the noise inputs $w^{(j)}$, we will target that the i-th residual is exactly decoupled from the i-th model if $w^{(i)} \equiv 0$ and is sensitive to the j-th model, for all $j \neq i$. These requirements can be easily translated into algebraic conditions using the internal form (46) of the i-th residual generator. If both control and disturbance inputs are involved in the model detection then the following conditions have to be fulfilled

$$\begin{array}{ll} (i) & [\, R_u^{(i,i)}(\lambda) \;\; R_d^{(i,i)}(\lambda)\,] = 0, \;\; i = 1, \ldots, N\,, \\ (ii) & [\, R_u^{(i,j)}(\lambda) \;\; R_d^{(i,j)}(\lambda)\,] \neq 0, \;\; \forall j \neq i, \;\text{with}\; [\, R_u^{(i,j)}(\lambda) \;\; R_d^{(i,j)}(\lambda)\,] \;\text{stable.} \end{array} \tag{54}$$

while if only control inputs have to be employed for model detection, then the following conditions have to be fulfilled

$$\begin{array}{ll} (i) & [\, R_u^{(i,i)}(\lambda) \;\; R_d^{(i,i)}(\lambda)\,] = 0, \;\; i = 1, \ldots, N\,, \\ (ii)' & R_u^{(i,j)}(\lambda) \neq 0, \;\; \forall j \neq i, \;\text{with}\; [\, R_u^{(i,j)}(\lambda) \;\; R_d^{(i,j)}(\lambda)\,] \;\text{stable.} \end{array} \tag{55}$$

Here, (i) is the *model decoupling condition* for the i-th model in the i-th residual component, while (ii) and $(ii)'$ are the *model sensitivity condition* of the i-th residual component to all models, excepting the i-th model. In the case when condition (i) cannot be fulfilled (e.g., due to lack of sufficient measurements), some (or even all) components of $d^{(i)}(t)$ can be redefined as noise inputs and included in $w^{(i)}(t)$.

In what follows, we formulate the exact and approximate model detection problems, for which we give the existence conditions of the solutions. For the proof of the results consult [16].

3.5.1 EMDP – Exact Model Detection Problem

The standard requirement for solving the *exact model detection problem* (EMDP) is to determine for the multiple model (43), in the absence of noise input (i.e., $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$), a set of N proper and stable filters $Q^{(i)}(\lambda)$ such that, for $i = 1, \ldots, N$, the conditions (54) or (55) are fulfilled. These conditions are similar to the model detectability requirement and lead to the following solvability condition:

Theorem 20. For the multiple model (43) with $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$, the EMDP is solvable with conditions (54) if and only if the multiple model (43) is extended model detectable.

Theorem 21. For the multiple model (43) with $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$, the EMDP is solvable with conditions (55) if and only if the multiple model (43) is model detectable.

Let $\Omega \subset \partial\mathbb{C}_s$ be a given set of frequencies which characterize the relevant persistent input and disturbance signals. We can give a similar result in the case when the EMDP is solved, by replacing the condition $(ii)'$ in (55), with the *strong model detection condition*:

$$(ii)'' \;\; \forall j \neq i, \;\; \exists \lambda_z \in \Omega \;\text{such that}\; R_u^{(i,j)}(\lambda_z) \neq 0, \quad \text{with}\; [\, R_u^{(i,j)}(\lambda) \;\; R_d^{(i,j)}(\lambda)\,] \;\text{stable.} \tag{56}$$

The solvability condition of the EMDP with the strong model detection condition above is precisely the strong model detectability requirement as stated by the following theorem.

Theorem 22. Let Ω be the set of frequencies which characterize the persistent control input signals. For the multiple model (43) with $w^{(j)} \equiv 0$ for $j = 1, \ldots, N$, the EMDP with the strong model detectability condition (56) is solvable if and only if the multiple model (43) is strong model detectable.

A similar result holds when targeting the strong extended model detectability property.

3.5.2 AMMP – Approximate Model Detection Problem

The effects of the noise input $w^{(i)}(t)$ can usually not be fully decoupled from the residual $r^{(i)}(t)$. In this case, the basic requirements for the choice of $Q^{(i)}(\lambda)$ can be expressed as achieving that the residual $r^{(i)}(t)$ is influenced by all models in the multiple model (43), while the influence of the i-th model is only due to the noise signal $w^{(i)}(t)$ and is negligible. Using the internal form (46) of the i-th residual generator, for the *approximate model detection problem* (AMDP) the following additional conditions to (54) or (55) have to be fulfilled:

$$\begin{array}{ll} (iii) & R_w^{(i,i)}(\lambda) \approx 0, \;\; \text{with } R_w^{(i,i)}(\lambda) \text{ stable;} \\ (iv) & R_w^{(i,j)}(\lambda) \text{ stable } \forall j \neq i. \end{array} \tag{57}$$

Here, (iii) is the *attenuation condition* of the noise input.

The solvability conditions of the AMDP are precisely those of the EMDP:

Theorem 23. For the multiple model (43) the AMDP is solvable if and only the EMDP is solvable.

3.6 Analysis and Performance Evaluation of Model Detection Filters

3.6.1 Distances between Models

For the setup of model detection applications, an important first step is the selection of a representative set of component models to serve for the design of model detection filter. A practical requirement to set up multiple models as in (43) or (44) is to choose a set of component models, such that, each component model is sufficiently far away of the rest of models. A suitable tool to measure the distance between two models is the ν-gap metric introduced in [18]. For two transfer function matrices $G_1(\lambda)$ and $G_2(\lambda)$ of the same dimensions, consider the normalized left coprime factorization $G_1(\lambda) = \widetilde{M}_1^{-1}(\lambda)\widetilde{N}_1(\lambda)$ (i.e., $\big[\, \widetilde{N}_1(\lambda) \;\, \widetilde{M}_1(\lambda) \,\big]$ is *coinner*) and the normalized right coprime factorizations $G_1(\lambda) = N_1(\lambda)M_1^{-1}(\lambda)$ and $G_2(\lambda) = N_2(\lambda)M_2^{-1}(\lambda)$ (i.e., $\left[\begin{smallmatrix} N_i(\lambda) \\ M_i(\lambda) \end{smallmatrix}\right]$ is *inner* for $i = 1, 2$). 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 $g(\lambda) := \det\big(R_2^{\sim}(\lambda)R_1(\lambda)\big)$, we have the following definition of the ν-gap metric between the two transfer-function matrices:

$$\delta_\nu(G_1(\lambda), G_2(\lambda)) := \left\{ \begin{array}{ll} \big\|\widetilde{L}_2(\lambda)R_1(\lambda)\big\|_\infty & \text{if } g(\lambda) \neq 0 \;\forall \lambda \in \partial\mathbb{C}_s \text{ and } \operatorname{wno}(g) = 0, \\ 1 & \text{otherwise,} \end{array} \right. \tag{58}$$

where $\operatorname{wno}(g)$ 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 $g(\lambda)$ [17]. Generally, for any $G_1(\lambda)$ and $G_2(\lambda)$, we have $0 \leq \delta_\nu(G_1(\lambda), G_2(\lambda)) \leq 1$. If $\delta_\nu\big(G_1(\lambda), G_2(\lambda)\big)$ is small, then we can say that $G_1(\lambda)$ and $G_2(\lambda)$ are close and it is likely that a model detection filter suited for $G_1(\lambda)$ will also work with $G_2(\lambda)$, and therefore, one of the two models can be probably removed from the set of component models. On the other side, if $\delta_\nu\big(G_1(\lambda), G_2(\lambda)\big)$ 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 $\delta_\nu\big(G_1(\lambda), G_2(\lambda)\big) = 1$ to a given $G_1(\lambda)$, but the metric fails to differentiate between them. However, this aspect should not rise difficulties in model detection applications.

In [19], the point-wise ν-gap metric is also defined to evaluate the distance between two models in a single frequency point. If λ_k 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 the frequency λ_k is:

$$\delta_\nu(G_1(\lambda_k), G_2(\lambda_k)) := \left\{ \begin{array}{cl} \big\|\widetilde{L}_2(\lambda_k)R_1(\lambda_k)\big\|_2 & \text{if } g(\lambda) \neq 0 \; \forall \lambda \in \partial\mathbb{C}_s \text{ and } \operatorname{wno}(g) = 0, \\ 1 & \text{otherwise,} \end{array} \right. \tag{59}$$

For a set of N component models with input-output forms as in (43), it is useful to determine the pairwise ν-gap distances between the control input channels of the component models by defining the symmetric matrix Δ, whose (i,j)-th entry is the ν-gap distance between the transfer-function matrices of the i-th and j-th model

$$\Delta_{ij} := \delta_\nu\big(G_u^{(i)}(\lambda), G_u^{(j)}(\lambda)\big). \tag{60}$$

It follows that Δ has all its diagonal elements zero. For model detection applications all off-diagonal elements of Δ must be nonzero, otherwise there are models which can not be potentially discriminated. The definition (60) of the distances between the i-th and j-th models focuses only on the control input channels. In most of practical applications of the model detection, this is perfectly justified by the fact that, a certain control activity is always necessary, to ensure reliable discrimination among models, independently of the presence or absence of disturbances. However, if the disturbance inputs are relevant to perform model detection (e.g., there are no control inputs), and all component models share the same disturbance inputs (i.e., $d^{(j)}(t) = d(t)$ for $j = 1, \ldots, N$), then the definition of Δ in (60) can be modified to include the disturbance inputs as well

$$\Delta_{ij} := \delta_\nu\big(\big[\, G_u^{(i)}(\lambda) \; G_d^{(i)}(\lambda) \,\big], \big[\, G_u^{(j)}(\lambda) \; G_d^{(j)}(\lambda) \,\big]\big). \tag{61}$$

If λ_k, $k = 1, \ldots, n_f$, is a set of n_f frequency values, then, instead (60), we can use the maximum of the point-wise distances

$$\Delta_{ij} := \max_k \delta_\nu\big(G_u^{(i)}(\lambda_k), G_u^{(j)}(\lambda_k)\big),$$

and similarly, instead (61), we can use the maximum of the point-wise distances

$$\Delta_{ij} := \max_k \delta_\nu\big(\big[\, G_u^{(i)}(\lambda_k) \; G_d^{(i)}(\lambda_k) \,\big], \big[\, G_u^{(j)}(\lambda_k) \; G_d^{(j)}(\lambda_k) \,\big]\big).$$

Besides the ν-gap distance between two transfer function matrices, it is possible to use distances defined in terms of the $\mathcal{H}_\infty$ norm or the $\mathcal{H}_2$ norm of the difference between them. Thus we can use instead (60)

$$\Delta_{ij} := \big\|G_u^{(i)}(\lambda) - G_u^{(j)}(\lambda)\big\|_\infty$$

or

$$\Delta_{ij} := \big\|G_u^{(i)}(\lambda) - G_u^{(j)}(\lambda)\big\|_2.$$

If λ_k, $k = 1,\ldots,n_f$, is a set of n_f frequency values, then, instead of the above norm-based distances, we can use the maximum of the point-wise distances

$$\Delta_{ij} := \max_k \big\|G_u^{(i)}(\lambda_k) - G_u^{(j)}(\lambda_k)\big\|_2.$$

3.6.2 Distances to a Current Model

An important aspect which arises in model detection applications, where the use of ν-gap metric could be instrumental, is to assess the nearness of a current model, with the input-output form

$$\mathbf{y}(\lambda) = \widetilde{G}_u(\lambda)\mathbf{u}(\lambda) + \widetilde{G}_d(\lambda)\mathbf{d}(\lambda) + \widetilde{G}_w(\lambda)\mathbf{w}(\lambda), \tag{62}$$

to the component models in (43). This involves evaluating, for $j = 1,\ldots,N$, the distances between the control input channels of the models (43) and (62) as

$$\eta_j := \delta_\nu\big(G_u^{(j)}(\lambda), \widetilde{G}_u(\lambda)\big). \tag{63}$$

It is also of interest to determine the index ℓ of that component model for which η_ℓ is the least distance. This allows to assign the model (62) to the (open) set of nearby models to the ℓ-th component model and can serve for checking the preservation of this property by the mapping achieved by the model detection filters via the norms of internal forms $R_u^{(i,j)}(\lambda)$ in (46).

If the disturbance inputs are also relevant to the model detection application, then a similar extension as above is possible to assess the distances between a current model and a set of component models by redefining η_j in (63) as

$$\eta_j := \delta_\nu\big(\big[\, G_u^{(j)}(\lambda)\; G_d^{(j)}(\lambda)\,\big], \big[\, \widetilde{G}_u(\lambda)\; \widetilde{G}_d(\lambda)\,\big]\big). \tag{64}$$

As before, we can alternatively use distances defined in terms of the $\mathcal{H}_\infty$ norm or the $\mathcal{H}_2$ norm. Thus, instead (63), we can use

$$\eta_j := \big\|G_u^{(j)}(\lambda) - \widetilde{G}_u(\lambda)\big\|_\infty$$

or

$$\eta_j := \big\|G_u^{(j)}(\lambda) - \widetilde{G}_u(\lambda)\big\|_2.$$

If λ_k, $k = 1,\ldots,n_f$, is a given set of n_f frequency values, then, instead of the above peak distances, we can use the maximum of the point-wise distances over the finite set of frequency values.

3.6.3 Distance Mapping Performance

One of the goals of the model detection is to achieve a special mapping of the distances between component models using N model detection filters of the form (45) such that the norms of the transfer-function matrices $R_u^{(i,j)}(\lambda)$ or of $\big[\, R_u^{(i,j)}(\lambda)\ R_d^{(i,j)}(\lambda) \,\big]$ in the internal forms of the filters (46) qualitatively reproduce the ν-gap distances expressed by the Δ matrix, whose (i,j)-th entries are defined in (60) or (61, respectively. The preservation of this distance mapping property is highly desirable, and the choice of model detection filters must be able to ensure this feature (at least partially for the nearest models). For example, the choice of the i-th filter $Q^{(i)}(\lambda)$ as a left annihilator of $\left[\begin{smallmatrix} G_u^{(i)}(\lambda) & G_d^{(i)}(\lambda) \\ I_{m_u} & 0 \end{smallmatrix}\right]$ ensures (see [16, Remark 6.1]) that the norm of $\big[\, R_u^{(i,j)}(\lambda)\ R_d^{(i,j)}(\lambda) \,\big]$ can be interpreted as a weighted distance between the i-th and j-th component models. It follows that the distance mapping performance of a set of model detection filters $Q^{(i)}(\lambda)$, $i = 1, \ldots, N$ can be assessed by computing mapped distance matrix Γ, whose (i,j)-th entry is

$$\Gamma_{ij} = \big\| R_u^{(i,j)}(\lambda) \big\|_\infty \tag{65}$$

or, if the disturbance inputs are relevant,

$$\Gamma_{ij} = \big\| \big[\, R_u^{(i,j)}(\lambda)\ R_d^{(i,j)}(\lambda) \,\big] \big\|_\infty . \tag{66}$$

Using the above choice of the filter $Q^{(i)}(\lambda)$, we have that all diagonal elements of Γ are zero. Additionally, to guarantee model detectability or extended model detectability (see Section 3.4), any valid design of the model detection filters must guarantee that all off-diagonal elements of Γ are nonzero. These two properties of Γ allows to unequivocally identify the exact matching of the current model with one (and only one) of the N component models.

Two other properties of Γ are desirable, when solving model detection applications. The first property is the symmetry of Γ. In contrast to Δ, Γ is generally not symmetric, excepting for some particular classes of component models and for special choices of model detection filters. For example, this property can be ensured if all component models are stable and have no disturbance inputs, by choosing $Q^{(i)}(\lambda) = \big[\, I \ \ -G_u^{(i)}(\lambda) \,\big]$, in which case $R_u^{(i,j)}(\lambda) = -R_u^{(j,i)}(\lambda)$. Ensuring the symmetry of Γ, although very desirable, is in general difficult to be achieved. In practice, it is often sufficient to ensure via suitable scaling that the gains of first row and first column are equal.

The second desirable property of the mapping $\Delta_{ij} \to \Gamma_{ij}$ is the *monotonic mapping property* of distances, which is the requirement that for all i and k $(i,k = 1, \ldots, N)$, if $\Delta_{ij} < \Delta_{ik}$, then $\Gamma_{ij} < \Gamma_{ik}$. Ensuring this property, makes easier to address model identification problems for which no exact matching between the current model and any one of the component models can be assumed.

If λ_k, $k = 1, \ldots, n_f$, is a given set of n_f frequency values, then, instead of the peak distances in (65) or in (66), we can use the maximum of the point-wise distances over the finite set of frequency values, to assess the strong model detectability or the extended strong model detectability, respectively (see Section 3.4).

3.6.4 Distance Matching Performance

To evaluate the distance matching property of the model detection filters in the case when no exact matching between the current model (62) and any one of the component models (43) can be assumed, we can define the corresponding current internal forms as

$$\left[\begin{array}{c|c|c} \widetilde{R}_u^{(i)}(\lambda) & \widetilde{R}_d^{(i)}(\lambda) & \widetilde{R}_w^{(i)}(\lambda) \end{array}\right] := Q^{(i)}(\lambda) \left[\begin{array}{c|c|c} \widetilde{G}_u(\lambda) & \widetilde{G}_d(\lambda) & \widetilde{G}_w(\lambda) \\ I_{m_u} & 0 & 0 \end{array}\right] \tag{67}$$

and evaluate the mapped distances γ_i, for $i = 1, \ldots, N$, defined as

$$\gamma_i := \big\| \widetilde{R}_u^{(i)}(\lambda) \big\|_\infty \tag{68}$$

or, if the disturbance inputs are relevant,

$$\gamma_i := \big\| \big[\, \widetilde{R}_u^{(i)}(\lambda)\ \widetilde{R}_d^{(i)}(\lambda) \,\big] \big\|_\infty . \tag{69}$$

The index ℓ of the smallest value γ_ℓ provides (for a well designed set of model detection filters) the index of the best matching component model of the current model.

If λ_k, $k = 1, \ldots, n_f$, is a given set of n_f frequency values, then, instead of the above peak distances, we can use the maximum of the point-wise distances over the finite set of frequency values.

3.6.5 Model Detection Noise Gaps

The noise attenuation performance of model detection filters can be characterized via the noise gaps achieved by individual filters. The noise gap for the i-th filter can be defined in terms of the resulting internal forms (46) as the ratio $\eta_i := \beta_i / \gamma_i$, where

$$\beta_i := \min_{j \neq i} \big\| R_u^{(i,j)}(\lambda) \big\|_\infty \tag{70}$$

and

$$\gamma_i := \big\| R_w^{(i,i)}(\lambda) \big\|_\infty . \tag{71}$$

The values of $\beta_i > 0$, for $i = 1, \ldots, N$ characterize the model detectability property of the collection of the N component models (43), while γ_i characterizes the worst-case influence of noise inputs on the i-th residual component. If $\gamma_i = 0$ (no noise), then $\eta_i = \infty$.

If the disturbance inputs are relevant for the model detection, then instead (70) we can use the following definition of β_i

$$\beta_i := \min_{j \neq i} \big\| \big[\, R_u^{(i,j)}(\lambda)\ R_d^{(i,j)}(\lambda) \,\big] \big\|_\infty . \tag{72}$$

In this case, $\beta_i > 0$, for $i = 1, \ldots, N$ characterize the extended model detectability property of the collection of the N component models (43).

If λ_k, $k = 1, \ldots, n_f$, is a given set of n_f frequency values, then, instead of (70) we use the maximum of the point-wise distances over the finite set of frequency values

$$\beta_i := \min_{j \neq i} \max_k \big\| R_u^{(i,j)}(\lambda_k) \big\|_\infty \tag{73}$$

and instead of (72) we use

$$\beta_i := \min_{j \neq i} \max_k \big\| \big[\, R_u^{(i,j)}(\lambda_k)\ R_d^{(i,j)}(\lambda_k) \,\big] \big\|_\infty . \tag{74}$$

4 Description of FDITOOLS

This user's guide is intended to provide users basic information on the **FDITOOLS** collection to solve the fault detection and isolation problems formulated in Section 2.6 and the model detection problem formulated in Section 3.5. The notations and terminology used throughout this guide have been introduced and extensively discussed in the accompanying book [16], which also represents the main reference for the implemented computational methods underlying the analysis and synthesis functions of **FDITOOLS**. Information on the requirements for installing **FDITOOLS** are given in Appendix A.

In this section, we present first a short overview of the existing functions of **FDITOOLS** and then, we illustrate a typical work flow by solving an EFDIP. In-depth information on the command syntax of the functions of the **FDITOOLS** collection is given is Sections 4.4 and 4.8. To execute the examples presented in this guide, simply paste the presented code sequences into the MATLAB command window. More involved examples are given in several case studies presented in [16].[2]

4.1 Quick Reference Tables

The current release of **FDITOOLS** is version V1.0.5, dated July 7, 2019. The corresponding `Contents.m` file is listed in Appendix B. This section contains quick reference tables for the functions of the **FDITOOLS** collection. The M-files available in the current version of **FDITOOLS**, which are documented in this user's guide, are listed below by category, with short descriptions.

Demonstration	
`FDIToolsdemo`	Demonstration of Fault Detection and Isolation Tools

Setup of synthesis models	
`fdimodset`	Setup of models for solving FDI synthesis problems.
`mdmodset`	Setup of models for solving model detection synthesis problems.

FDI Related Analysis	
`fdigenspec`	Generation of achievable FDI specifications.
`fdichkspec`	Feasibility analysis of a set of FDI specifications.

Model Detection Related Analysis	
`mddist`	Computation of distances between component models.
`mddist2c`	Computation of distances to a set of component models.

[2]Use `https://sites.google.com/site/andreasvargacontact/home/book/matlab` to download the case study examples presented in [16].

Performance evaluation of FDI filters	
`fditspec`	Computation of the weak or strong structure matrix.
`fdisspec`	Computation of the strong structure matrix.
`fdifscond`	Fault sensitivity condition of FDI filters.
`fdif2ngap`	Fault-to-noise gap of FDI filters.
`fdimmperf`	Model-matching performance of FDI filters.

Performance evaluation of model detection filters	
`mdperf`	Model detection distance mapping performance.
`mdmatch`	Model detection distance matching performance.
`mdgap`	Noise gaps of model detection filters.

Synthesis of fault detection filters	
`efdsyn`	Exact synthesis of fault detection filters.
`afdsyn`	Approximate synthesis of fault detection filters.
`efdisyn`	Exact synthesis of fault detection and isolation filters.
`afdisyn`	Approximate synthesis of fault detection and isolation filters.
`emmsyn`	Exact model matching based synthesis of FDI filters.
`ammsyn`	Approximate model matching based synthesis of FDI filters.

Synthesis of model detection filters	
`emdsyn`	Exact synthesis of model detection filters.
`amdsyn`	Approximate synthesis of model detection filters.

4.2 Getting Started

In this section we shortly illustrate the typical steps of solving a fault detection and isolation problem, starting with building an adequate fault model, performing preliminary analysis, selecting the suitable synthesis approach, and evaluating the computed results.

4.2.1 Building Models with Additive Faults

In-depth information on how to create and manipulate LTI system models and arrays of LTI system models are available in the online documentation of the Control System Toolbox and in its User' Guide [2]. These types of models are the basis of the data objects used in the **FDITOOLS** collection.

The input plant models with additive faults used by all synthesis functions of the **FDITOOLS** collection are LTI models of the form (2), given via their equivalent descriptor system state-space realizations of the form (3). The object-oriented framework employed in the Control System Toolbox has been used to define the LTI plant models, by defining several input groups corresponding to various input signal. The employed standard definitions of input groups are: `'controls'` for the control inputs $u(t)$, `'disturbances'` for the disturbance inputs $d(t)$, `'faults'` for the fault inputs $f(t)$, and `'noise'` for the noise inputs $w(t)$. For convenience, occasionally an input group `'aux'` can be also defined for additional inputs. For different ways to define input groups, see the documentation of the Control System Toolbox [2].

Once you have a plant model for a system without faults, you can construct models with faults using simple commands in the Control System Toolbox or using the model setup function `fdimodset`. For example, consider a plant model `sys` with 3 inputs, 3 outputs and 3 state components. Assume that the first two inputs are control inputs which are susceptible to actuator faults and the third input is a disturbance input, which is not measurable and therefore is considered as an unknown input. All outputs are measurable, and assume that the first output is susceptible to sensor fault. The following commands generate a plant model with additive faults as described above:

```
rng(50); sys = rss(3,3,3);
inputs = struct('c',1:2,'d',3,'f',1:2,'fs',1);
sysf = fdimodset(sys,inputs);
```

4.2.2 Determining the Achievable FDI Specifications

To determine the achievable strong FDI specifications with respect to constant faults, the function `fdigenspec` can be used as follows:

```
S = fdigenspec(sysf,struct('FDFreq',0));
```

For the above example, the possible fault signatures are contained in the generic structure matrix

$$S = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix},$$

which indicates that the EFDP is solvable (last row of S) and the EFDIP is also solvable using the specifications contained in the rows of

$$S_{FDI} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

4.2.3 Designing an FDI Filter Using `efdisyn`

To solve the EFDIP, an option structure is used to specify various user options for the synthesis function `efdisyn`. Frequently used options are the desired stability degree for the poles of the fault detection filter, the requirement for performing least order synthesis, or values for the tolerances used for rank computations or fault detectability tests. The user options can be specified by setting appropriately the respective fields in a MATLAB structure `options`. For example, the desired stability degree of -2 of the filter, the targeted fault signature specification S_{FDI}, and the frequency 0 for strong synthesis (for constant faults), can be set using

```
options = struct('sdeg',-2,'SFDI',S(1:3,:),'FDFreq',0);
```

A solution of the EFDIP, for the selected structure matrix S_{FDI}, can be computed using the function `efdisyn` as given below

```
[Q,R,info] = efdisyn(sysf,options);
```

The resulting bank of scalar output fault detection filters, in implementation form, is contained in `Q` (stored as an one-dimensional cell array of systems), while the corresponding internal forms of the bank of filters are contained in the cell array `R`. The information structure `info` contains further information on the resulting designs.

4.2.4 Assessing the Residual Generator

For assessment purposes, often simulations performed using the resulting internal form provide sufficient qualitative information to verify the obtained results. The example below illustrates how to simulate step inputs from faults using the computed cell array `R` containing the internal form of the filter.

```
Rf = vertcat(R{:});  % build the global internal form of the filter
Rf.OutputName = strcat(strseq('r_{',1:3),'}');
Rf.InputName  = strcat(strseq('f_{',1:3),'}');
step(Rf,8);
```

A typical output of this computation can be used to assess the achieved fault signatures as shown in Fig. 3. As it can be observed, the diagonal entries of the overall transfer-function matrix of `Rf` are zero, while all off-diagonal entries are nonzero.

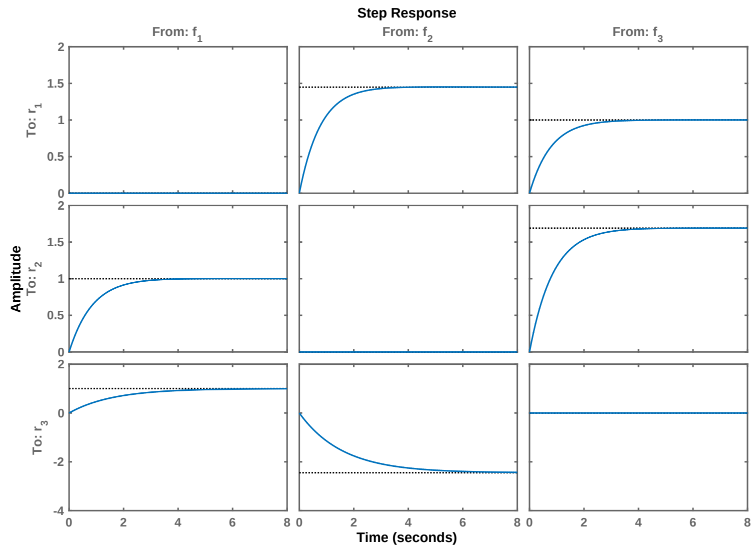

Figure 3: Step responses from the fault inputs.

The resulting strong structure matrix and the fault sensitivity conditions the resulting bank of internal filters can be obtained directly from the computed internal form R:

```
S_strong = fdisspec(R)
fscond   = fdifscond(R,0,S_strong)
```

The resulting values of the fault sensitivity conditions $\{0.6905, 0.5918, 0.4087\}$ indicate an acceptable sensitivity of all residual components to individual faults.

4.3 Functions for the Setup of Synthesis Models

The functions for the setup of synthesis models allow to easily define models in forms suitable for the use of analysis and synthesis functions.

4.3.1 `fdimodset`

Syntax

```
SYSF = fdimodset(SYS,INPUTS)
```

Description

`fdimodset` builds synthesis models with additive faults to be used in conjunction with the analysis and synthesis functions of FDI filters.

Input data

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

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B\widetilde{u}(t), \\ y(t) &= Cx(t) + D\widetilde{u}(t), \end{aligned} \tag{75}$$

where $y(t) \in \mathbb{R}^p$ is the system output and $\widetilde{u}(t) \in \mathbb{R}^m$ is the system global input, which usually includes the control and disturbance inputs, but may also explicitly include fault inputs, noise inputs and auxiliary inputs.

`INPUTS` is a MATLAB structure used to specify the indices of the columns of the matrices B and, respectively D, which correspond to the control, disturbance, fault, noise and auxiliary inputs, using the following fields:

`INPUTS` **fields**	**Description**
`controls`	vector of indices of the control inputs (Default: void)
`c`	alternative short form to specify the indices of the control inputs (Default: void)
`disturbances`	vector of indices of the disturbance inputs (Default: void)
`d`	alternative short form to specify the indices of the disturbance inputs (Default: void)

`faults`	vector of indices of the fault inputs (Default: void)
`f`	alternative short form to specify the indices of the fault inputs (Default: void)
`faults_sen`	vector of indices of the outputs subject to sensor faults (Default: void)
`fs`	alternative short form to specify the indices of the outputs subject to sensor faults (Default: void)
`noise`	vector of indices of the noise inputs (Default: void)
`n`	alternative short form to specify the indices of the noise inputs (Default: void)
`aux`	vector of indices of the auxiliary inputs (Default: void)

Output data

`SYSF` is a LTI system in the state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_u u(t) + B_d d(t) + B_f f(t) + B_w w(t) + B_v v(t), \\ y(t) &= Cx(t) + D_u u(t) + D_d d(t) + D_f f(t) + D_w w(t) + B_v v(t). \end{aligned} \tag{76}$$

where $u(t) \in \mathbb{R}^{m_u}$ are the control inputs, $d(t) \in \mathbb{R}^{m_d}$ are the disturbance inputs, $f(t) \in \mathbb{R}^{m_f}$ are the fault inputs, $w(t) \in \mathbb{R}^{m_w}$ are the noise inputs and $v(t) \in \mathbb{R}^{m_v}$ are auxiliary inputs. Any of the inputs components $u(t)$, $d(t)$, $f(t)$, $w(t)$ or $v(t)$ can be void. The input groups for $u(t)$, $d(t)$, $f(t)$, $w(t)$ and $v(t)$ have the standard names `'controls'`, `'disturbances'`, `'faults'`, `'noise'` and `'aux'`, respectively. The resulting model `SYSF` inherits the sampling time of the original model `SYS`.

Remark on input and output data

The function `fdimodset` can also be employed if `SYS` is an $N \times 1$ or $1 \times N$ array of LTI models, in which case the resulting `SYSF` is also an $N \times 1$ or $1 \times N$ array of LTI models, respectively.

Method

The system matrices B_u, B_d, B_f, B_w, B_v, and D_u, D_d, D_f, D_w, D_v in the resulting model (76) are defined by specifying the indices of the columns of the matrices B and, respectively D, in the model (75) which correspond to the control, disturbance, fault, noise and auxiliary inputs. If the system `SYS` in (75) has the equivalent input-output form

$$\mathbf{y}(\lambda) = G(\lambda)\widetilde{\mathbf{u}}(\lambda) \tag{77}$$

and the resulting system `SYSF` in (76) has the equivalent input-output form

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_f(\lambda)\mathbf{f}(\lambda) + G_w(\lambda)\mathbf{w}(\lambda) + G_v(\lambda)\mathbf{v}(\lambda), \tag{78}$$

then the following relations exist among the above transfer function matrices

$$\begin{array}{rcl} G_u(\lambda) & = & G(\lambda)S_u, \\ G_d(\lambda) & = & G(\lambda)S_d, \\ G_f(\lambda) & = & [\,G(\lambda)S_f \ S_s\,], \\ G_w(\lambda) & = & G(\lambda)S_w, \\ G_v(\lambda) & = & G(\lambda)S_v, \end{array}$$

where S_u, S_d, S_f, S_w, and S_v are columns of the identity matrix I_m and S_s is formed from the columns of the indentity matrix I_p. These (selection) matrices are used to select the corresponding columns of $G(\lambda)$, and thus to obtain the input and feedthrough matrices of the model (76) from those of the model (75). The indices of the selected columns are specified by the vectors of indices contained in the `INPUTS` structure.

Example

Example 1. For the setup of a LTI synthesis model with actuator and sensor faults consider the continuous-time input-output model defined with the transfer function matrices

$$G_u(s) = \left[\begin{array}{c} \frac{s+1}{s+2} \\[2mm] \frac{s+2}{s+3} \end{array} \right], \quad G_d(s) = \left[\begin{array}{c} \frac{s-1}{s+2} \\[2mm] 0 \end{array} \right], \quad G_f(s) = [\,G_u(s)\ I\,].$$

As it can be observed, the fault inputs correspond to an actuator fault and two sensor faults for both output measurements. To setup the state-space synthesis model, the following MATLAB commands can be employed:

```
% setup the system with additive faults
% [Gu(s) Gd(s) Gf(s)], where Gf(s) = [ Gu(s) I]
s = tf('s'); % define the Laplace variable s
Gu = [(s+1)/(s+2); (s+2)/(s+3)];  % enter Gu(s)
Gd = [(s-1)/(s+2); 0];            % enter Gd(s)
% build state space model of [Gu(s) Gd(s) Gf(s)] and set input groups
sysf = fdimodset(ss([Gu Gd]),struct('c',1,'d',2,'f',1,'fs',1:2));
```

4.3.2 `mdmodset`

Syntax

`SYSM = mdmodset(SYS,INPUTS)`

Description

`mdmodset` builds synthesis models to be used in conjunction with the synthesis functions of model detection filters.

Input data

SYS is a multiple model which contains N LTI systems, with the j-th model having the state-space form

$$\begin{array}{rcl} E^{(j)}\lambda x^{(j)}(t) & = & A^{(j)}x^{(j)}(t) + B^{(j)}\widetilde{u}^{(j)}(t)\,, \\ y^{(j)}(t) & = & C^{(j)}x^{(j)}(t) + D^{(j)}\widetilde{u}^{(j)}(t)\,, \end{array} \tag{79}$$

where $y^{(j)}(t) \in \mathbb{R}^p$, $x^{(j)}(t) \in \mathbb{R}^{n^{(j)}}$, and $\widetilde{u}^{(j)}(t) \in \mathbb{R}^{m^{(j)}}$ are the output, state and input vectors of the j-th model. The (global) input $\widetilde{u}^{(j)}(t)$ usually includes the control inputs and may also include disturbance and noise inputs. The multiple model SYS is either an one-dimensional array of N LTI systems of the form (79), in which case $m^{(j)} = m\ \forall j$, or is a $1 \times N$ cell array, with SYSM$\{j\}$ containing the j-th component system in the form (79).

INPUTS is a MATLAB structure used to specify the indices of the columns of the matrices $B^{(j)}$ and, respectively $D^{(j)}$, which correspond to the control, disturbance, and noise inputs, using the following fields:

INPUTS **fields**	**Description**
controls	vector of indices of the control inputs (Default: void)
c	alternative short form to specify the indices of the control inputs (Default: void)
disturbances	vector of indices of the disturbance inputs or an N-dimensional cell array, with INPUTS.disturbances$\{j\}$ containing the vector of indices of the disturbance inputs of the j-th component model (Default: void)
d	alternative short form to specify the indices of the disturbance inputs or an N-dimensional cell array, with INPUTS.d$\{j\}$ containing the vector of indices of the disturbance inputs of the j-th component model (Default: void)
noise	vector of indices of the noise inputs, or an N-dimensional cell array, with INPUTS.noise$\{j\}$ containing the vector of indices of the noise inputs of the j-th component model (Default: void)
n	alternative short form to specify the indices of the noise inputs, or an N-dimensional cell array, with INPUTS.n$\{j\}$ containing the vector of indices of the noise inputs of the j-th component model (Default: void)

Output data

SYSM is a multiple LTI system, with the j-th model in the state-space form

$$\begin{array}{rcl} E^{(j)}\lambda x^{(j)}(t) & = & A^{(j)}x^{(j)}(t) + B_u^{(j)}u^{(j)}(t) + B_d^{(j)}d^{(j)}(t) + B_w^{(j)}w^{(j)}(t)\,, \\ y^{(j)}(t) & = & C^{(j)}x^{(j)}(t) + D_u^{(j)}u^{(j)}(t) + D_d^{(j)}d^{(j)}(t) + D_w^{(j)}w^{(j)}(t)\,, \end{array} \tag{80}$$

where $u^{(j)}(t) \in \mathbb{R}^{m_u}$ are the control inputs, $d^{(j)}(t) \in \mathbb{R}^{m_d^{(j)}}$ are the disturbance inputs, and $w^{(j)}(t) \in \mathbb{R}^{m_w^{(j)}}$ are the noise inputs. Any of the inputs components $u^{(j)}(t)$, $d^{(j)}(t)$, or

$w^{(j)}(t)$ can be void. The input groups for $u^{(j)}(t)$, $d^{(j)}(t)$, and $w^{(j)}(t)$ have the standard names `'controls'`, `'disturbances'`, and `'noise'`, respectively. The resulting multiple model `SYSM` has the same representation as the original model `SYS` (i.e., either an one-dimensional array of N LTI systems or a $1 \times N$ cell array) and inherits the sampling time of `SYS`.

Method

The system matrices $B_u^{(j)}$, $B_d^{(j)}$, $B_w^{(j)}$, and $D_u^{(j)}$, $D_d^{(j)}$, $D_w^{(j)}$, are defined by specifying the indices of the columns of the matrices $B^{(j)}$ and, respectively $D^{(j)}$, which correspond to the control, disturbance, and noise inputs. If the j-th component system of `SYS` in (79) has the equivalent input-output form

$$\mathbf{y}^{(j)}(\lambda) = G^{(j)}(\lambda)\widetilde{\mathbf{u}}^{(j)}(\lambda) \tag{81}$$

and the resulting j-th component system of `SYSM` in (80) has the equivalent input-output form

$$\mathbf{y}^{(j)}(\lambda) = G_u^{(j)}(\lambda)\mathbf{u}(\lambda) + G_d^{(j)}(\lambda)\mathbf{d}^{(j)}(\lambda) + G_w^{(j)}(\lambda)\mathbf{w}^{(j)}(\lambda), \tag{82}$$

then the following relations exist among the above transfer function matrices

$$\begin{array}{rcl} G_u^{(j)}(\lambda) & = & G^{(j)}(\lambda)S_u, \\ G_d^{(j)}(\lambda) & = & G^{(j)}(\lambda)S_d^{(j)}, \\ G_w^{(j)}(\lambda) & = & G^{(j)}(\lambda)S_w^{(j)}, \end{array}$$

where S_u, $S_d^{(j)}$, and $S_w^{(j)}$, are columns of the identity matrix $I_{m^{(j)}}$ and are used to select the corresponding columns of $G^{(j)}(\lambda)$. The indices of the selected columns are specified by the vectors of indices contained in the `INPUTS` structure.

Examples

Example 2. Consider the first-order input-output flight actuator model

$$G(s,k) = \frac{k}{s+k}\ ,$$

where k is the actuator gain. An input-output multiple model of the form (43), defined as

$$G_u^{(j)}(s) := G(s,k^{(j)}), \quad j = 1, ..., 4,$$

covers, via suitable choices of the values of the gain k, the normal case for $k = k^{(1)}$ as well as three main classes of parametric actuator faults $k = k^{(j)}$, for $j = 2, 3, 4$, as follows:

$k^{(1)} = 14$	–	normal (nominal) case
$k^{(2)} = 0.5k^{(1)}$	–	loss of efficiency (LOE) fault
$k^{(3)} = 10k^{(1)}$	–	surface disconnection fault
$k^{(4)} = 0.01k^{(1)}$	–	stall load fault

For the setup of the multiple synthesis model to be used for model detection purposes, the following MATLAB commands can be used:

```
% Generation of a multiple model for actuator faults
s = tf('s');
k1 = 14;  % nominal gain
sysact(:,:,1) = k1/(s+k1);            % nominal case
sysact(:,:,2) = 0.5*k1/(s+0.5*k1);    % 50% LOE
sysact(:,:,3) = 10*k1/(s+10*k1);      % disconnection
sysact(:,:,4) = 0.01*k1/(s+0.01*k1); % stall load
% setup the multiple synthesis model
sysmact = mdmodset(ss(sysact),struct('c',1))
```

Example 3. This example illustrates the setup of a multiple synthesis model with variable numbers of disturbance and noise inputs. Let $N = 2$ be the number of models of the form (80), with $p = 3$ outputs, $m_u = 2$ control inputs, $m_d^{(1)} = 1$ and $m_d^{(2)} = 2$ disturbance inputs, $m_w^{(1)} = 2$ and $m_w^{(2)} = 1$ noise inputs.

For the setup of the multiple synthesis model to be used for model detection purposes, the following MATLAB commands can be used:

```
% Generation of a multiple model with two component models
p = 3; mu = 2; md1 = 1; md2 = 2; mw1 = 2; mw2 = 1;
sysm{1} = rss(4,p,mu+md1+mw1);
sysm{2} = rss(2,p,mu+md2+mw2);
% setup the multiple synthesis model
% note the compulsory use of double braces here
sysm = mdmodset(sysm,struct('c',1,'d',{{2,2:3}},'n',{{3:4,4}}))
```

4.4 Functions for FDI Related Analysis

These functions cover the generation of achievable weak and strong FDI specifications to be used for solving synthesis problems of FDI filters and the analysis of the feasibility of a set of FDI specifications. For the definitions of isolability related concepts see Section 2.5.

4.4.1 `fdigenspec`

Syntax

```
S = fdigenspec(SYSF,OPTIONS)
```

Description

`fdigenspec` determines all achievable fault detection specifications for the LTI state-space system `SYSF` with additive faults.

Input data

SYSF is a LTI system in the state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_u u(t) + B_d d(t) + B_f f(t), \\ y(t) &= Cx(t) + D_u u(t) + D_d d(t) + D_f f(t), \end{aligned} \tag{83}$$

where any of the inputs components $u(t)$, $d(t)$, and $f(t)$ can be void. For the system SYSF, the input groups for $u(t)$, $d(t)$, and $f(t)$, have the standard names 'controls', 'disturbances', and 'faults', respectively. Any additionally defined input groups are ignored.

If no standard input groups are explicitly defined, then SYSF is assumed to be a partitioned LTI system SYSF = [SYS1 SYS2] in a state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_d d(t) + B_f f(t), \\ y(t) &= Cx(t) + D_d d(t) + D_f f(t), \end{aligned} \tag{84}$$

where the inputs components $d(t) \in \mathbb{R}^{m_1}$, and $f(t) \in \mathbb{R}^{m_2}$ (both input components can be void). SYS1 has $d(t)$ as input vector and the corresponding state-space realization is $(A-\lambda E, B_d, C, D_d)$, while SYS2 has $f(t)$ as input vector and the corresponding realization is $(A-\lambda E, B_f, C, D_f)$. The dimension m_1 of the input vector $d(t)$ is specified by the OPTIONS field OPTIONS.m1 (see below). For compatibility with a previous version, if OPTIONS.m1 is specified, then the form (84) is assumed, even if the standard input groups have been explicitly defined.

OPTIONS is a MATLAB structure used to specify various options and has the following fields:

Option fields	**Description**
tol	tolerance for rank determinations (Default: internally computed)
FDTol	threshold for fault detectability checks (Default: 0.0001)
FDGainTol	threshold for strong fault detectability checks (Default: 0.01)
m1	the number m_1 of the inputs of SYS1 (Default: 0); if OPTIONS.m1 is explicitly specified, then SYSF is assumed to be partitioned as SYSF = [SYS1 SYS2] with a state-space realization of the form (84) and the definitions of input groups are ignored.
FDFreq	vector of n_f real frequency values ω_k, $k = 1, \ldots, n_f$, for strong fault detectability checks. To each real frequency ω_k, corresponds a complex frequency $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the sampling time of the system. (Default: [ ])

`sdeg`	prescribed stability degree for the poles of the internally generated filters (see **Method**): in the continuous-time case, the real parts of filters poles must be less than or equal to `OPTIONS.sdeg`, while in discrete-time case, the magnitudes of filter poles must be less than or equal to `OPTIONS.sdeg`; (Default: if `OPTIONS.FDFreq` is empty, then `OPTIONS.sdeg = [ ]`, i.e., no stabilization is performed; if `OPTIONS.FDFreq` is nonempty, then `OPTIONS.sdeg` $= -0.05$ in the continuous-time case and `OPTIONS.sdeg` $= 0.9$ in the discrete-time case).

Output data

`S` is a logical array, whose rows contains the achievable fault detection specifications. Specifically, the i-th row of `S` contains the i-th achievable specification, obtainable by using a certain (e.g., scalar output) fault detection filter $Q^{(i)}(\lambda)$, whose internal form is $R_f^{(i)}(\lambda)$, with $R_f^{(i)}(\lambda) \neq 0$ (see **Method**). Thus, the row `S`$(i,:)$ is the block-structure based structure matrix of $R_f^{(i)}(\lambda)$, such that `S`$(i,j) =$ `true` if $R_{f_j}^{(i)}(\lambda) \neq 0$ and `S`$(i,j) =$ `false` if $R_{f_j}^{(i)}(\lambda) = 0$. If the real frequency values ω_k, $k = 1,\ldots,n_f$, are provided in `OPTIONS.FDFreq` for determining strong specifications, then `S`$(i,j) =$ `true` if $\left\|R_{f_j}^{(i)}(\lambda_k)\right\| \geq$ `OPTIONS.FDGainTol` for all λ_k, $k = 1,\ldots,n_f$, where λ_k is the complex frequency corresponding to ω_k, and `S`(i,j) $=$ `false` if there exists λ_k such that $\left\|R_{f_j}^{(i)}(\lambda_k)\right\| <$ `OPTIONS.FDGainTol`.

Method

The implementation of `fdigenspec` is based on the **Procedure GENSPEC** from [16, Sect. 5.4]. The nullspace method of [8] is recursively employed to generate the complete set of achievable specifications, obtainable using suitable fault detection filters. The method is also described in [11]. In what follows we give some details of this approach.

Assume the system `SYSF` in (83) has the input-output form

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_f(\lambda)\mathbf{f}(\lambda). \tag{85}$$

If `SYSF` has the form (84), then we simply assume that $u(t)$ is void in (85). To determine the i-th row of `S`, which contains the i-th achievable specification, a certain (e.g., scalar output) fault detection filter is employed, with the input-output implementation form

$$\mathbf{r}^{(i)}(\lambda) = Q^{(i)}(\lambda)\begin{bmatrix} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{bmatrix} \tag{86}$$

and its internal form

$$\mathbf{r}^{(i)}(\lambda) = R_f^{(i)}(\lambda)\mathbf{f}(\lambda), \tag{87}$$

with $R_f^{(i)}(\lambda)$ defined as

$$R_f^{(i)}(\lambda) := Q^{(i)}(\lambda)\begin{bmatrix} G_f(\lambda) \\ 0 \end{bmatrix}. \tag{88}$$

The resulting i-th row of S is the structure matrix (weak or strong) of $R_f^{(i)}(\lambda)$. Recursive filter updating based on nullspace techniques is employed to systematically generate particular filters which are sensitive to certain fault inputs and insensitive to the rest of inputs.

The check for nonzero elements of $R_f^{(i)}(\lambda)$ is performed by using the function `fditspec` to evaluate the corresponding weak specifications. The corresponding threshold is specified via `OPTIONS.FDTol`. If frequency values for strong detectability tests are provided in `OPTIONS.FDFreq`, then the magnitudes of the elements of $R_f^{(i)}(\lambda)$ must be above a certain threshold for all complex frequencies corresponding to the specified real frequency values in `OPTIONS.FDFreq`. For this purpose, the function `fdisspec` is used to evaluate the corresponding strong specifications. The corresponding threshold is specified via `OPTIONS.FDGainTol`. The call of `fdisspec` requires that none of the complex frequencies λ_k, $k = 1, \ldots, n_f$, corresponding to the real frequencies ω_k, $k = 1, \ldots, n_f$, specified in `OPTIONS.FDFreq`, belongs to the set of poles of $R_f^{(i)}(\lambda)$. This condition is fulfilled by ensuring a certain stability degree for the poles of $R_f^{(i)}(\lambda)$, specified via `OPTIONS.sdeg`.

Example

Example 4. This is the example of [20] of a continuous-time state-space model of the form (83) with $E = I_4$,

$$A = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix}, \quad B_u = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad B_d = 0, \quad B_f = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad D_u = 0, \quad D_d = 0, \quad D_f = 0.$$

The achievable 18 weak fault specifications and 12 strong fault specifications, computed with the following script, are:

$$S_{weak} = \begin{bmatrix} 0&0&0&1&0&0&1&1\\ 0&1&1&0&1&1&1&0\\ 0&1&1&1&1&1&0&1\\ 0&1&1&1&1&1&1&1\\ 1&0&1&0&1&1&1&0\\ 1&0&1&1&1&1&0&1\\ 1&0&1&1&1&1&1&1\\ 1&1&0&0&1&1&0&0\\ 1&1&0&1&1&1&1&1\\ 1&1&1&0&0&1&1&0\\ 1&1&1&0&1&0&1&0\\ 1&1&1&0&1&1&1&0\\ 1&1&1&1&0&1&0&1\\ 1&1&1&1&0&1&1&1\\ 1&1&1&1&1&0&0&1\\ 1&1&1&1&1&0&1&1\\ 1&1&1&1&1&1&0&1\\ 1&1&1&1&1&1&1&1 \end{bmatrix}, \qquad S_{strong} = \begin{bmatrix} 0&0&0&1&0&0&1&1\\ 0&1&1&0&1&1&1&0\\ 0&1&1&1&1&1&0&1\\ 0&1&1&1&1&1&1&1\\ 1&0&1&0&1&1&1&0\\ 1&0&1&1&1&1&0&1\\ 1&0&1&1&1&1&1&1\\ 1&1&0&0&1&1&0&0\\ 1&1&0&1&1&1&1&1\\ 1&1&1&0&1&1&1&0\\ 1&1&1&1&1&1&0&1\\ 1&1&1&1&1&1&1&1 \end{bmatrix}$$

Observe that there are 6 weak specifications, which are not strong specifications.

```
% Example of Yuan et al. IJC (1997)
p = 3; mu = 1; mf = 8;
A = [ -1 1 0 0; 1 -2 1 0; 0 1 -2 1; 0 0 1 -2  ]; Bu = [1 0 0 0]';
Bf = [ 1 0 0 0 1 0 0 0; 0 1 0 0 -1 1 0 0; 0 0 1 0 0 -1 1 0; 0 0 0 1 0 0 -1 1];
C = [ 1 0 0 0; 0 0 1 0; 0 0 0 1];
Du = zeros(p,mu); Df = zeros(p,mf);
% setup the model with additive faults
sysf = ss(A,[Bu Bf],C,[Du Df]);
% set input groups
set(sysf,'InputGroup',struct('controls',1:mu,'faults',mu+(1:mf)));

% compute the achievable weak specifications
opt = struct('tol',1.e-7,'FDTol',1.e-5);
S_weak = fdigenspec(sysf,opt), size(S_weak)

% compute the achievable strong specifications for constant faults
opt = struct('tol',1.e-7,'FDTol',0.0001,'FDGainTol',.001,...
    'FDFreq',0,'sdeg',-0.05);
S_strong = fdigenspec(sysf,opt), size(S_strong)
```

4.4.2 `fdichkspec`

Syntax

```
[RDIMS,ORDERS,LEASTORDERS] = fdichkspec(SYSF)
[RDIMS,ORDERS,LEASTORDERS] = fdichkspec(SYSF,SFDI,OPTIONS)
```

Description

`fdichkspec` checks for the LTI state-space system `SYSF` with additive faults, the feasibility of a given set of FDI specifications `SFDI` and determines information related to the synthesis of FDI filters to achieve the feasible specifications.

Input data

`SYSF` is a LTI system in the state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_u u(t) + B_d d(t) + B_f f(t), \\ y(t) &= Cx(t) + D_u u(t) + D_d d(t) + D_f f(t), \end{aligned} \tag{89}$$

where any of the inputs components $u(t)$, $d(t)$, and $f(t)$ can be void. For the system `SYSF`, the input groups for $u(t)$, $d(t)$, and $f(t)$, have the standard names `'controls'`, `'disturbances'`, and `'faults'`, respectively. Any additionally defined input groups are ignored.

`SFDI` is an $N \times m_f$ logical array whose rows contain the set of FDI specifications, whose feasibility has to be checked. If `SFDI` is empty or not specified, then the fault inputs are considered void and therefore ignored.

`OPTIONS` is a MATLAB structure used to specify various synthesis options and has the following fields:

Option fields	**Description**
`tol`	tolerance for rank determinations (Default: internally computed)
`tolmin`	absolute tolerance for observability tests (Default: internally computed)
`FDTol`	threshold for fault detectability checks (Default: 0.0001)
`FDGainTol`	threshold for strong fault detectability checks (Default: 0.01)
`FDFreq`	vector of n_f real frequency values ω_k, $k = 1, \ldots, n_f$, for strong fault detectability checks. To each real frequency ω_k, corresponds a complex frequency $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the sampling time of the system. (Default: `[ ]`)

Output data

`RDIMS` is an N-dimensional integer vector, whose i-th component `RDIMS`(i), if nonzero, contains the number of residual outputs of a FDI filter based on a minimal nullspace basis, which can be used to achieve the i-th specification contained in `SFDI(`i`,:)`. If `RDIMS`$(i) = 0$, then the i-th specification is not feasible.

`ORDERS` is an N-dimensional integer vector, whose i-th component `ORDERS`(i) contains, for a feasible specification `SFDI(`i`,:)`, the order of the minimal nullspace basis based FDI filter (see above). If the i-th specification is not feasible, then `ORDERS`(i) is set to -1.

`LEASTORDERS` is an N-dimensional integer vector, whose i-th component `LEASTORDERS`(i) contains, for a feasible specification `SFDI(`i`,:)`, the least achievable order for a scalar output FDI filter which can be used to achieve the i-th specification. If the i-th specification is not feasible, then `LEASTORDERS`(i) is set to -1.

Method

The nullspace method of [8] is successively employed to determine FDI filters as minimal left nullspace bases which solve suitably formulated fault detection problems. In what follows we give some details of this approach.

Assume the system `SYSF` in (89) has the input-output form

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_f(\lambda)\mathbf{f}(\lambda). \tag{90}$$

To determine a FDI filter which achieves the i-th specification contained in the i-th row of `SFDI`, we can reformulate this FDI problem as a fault detection problem for modified sets of disturbance and fault inputs. Let $f^{(i)}$ be formed from the subset of faults corresponding to nonzero entries in the i-th row of `SFDI` and let $G_f^{(i)}(\lambda)$ be formed from the corresponding columns of $G_f(\lambda)$. Similarly, let $d^{(i)}$ be formed from the subset of faults corresponding to zero entries in the i-th row of `SFDI` and let $G_d^{(i)}(\lambda)$ be formed from the corresponding columns of $G_f(\lambda)$. The solution of the EFDIP for the i-th row of `SFDI` is thus equivalent to solve the EFDP for the modified system

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + \left[\, G_d(\lambda)\; G_d^{(i)}(\lambda) \,\right] \left[\begin{array}{c} \mathbf{d}(\lambda) \\ \mathbf{d}^{(i)}(\lambda) \end{array} \right] + G_f^{(i)}(\lambda)\mathbf{f}^{(i)}(\lambda). \tag{91}$$

A candidate fault detector filter $Q^{(i)}(\lambda)$ can be determined as a left proper nullspace basis of the transfer function matrix

$$G^{(i)}(\lambda) := \left[\begin{array}{ccc} G_u(\lambda) & G_d(\lambda) & G_d^{(i)}(\lambda) \\ I_{m_u} & 0 & 0 \end{array} \right]$$

satisfying $Q^{(i)}(\lambda)G^{(i)}(\lambda) = 0$. The corresponding internal form is

$$\mathbf{r}^{(i)}(\lambda) = R_f^{(i)}(\lambda)\mathbf{f}^{(i)}(\lambda),$$

where

$$R_f^{(i)}(\lambda) := Q^{(i)}(\lambda) \left[\begin{array}{c} G_f^{(i)}(\lambda) \\ 0 \end{array} \right]$$

can be determined proper as well. If the nullspace basis is nonempty (i.e., $Q^{(i)}(\lambda)$ has at least one row) and all columns of the resulting $R_f^{(i)}(\lambda)$ are nonzero, then the i-th specification is feasible. In this case, the number of basis vectors (i.e., the number of rows of $Q^{(i)}(\lambda)$) is returned in `RDIMS`(i) and the order of the realization of $Q^{(i)}(\lambda)$ (and also of $R_f^{(i)}(\lambda)$) is returned in `ORDERS`(i).

The check for nonzero elements of $R_f^{(i)}(\lambda)$ is performed by using the function **`fditspec`** to evaluate the corresponding weak specifications. The corresponding threshold is specified via `OPTIONS.FDTol`. If n_f frequency values ω_k, $k = 1,\ldots,n_f$, are provided in the vector `OPTIONS.FDFreq` for strong detectability tests, then the magnitudes of the elements of $R_f^{(i)}(\lambda_k)$ must be above a certain threshold for the complex frequencies λ_k, $k = 1,\ldots,n_f$, corresponding to the specified real frequency values in `OPTIONS.FDFreq`. For this purpose, the function **`fdisspec`** is used to evaluate the corresponding strong specifications. The corresponding threshold is specified via `OPTIONS.FDGainTol`. The call of **`fdisspec`** requires that the set of poles of $R_f^{(i)}(\lambda)$ and the complex frequencies λ_k, $k = 1,\ldots,n_f$, corresponding to the real frequencies specified in `OPTIONS.FDFreq`, are disjoint. This condition is fulfilled by ensuring a certain stability degree for the poles of $R_f^{(i)}(\lambda)$.

A least order scalar output filter, which fulfills the above fault detectability conditions, can be determined using minimum dynamic cover techniques [16]. This computation essentially involves the determination of a linear combination of the basis vectors using a rational vector $h(\lambda)$ such that $h(\lambda)R_f^{(i)}(\lambda)$ has all columns nonzero and $h(\lambda)Q^{(i)}(\lambda)$ has the least McMillan degree. The resulting least order of the scalar output FDI filter $h(\lambda)Q^{(i)}(\lambda)$ is returned in `LEASTORDERS`(i).

Example

Example 5. This is the example of [20] already considered in Example 4. Of the 18 weak achievable fault specifications 12 are strong fault specifications for constant faults. This can be also checked using the following MATLAB script, where the strong fault detectability checks are performed on the set of 18 weak specifications. The resulting least orders of the scalar output FDI filters to achieve the 12 feasible specifications are: 1, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2.

```
% Example of Yuan et al. IJC (1997)
p = 3; mu = 1; mf = 8;
A = [ -1 1 0 0; 1 -2 1 0; 0 1 -2 1; 0 0 1 -2  ]; Bu = [1 0 0 0]';
Bf = [ 1 0 0 0 1 0 0 0; 0 1 0 0 -1 1 0 0; 0 0 1 0 0 -1 1 0; 0 0 0 1 0 0 -1 1];
C = [ 1 0 0 0; 0 0 1 0; 0 0 0 1];
Du = zeros(p,mu); Df = zeros(p,mf);
% setup the model with additive faults
sysf = ss(A,[Bu Bf],C,[Du Df]);
% set input groups
set(sysf,'InputGroup',struct('controls',1:mu,'faults',mu+(1:mf)));
```

```
% compute the achievable weak specifications
opt = struct('tol',1.e-7,'FDTol',1.e-5);
S_weak = fdigenspec(sysf,opt), size(S_weak)

% check for the achievable strong specifications for constant faults
opt = struct('tol',1.e-7,'FDTol',0.0001,'FDGainTol',.001,...
    'FDFreq',0);
[rdims,orders,leastorders] = fdichkspec(sysf,S_weak,opt);

% select strong specifications and display the least achievable orders
S_strong = S_weak(rdims > 0,:), size(S_strong)
leastord = leastorders(rdims>0)'
```

4.5 Functions for Model Detection Related Analysis

These functions cover the evaluation of the pairwise distances between the component models of a multiple model or between the component models and a given model as defined in Section 3.6.1.

4.5.1 `mddist`

Syntax

`[DIST,FPEAK,PERM,RELDIST] = mddist(SYSM,OPTIONS)`

Description

`mddist` determines the pairwise distances between the component models of a given LTI multiple model `SYSM` containing N models.

Input data

`SYSM` is a multiple model which contains N LTI systems in the state-space form

$$\begin{aligned} E^{(j)}\lambda x^{(j)}(t) &= A^{(j)}x^{(j)}(t) + B_u^{(j)}u(t) + B_d^{(j)}d^{(j)}(t) + B_w^{(j)}w^{(j)}(t)\,, \\ y^{(j)}(t) &= C^{(j)}x^{(j)}(t) + D_u^{(j)}u(t) + D_d^{(j)}d^{(j)}(t) + D_w^{(j)}w^{(j)}(t)\,, \end{aligned} \tag{92}$$

where $x^{(j)}(t) \in \mathbb{R}^{n^{(j)}}$ is the state vector of the j-th system with control input $u(t) \in \mathbb{R}^{m_u}$, disturbance input $d^{(j)}(t) \in \mathbb{R}^{m_d^{(j)}}$ and noise input $w^{(j)}(t) \in \mathbb{R}^{m_w^{(j)}}$, and where any of the inputs components $u(t)$, $d^{(j)}(t)$, or $w^{(j)}(t)$ can be void. The multiple model `SYSM` is either an array of N LTI systems of the form (92), in which case $m_d^{(j)} = m_d$ and $m_w^{(j)} = m_w$ for $j = 1, \ldots, N$, or is an1 $\times N$ cell array, with `SYSM`$\{j\}$ containing the j-th component system in the form (92). The input groups for $u(t)$, $d^{(j)}(t)$, and $w^{(j)}(t)$ have the standard names `'controls'`, `'disturbances'`, and `'noise'`, respectively. If `OPTIONS.cdinp = true` (see below), then the same disturbance input d is assumed for all component models (i.e., $d^{(j)} = d$ and $m_d^{(j)} = m_d$). The state-space form (92) corresponds to the input-output form

$$\mathbf{y}^{(j)}(\lambda) = G_u^{(j)}(\lambda)\mathbf{u}^{(j)}(\lambda) + G_d^{(j)}(\lambda)\mathbf{d}^{(j)}(\lambda) + G_w^{(j)}(\lambda)\mathbf{w}^{(j)}(\lambda), \tag{93}$$

where $G_u^{(j)}(\lambda)$, $G_d^{(j)}(\lambda)$ and $G_w^{(j)}(\lambda)$ are the TFMs from the corresponding inputs to outputs.

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

`OPTIONS` **fields**	**Description**
`MDSelect`	M-dimensional integer vector σ with increasing elements containing the indices of the selected component models to which the distances have to be evaluated (Default: $[\,1, \ldots, N\,]$)
`tol`	relative tolerance for rank computations (Default: internally computed)
`distance`	option for the selection of the distance function $\text{dist}(G_1, G_2)$ between two transfer function matrices $G_1(\lambda)$ and $G_2(\lambda)$: `'nugap'` – $\text{dist}(G_1, G_2) = \delta_\nu(G_1, G_2)$, the ν-gap distance (default) `'Inf'` – $\text{dist}(G_1, G_2) = \|G_1 - G_2\|_\infty$, the $\mathcal{H}_\infty$-norm based distance `'2'` – $\text{dist}(G_1, G_2) = \|G_1 - G_2\|_2$, the $\mathcal{H}_2$-norm based distance
`MDFreq`	real vector, which contains the frequency values ω_k, $k = 1, \ldots, n_f$, for which the point-wise distances have to be computed. For each real frequency ω_k, there corresponds a complex frequency λ_k which is used to evaluate the point-wise distance. Depending on the system type, $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the common sampling time of the component models. (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: $\beta = 1.4901 \cdot 10^{-08}$).
`cdinp`	option to use both control and disturbance input channels to evaluate the ν-gap distances, as follows: `true` – use both control and disturbance input channels; `false` – use only the control input channels (default)
`MDIndex`	index ℓ of the ℓ-th smallest distances to be used to evaluate the relative distances to the second smallest distances (Default: $\ell = 3$)

Output data

`DIST` is an $M \times N$ nonnegative matrix, whose (i, j)-th element `DIST`(i, j) contains the computed distance (see `OPTIONS.distance`) between the selected input channels of the σ_i-th and j-th component models as follows:

– if `OPTIONS.MDFreq = []` and `OPTIONS.cdinp = false` then

$$\texttt{DIST}(i,j) = \operatorname{dist}\left(G_u^{(\sigma_i)}(\lambda), G_u^{(j)}(\lambda)\right)$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = false` then

$$\texttt{DIST}(i,j) = \max_k \operatorname{dist}\left(G_u^{(\sigma_i)}(\lambda_k), G_u^{(j)}(\lambda_k)\right)$$

– if `OPTIONS.MDFreq = []` and `OPTIONS.cdinp = true` then

$$\texttt{DIST}(i,j) = \operatorname{dist}\left(\left[\, G_u^{(\sigma_i)}(\lambda)\ G_d^{(\sigma_i)}(\lambda)\,\right], \left[\, G_u^{(j)}(\lambda)\ G_d^{(j)}(\lambda)\,\right]\right)$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = true` then

$$\texttt{DIST}(i,j) = \max_k \operatorname{dist}\left(\left[\, G_u^{(\sigma_i)}(\lambda_k)\ G_d^{(\sigma_i)}(\lambda_k)\,\right], \left[\, G_u^{(j)}(\lambda_k)\ G_d^{(j)}(\lambda_k)\,\right]\right)$$

`FPEAK` is an $M \times N$ nonnegative matrix, whose (i,j)-th element `FPEAK`(i,j) contains the peak frequency (in rad/TimeUnit), where `DIST`(i,j) is achieved.

`PERM` is an $M \times N$ integer matrix, whose i-th row contains the permutation to be applied to increasingly reorder the i-th row of `DIST`.

`RELDIST` is an M-dimensional vector, whose i-th element `RELDIST`(i) contains the ratio of the second and ℓ-th smallest distances in the i-th row of `DIST`, where $\ell =$ `OPTIONS.MDIndex`.

Method

The definition of the distances between component models is given in Section 3.6.1. The evaluation of the ν-gap distances relies on the definition proposed in [18]. For efficiency purposes, the intervening normalized factorizations of the components systems are performed only once and all existing symmetries are exploited. The point-wise distances $\operatorname{dist}\left(G_u^{(i)}(\lambda_k), G_u^{(j)}(\lambda_k)\right)$ for the $\mathcal{H}_\infty$- and $\mathcal{H}_2$ norms are simply the 2-norm of the difference of the frequency responses $\operatorname{dist}\left(G_u^{(j)}(\lambda_k), G_u^{(j)}(\lambda_k)\right) = \left\|G_u^{(j)}(\lambda_k) - G_u^{(j)}(\lambda_k)\right\|_2$. Similar formulas apply if the disturbance inputs are also selected.

4.5.2 `mddist2c`

Syntax

```
[DIST,FPEAK,MIND] = mddist2c(SYSM,SYS,OPTIONS)
```

Description

`mddist2c` determines the distances of the component models of a LTI multiple model `SYSM` to the current model `SYS`.

Input data

`SYSM` is a multiple model which contains N LTI systems in the state-space form

$$\begin{array}{rcl} E^{(j)}\lambda x^{(j)}(t) & = & A^{(j)}x^{(j)}(t) + B_u^{(j)}u(t) + B_d^{(j)}d^{(j)}(t) + B_w^{(j)}w^{(j)}(t)\,, \\ y^{(j)}(t) & = & C^{(j)}x^{(j)}(t) + D_u^{(j)}u(t) + D_d^{(j)}d^{(j)}(t) + D_w^{(j)}w^{(j)}(t)\,, \end{array} \tag{94}$$

where $x^{(j)}(t) \in \mathbb{R}^{n^{(j)}}$ is the state vector of the j-th system with control input $u(t) \in \mathbb{R}^{m_u}$, disturbance input $d^{(j)}(t) \in \mathbb{R}^{m_d^{(j)}}$ and noise input $w^{(j)}(t) \in \mathbb{R}^{m_w^{(j)}}$, and where any of the inputs components $u(t)$, $d^{(j)}(t)$, or $w^{(j)}(t)$ can be void. The multiple model `SYSM` is either an array of N LTI systems of the form (94), in which case $m_d^{(j)} = m_d$ and $m_w^{(j)} = m_w$ for $j = 1, \ldots, N$, or is an $1 \times N$ cell array, with `SYSM`$\{j\}$ containing the j-th component system in the form (94). The input groups for $u(t)$, $d^{(j)}(t)$, and $w^{(j)}(t)$ have the standard names `'controls'`, `'disturbances'`, and `'noise'`, respectively. If `OPTIONS.cdinp = true` (see below), then the same disturbance input d is assumed for all component models (i.e., $d^{(j)} = d$ and $m_d^{(j)} = m_d$). The state-space form (94) corresponds to the input-output form

$$\mathbf{y}^{(j)}(\lambda) = G_u^{(j)}(\lambda)\mathbf{u}^{(j)}(\lambda) + G_d^{(j)}(\lambda)\mathbf{d}^{(j)}(\lambda) + G_w^{(j)}(\lambda)\mathbf{w}^{(j)}(\lambda), \tag{95}$$

where $G_u^{(j)}(\lambda)$, $G_d^{(j)}(\lambda)$ and $G_w^{(j)}(\lambda)$ are the TFMs from the corresponding inputs to outputs.

`SYS` is a LTI model in the state-space form

$$\begin{array}{rcl} E\lambda x(t) & = & Ax(t) + B_u u(t) + B_d d(t) + B_w w(t)\,, \\ y(t) & = & Cx(t) + D_u u(t) + D_d d(t) + D_w w(t)\,, \end{array} \tag{96}$$

where $x(t) \in \mathbb{R}^n$ is the state vector of the system with control input $u(t) \in \mathbb{R}^{m_u}$, disturbance input $d(t) \in \mathbb{R}^{m_d}$ and noise input $w(t) \in \mathbb{R}^{m_w}$, and where any of the inputs components $u(t)$, $d(t)$, or $w(t)$ can be void. The input groups for $u(t)$, $d(t)$, and $w(t)$ have the standard names `'controls'`, `'disturbances'`, and `'noise'`, respectively. The state-space form (96) corresponds to the input-output form

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_w(\lambda)\mathbf{w}(\lambda), \tag{97}$$

where $G_u(\lambda)$, $G_d(\lambda)$ and $G_w(\lambda)$ are the TFMs from the corresponding inputs to output.

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

`OPTIONS` **fields**	**Description**
`tol`	relative tolerance for rank computations (Default: internally computed)
`distance`	option for the selection of the distance function $\mathrm{dist}(G_1, G_2)$ between two transfer function matrices $G_1(\lambda)$ and $G_2(\lambda)$: `'nugap'` – $\mathrm{dist}(G_1, G_2) = \delta_\nu(G_1, G_2)$, the ν-gap distance (default) `'Inf'` – $\mathrm{dist}(G_1, G_2) = \lVert G_1 - G_2 \rVert_\infty$, the $\mathcal{H}_\infty$-norm based distance `'2'` – $\mathrm{dist}(G_1, G_2) = \lVert G_1 - G_2 \rVert_2$, the $\mathcal{H}_2$-norm based distance

`MDFreq`	real vector, which contains the frequency values ω_k, $k = 1, \ldots, n_f$, for which the point-wise distances have to be computed. For each real frequency ω_k, there corresponds a complex frequency λ_k which is used to evaluate the point-wise distance. Depending on the system type, $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the common sampling time of the component models. (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: $\beta = 1.4901\cdot 10^{-08}$).
`cdinp`	option to use both control and disturbance input channels to evaluate the distances, as follows: `true` – use both control and disturbance input channels; `false` – use only the control input channels (default)

Output data

`DIST` is an N-dimensional row vector with nonnegative elements whose j-th element `DIST`(j) contains the computed distance (see `OPTIONS.distance`) between the selected input channels of `SYS` and the j-th component model `SYSM`(j) as follows:

– if `OPTIONS.MDFreq = []` and `OPTIONS.cdinp = false` then

$$\texttt{DIST}(j) = \mathrm{dist}\left(G_u(\lambda), G_u^{(j)}(\lambda)\right)$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = false` then

$$\texttt{DIST}(j) = \max_k \mathrm{dist}\left(G_u(\lambda_k), G_u^{(j)}(\lambda_k)\right)$$

– if `OPTIONS.MDFreq = []` and `OPTIONS.cdinp = true` then

$$\texttt{DIST}(j) = \mathrm{dist}\left(\left[\, G_u(\lambda)\ G_d(\lambda)\,\right], \left[\, G_u^{(j)}(\lambda)\ G_d^{(j)}(\lambda)\,\right]\right)$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = true` then

$$\texttt{DIST}(j) = \max_k \mathrm{dist}\left(\left[\, G_u(\lambda_k)\ G_d(\lambda_k)\,\right], \left[\, G_u^{(j)}(\lambda_k)\ G_d^{(j)}(\lambda_k)\,\right]\right)$$

`FPEAK` is an N-dimensional row vector, whose j-th element `FPEAK`(j) contains the peak frequency (in rad/TimeUnit), where `DIST`(j) is achieved.

`MIND` is the index ℓ of the component model for which the minimum value of the distances in `DIST` is achieved.

Method

The definition of the distances of a set of component models to a current model is given in Section 3.6.2. The evaluation of the ν-gap distances relies on the definition proposed in [18]. The pointwise distances $\text{dist}\big(G_u(\lambda_k), G_u^{(j)}(\lambda_k)\big)$ for the $\mathcal{H}_\infty$- and $\mathcal{H}_2$ norms are simply the 2-norm of the difference of the frequency responses $\text{dist}\big(G_u(\lambda_k), G_u^{(j)}(\lambda_k)\big) = \big\|G_u(\lambda_k) - G_u^{(j)}(\lambda_k)\big\|_2$. Similar formulas apply if the disturbance inputs are also selected.

Example

Example 6. This is *Example* 6.1 from the book [16], which deals with a continuous-time state-space model, describing, in the fault-free case, the lateral dynamics of an F-16 aircraft with the matrices

$$A^{(1)} = \begin{bmatrix} -0.4492 & 0.046 & 0.0053 & -0.9926 \\ 0 & 0 & 1.0000 & 0.0067 \\ -50.8436 & 0 & -5.2184 & 0.7220 \\ 16.4148 & 0 & 0.0026 & -0.6627 \end{bmatrix}, \quad B_u^{(1)} = \begin{bmatrix} 0.0004 & 0.0011 \\ 0 & 0 \\ -1.4161 & 0.2621 \\ -0.0633 & -0.1205 \end{bmatrix},$$

$$C^{(1)} = I_4, \qquad D_u^{(1)} = 0_{4\times 2}\,.$$

The four state variables are the sideslip angle, roll angle, roll rate and yaw rate, and the two input variables are the aileron deflection and rudder deflection. The model detection problem addresses the synthesis of model detection filters for the detection and identification of loss of efficiency of the two flight actuators, which control the deflections of the aileron and rudder. The individual fault models correspond to different degrees of surface efficiency degradation. A multiple model with $N = 9$ component models is used, which correspond to a two-dimensional parameter grid for N values of the parameter vector $\rho := [\rho_1, \rho_2]^T$. For each component of ρ, we employ the three grid points $\{0, 0.5, 1\}$. The component system matrices in (226) are defined for $i = 1, 2, \ldots, N$ as: $E^{(i)} = I_4$, $A^{(i)} = A^{(1)}$, $C^{(i)} = C^{(1)}$, and $B_u^{(i)} = B_u^{(1)}\Gamma^{(i)}$, where $\Gamma^{(i)} = \text{diag}\big(1-\rho_1^{(i)}, 1-\rho_2^{(i)}\big)$ and $\big(\rho_1^{(i)}, \rho_2^{(i)}\big)$ are the values of parameters (ρ_1, ρ_2) on the chosen grid:

ρ_1 :	0	0	0	0.5	0.5	0.5	1	1	1
ρ_2 :	0	0.5	1	0	0.5	1	0	0.5	1

.

For example, $\big(\rho_1^{(1)}, \rho_2^{(1)}\big) = (0,0)$ corresponds to the fault-free situation, while $\big(\rho_1^{(9)}, \rho_2^{(9)}\big) = (1,1)$ corresponds to complete failure of both control surfaces. It follows, that the TFM $G_u^{(i)}(s)$ of the i-th system can be expressed as

$$G_u^{(i)}(s) = G_u^{(1)}(s)\Gamma^{(i)}, \tag{98}$$

where

$$G_u^{(1)}(s) = C^{(1)}\big(sI - A^{(1)}\big)^{-1}B_u^{(1)}$$

is the TFM of the fault-free system. Note that $G_u^{(N)}(s) = 0$ describes the case of complete failure.

We evaluate the distances between a potential current model to the set of component models using a finer grid of the damage parameters with a length of 0.05, leading to a 441 parameter combinations. For each pair of values (ρ_1, ρ_2) on this grid we determine the current transfer function matrix

$$G_u(s) := C^{(1)}\big(sI - A^{(1)}\big)^{-1} B^{(1)} \left[\begin{smallmatrix} 1-\rho_1 & 0 \\ 0 & 1-\rho_2 \end{smallmatrix}\right]$$

and compute the distances to the component models $G_u^{(i)}(s)$ defined in (98). For the evaluation of least distances, we employ four methods. The first is simply to determine the least distance between the current values of (ρ_1, ρ_2) to the values defined in the above coarse grid. The second and third evaluations of the least distance are based on evaluating the minimum $\mathcal{H}_\infty$- or $\mathcal{H}_2$-norm of the difference $G_u^{(i)}(s) - G_u(s)$, respectively. The fourth evaluation computes the minimum of ν-gap distances $\delta_\nu\big(G_u^{(i)}(s), G_u(s)\big)$. The resulting numbers of matches of the models defined on the finer grid with those on the original coarse grid result by counting the model indices at the least distances, which are plotted in the histogram in Fig. 4. As it can be observed, there are differences between the least distances determined with different methods. While the direct comparison of parameters and the $\mathcal{H}_\infty$- and $\mathcal{H}_2$-norm based values agree reasonably well, the information provided by the computation of ν-gap distances significantly differ, showing strong preference for the 4-th model and significantly less preferences for the 7-th, 8-th, and 9-th models. The largest number of matches occurs for the 5-th model, which corresponds to $\rho_1 = 0.5$ and $\rho_2 = 0.5$.

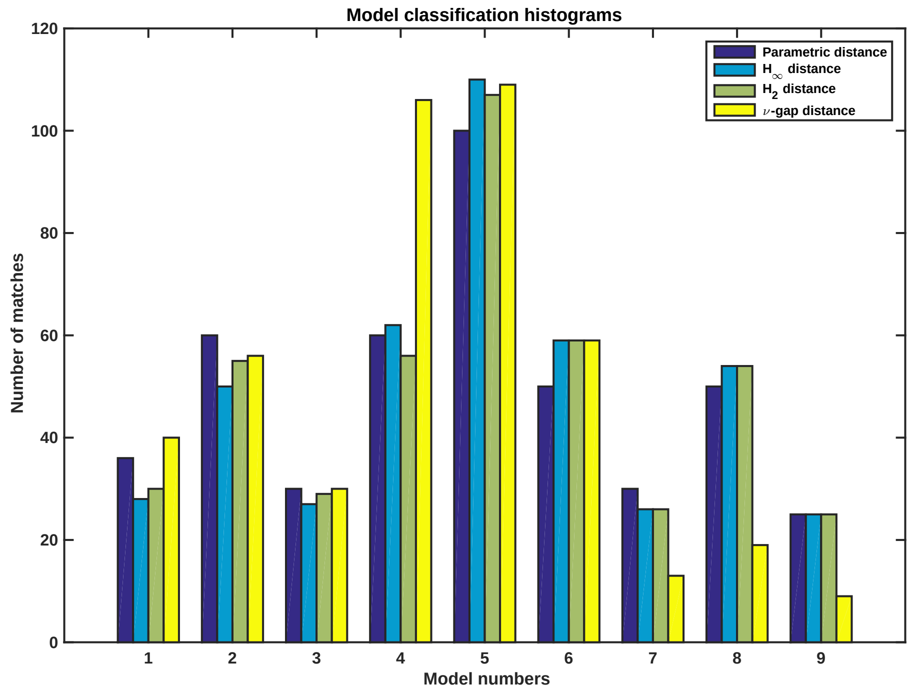

Figure 4: Classification of 441 models using different distances: parametric distance (dark blue), $\mathcal{H}_\infty$-norm (blue), $\mathcal{H}_2$-norm (green), ν-gap distance (yellow)

The following MATLAB script produces the histogram in Fig. 4.

```
% Example - Comparison of estimations of minimum distances

% Define a lateral aircraft dynamics model with
% n  = 4 states, mu = 2 control inputs, p  = 4 measurable outputs
A = [-.4492 0.046 .0053 -.9926;
       0    0     1     0.0067;
   -50.8436 0   -5.2184  .722;
    16.4148 0     .0026 -.6627];
Bu = [0.0004 0.0011; 0 0; -1.4161 .2621; -0.0633 -0.1205];
C = eye(4); p = size(C,1); mu = size(Bu,2);

% define the loss of efficiency (LOE) faults as input scaling gains
% Gamma(i,:) = [ 1-rho1(i) 1-rho2(i) ]
Gamma = 1 - [ 0  0 0 .5 .5 .5 1  1 1;
              0 .5 1  0 .5  1 0 .5 1 ]';
N = size(Gamma,1);  % number of LOE cases

% define a multiple physical fault model Gui = Gu*diag(Gamma(i,:))
sysu = ss(zeros(p,mu,N,1));
for i=1:N
    sysu(:,:,i,1) = ss(A,Bu*diag(Gamma(i,:)),C,0);
end
% setup the multiple model
sysu = mdmodset(sysu,struct('controls',1:mu));

% define fine grid and number of samples
rhogrid = 0:0.05:1; K = length(rhogrid)^2;
% perform least distance based model classification
ind = zeros(K,4);
i = 0;
for rho1 = rhogrid
    for rho2 = rhogrid
        i = i+1;
        % define actual model
        rho = [ rho1 rho2];
        sys = ss(A,Bu*diag(1-rho),C,0);
        set(sys,'InputGroup',struct('controls',1:mu));
        temp = Gamma - repmat(1-rho,N,1);
        [~,ind(i,1)]   = min(sqrt(temp(:,1).^2+temp(:,2).^2));
        [~,~,ind(i,2)] = mddist2c(sysu,sys,struct('distance','Inf'));
        [~,~,ind(i,3)] = mddist2c(sysu,sys,struct('distance','2'));
        [~,~,ind(i,4)] = mddist2c(sysu,sys);
    end
end
```

```
% plot classification histograms
hist(ind,1:N)
title('Model classification histograms')
xlabel('\bf Model numbers')
ylabel('\bf Number of matches')
legend('\bf Parametric distance','\bf H_\infty distance',...
       '\bf H_2 distance','\bf \nu-gap distance')
```

4.6 Functions for Performance Evaluation of FDI Filters

These functions address the determination of the structure matrices defined in Section 2.5 and the computation of the performance criteria of FDI filters defined in Section 2.7. All functions are fully compatible with the results computed by the synthesis functions of FDI filters described in Section 4.8.

4.6.1 `fditspec`

Syntax

```
SMAT = fditspec(R)
SMAT = fditspec(R,TOL)
SMAT = fditspec(R,TOL,FDTOL)
SMAT = fditspec(R,TOL,FDTOL,FREQ)
SMAT = fditspec(R,TOL,FDTOL,[],BLKOPT)
SMAT = fditspec(R,TOL,FDTOL,FREQ,BLKOPT)
```

Description

`fditspec` determines the weak or strong structure matrix corresponding to the fault inputs of the internal form of a FDI filter or of a collection of internal forms of FDI filters.

Input data

`R` is a LTI system or a cell array of LTI systems.

If `R` is a LTI system representing the internal form of a FDI filter, then it is in a descriptor system state-space form

$$\begin{aligned} E_R\lambda x_R(t) &= A_Rx_R(t) + B_{R_f}f(t) + B_{R_v}v(t),\\ r(t) &= C_Rx_R(t) + D_{R_f}f(t) + D_{R_v}v(t), \end{aligned} \tag{99}$$

where $r(t) \in \mathbb{R}^q$ is the residual output, $f(t) \in \mathbb{R}^{m_f}$ is the fault input and $v(t)$ contains all additional inputs. Any of the input components $f(t)$ and $v(t)$ can be void. For the fault input $f(t)$ the input group `'faults'` has to be defined. If there is no input group `'faults'` defined, then, by default, all system inputs are considered faults and the auxiliary input is assumed void. The input-output form of `R` corresponding to (99) is

$$\mathbf{r}(\lambda) = R_f(\lambda)\mathbf{f}(\lambda) + R_v(\lambda)\mathbf{v}(\lambda), \tag{100}$$

where $R_f(\lambda)$ and $R_v(\lambda)$ are the transfer function matrices from the corresponding inputs, respectively. If `R` is specified in the input-output representation (100), then it is automatically converted to an equivalent minimal order state-space form as in (99).

If `R` is a $N \times 1$ cell array of LTI systems representing the internal forms of N FDI filters, then the i-th component system `R{`i`}` is in the state-space form

$$\begin{aligned} E_R^{(i)}\lambda x_R^{(i)}(t) &= A_R^{(i)} x_R^{(i)}(t) + B_{R_f}^{(i)} f(t) + B_{R_v}^{(i)} v(t), \\ r^{(i)}(t) &= C_R^{(i)} x_R^{(i)}(t) + D_{R_f}^{(i)} f(t) + D_{R_v}^{(i)} v(t), \end{aligned} \tag{101}$$

where $r^{(i)}(t) \in \mathbb{R}^{q^{(i)}}$ is the i-th residual component, $f(t) \in \mathbb{R}^{m_f}$ is the fault input and $v(t)$ contains the rest of inputs. Any of the input components $f(t)$ and $v(t)$ can be void. For the fault input $f(t)$ the input group `'faults'` has to be defined, while $v(t)$ includes any other (not relevant) system inputs. If there is no input group `'faults'` defined, then, by default, all system inputs are considered faults and the auxiliary input is assumed void. All component systems `R{`i`}`, $i = 1, \ldots, N$, must have the same number of fault inputs and the same sampling time. The input-output form of `R{`i`}` corresponding to (101) is

$$\mathbf{r}^{(i)}(\lambda) = R_f^{(i)}(\lambda)\mathbf{f}(\lambda) + R_v^{(i)}(\lambda)\mathbf{v}(\lambda), \tag{102}$$

where $R_f^{(i)}(\lambda)$ and $R_v^{(i)}(\lambda)$ are the transfer function matrices from the corresponding inputs, respectively. If `R{`i`}` is specified in the input-output representation (102), then it is automatically converted to an equivalent minimal order state-space form as in (101).

`TOL` is a relative tolerance used for controllability tests. A default value is internally computed if `TOL` ≤ 0 or is not specified at input.

`FDTOL` is an absolute threshold for the magnitudes of the zero elements in the system matrices B_{R_f}, C_R and D_{R_f}, in the case of model (99), or in the matrices $B_{R_f}^{(i)}$, $C_R^{(i)}$ and $D_{R_f}^{(i)}$, in the case of models of the form (101). Any element of these matrices whose magnitude does not exceed `FDTOL` is considered zero. Additionally, if `FREQ` is nonempty, `FDTOL` is also used for the singular-value-based rank tests performed on the system matrix $\left[\begin{smallmatrix} A_R-\lambda E_R & B_{R_f} \\ C_R & D_{R_f} \end{smallmatrix}\right]$, in the case of model (99), or on the system matrices $\left[\begin{smallmatrix} A_R^{(i)}-\lambda E_R^{(i)} & B_{R_f}^{(i)} \\ C_R^{(i)} & D_{R_f}^{(i)} \end{smallmatrix}\right]$ for $i = 1, \ldots, N$, in the case of models of the form (101). If `FDTOL` ≤ 0 or not specified at input, the default value `FDTOL` $= 10^{-4}\max\left(1, \|B_{R_f}\|_1, \|C_R\|_\infty, \|D_{R_f}\|_1\right)$ is used in the case of model (99). In the case of models of the form (101), the default value `FDTOL` $= 10^{-4}\max\left(1, \|B_{R_f}^{(i)}\|_1, \|C_R^{(i)}\|_\infty, \|D_{R_f}^{(i)}\|_1\right)$ is used for handling the i-th model (101). If `FDTOL` ≤ 0 and if `FREQ` is nonempty, the default value `FDTOL` $= 10^{-4}\max\left(1, \left\| \left[\begin{smallmatrix} A_R & B_{R_f} \\ C_R & D_{R_f} \end{smallmatrix}\right] \right\|_1, \|E_R\|_1\right)$

is used for the rank tests on the system matrix in the case of model (99). In the case of models of the form (101), the default value $\texttt{FDTOL} = 10^{-4}\max\left(1, \left\|\begin{bmatrix} A_R^{(i)} & B_{R_f}^{(i)} \\ C_R^{(i)} & D_{R_f}^{(i)} \end{bmatrix}\right\|_1, \|E_R^{(i)}\|_1\right)$ is used for handling the system matrix of the i-th model (101).

`FREQ` is a real vector, which contains the frequency values ω_k, $k = 1, \ldots, n_f$, to be used to check for zeros of the individual (rational) elements or individual columns of the transfer function matrix $R_f(\lambda)$ in the case of model (99), or of the transfer function matrices $R_f^{(i)}(\lambda)$ in the case of models (101). By default, `FREQ` is empty if it is not specified. For each real frequency ω_k contained in `FREQ`, there corresponds a complex frequency λ_k, which is used to check the elements or columns of the respective transfer function matrices to have λ_k as zero. Depending on the system type, $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the sampling time of the system.

`BLKOPT` is a character variable to be set to `'block'` to specify the block-structure based evaluation option of the structure matrix.

Output data

`SMAT` is a logical array which contains the resulting structure matrix.

In the case of a LTI system `R`, `SMAT` is determined depending on the selected option `BLKOPT` and the frequency values specified in `FREQ`, as follows:

- If `BLKOPT` is not specified (or empty), then `SMAT` is the structure matrix corresponding to the zero and nonzero elements of the transfer function matrix $R_f(\lambda)$:
  - If `FREQ` is empty or not specified at input, then `SMAT` is a $q \times m_f$ logical array, which contains the *weak* structure matrix corresponding to the zero and nonzero elements of $R_f(\lambda)$ (see (14) for the definition of the weak structure matrix). Accordingly, $\texttt{SMAT}(i,j) = \texttt{true}$, if the (i,j)-th element of $R_f(\lambda)$ is nonzero. Otherwise, $\texttt{SMAT}(i,j) = \texttt{false}$.
  - If `FREQ` is nonempty, then `SMAT` is a $q \times m_f \times n_f$ logical array which contains in the k-th page $\texttt{SMAT}(:,:,k)$, the *strong* structure matrix corresponding to the presence or absence of zeros of the elements of $R_f(\lambda)$ in the complex frequency λ_k corresponding to the k-th real frequency ω_k contained in `FREQ` (see description of `FREQ`) (see also (18) for the definition of the strong structure matrix at a complex frequency λ_k). Accordingly, $\texttt{SMAT}(i,j,k) = \texttt{true}$, if the (i,j)-th element of $R_f(\lambda)$ has no zero in λ_k. Otherwise, $\texttt{SMAT}(i,j,k) = \texttt{false}$.
- If `BLKOPT = 'block'` is specified, then `SMAT` is the structure (row) vector corresponding to the zero and nonzero columns of the transfer function matrix $R_f(\lambda)$:
  - If `FREQ` is empty or not specified at input, then `SMAT` is a $1 \times m_f$ logical (row) vector, which contains the *weak* structure matrix corresponding to the zero and nonzero columns of $R_f(\lambda)$ (see (14) for the block-structured definition of the weak structure matrix). Accordingly, $\texttt{SMAT}(1,j) = \texttt{true}$, if the j-th column of $R_f(\lambda)$ is nonzero. Otherwise, $\texttt{SMAT}(1,j) = \texttt{false}$.

  - If `FREQ` is nonempty, then `SMAT` is a $1 \times m_f \times n_f$ logical array which contains in the k-th page `SMAT`$(:,:,k)$, the *strong* structure vector corresponding to the presence or absence of a zero in the columns of $R_f(\lambda)$ in the complex frequency λ_k corresponding to the k-th real frequency ω_k contained in `FREQ` (see description of `FREQ`). Accordingly, `SMAT`$(1,j,k)$ = `true`, if the j-th column of $R_f(\lambda)$ has no zero in the complex frequency λ_k. Otherwise, `SMAT`$(1,j,k)$ = `false`.

In the case of a cell array of LTI systems `R`$\{i\}$, $i = 1,\ldots,N$, `SMAT` is determined depending on the frequency values specified in `FREQ`, as follows:

- If `FREQ` is empty or not specified at input, then `SMAT` is an $N \times m_f$ logical matrix, whose i-th row contains the *weak* structure vector corresponding to the zero and nonzero columns of the transfer function matrix $R_f^{(i)}(\lambda)$ (see (14) for the block-structured definition of the weak structure matrix). Accordingly, `SMAT`(i,j) = `true`, if the j-th column of $R_f^{(i)}(\lambda)$ is nonzero. Otherwise, `SMAT`(i,j) = `false`. All entries of the i-th row of `SMAT` are set to `false` if `R`$\{i\}$ is empty.
- If `FREQ` is nonempty, then `SMAT` is a $N \times m_f \times n_f$ logical array which contains in the i-th row of the k-th page `SMAT`$(:,:,k)$, the *strong* structure (row) vector corresponding to the presence or absence of a zero in the columns of $R_f^{(i)}(\lambda)$ in the complex frequency λ_k corresponding to the k-th real frequency ω_k contained in `FREQ` (see description of `FREQ`) (see also (18) for the definition of the strong structure matrix at a complex frequency λ_k). Accordingly, `SMAT`(i,j,k) = `true`, if the j-th column of $R_f^{(i)}(\lambda)$ has no zero in λ_k. Otherwise, `SMAT`(i,j,k) = `false`. All entries of the i-th row of `SMAT` are set to `false` if `R`$\{i\}$ is empty.

Method

We first describe the implemented analysis method for the case when `R` is a LTI system in a state-space form as in (99) and $R_f(\lambda)$ is the transfer function matrix from the fault inputs to the residual output as in the input-output model (100). For the definition of the weak and strong structure matrices, see Section 2.5. For the determination of the weak structure matrix, controllable realizations are determined for each column of $R_f(\lambda)$ and tests are performed to identify the nonzero elements in the respective column of $R_f(\lambda)$ by using [16, Corollary 7.1] in a controllability related dual formulation. The block-structure based evaluation is based on the input observability tests of [16, Corollary 7.1], performed for the controllable realizations of each column of $R_f(\lambda)$.

For the determination of the strong structure matrix, minimal realizations are determined for each element of $R_f(\lambda)$ and the absence of zeros is assessed by checking the full rank of the corresponding system matrix for all complex frequencies corresponding to the real frequencies specified in `FREQ` (see [16, Corollary 7.2]). For the block-structure based evaluation, controllable realizations are determined for each column of $R_f(\lambda)$ and the test used in [16, Corollary 7.2] is employed.

For the case when `R` is a cell array containing N LTI systems in state-space forms as in (101) and $R_f^{(i)}(\lambda)$ is the transfer function matrix of i-th LTI system `R`$\{i\}$ from the fault inputs to the i-th residual component as in the input-output models (102), the block-structure based evaluations, described above, are employed for each $R_f^{(i)}(\lambda)$ to determine the corresponding i-th row of the structure matrix `SMAT`.

4.6.2 `fdisspec`

Syntax

```
[SMAT,GAINS] = fdisspec(R)
[SMAT,GAINS] = fdisspec(R,FDGAINTOL)
[SMAT,GAINS] = fdisspec(R,FDGAINTOL,FREQ)
[SMAT,GAINS] = fdisspec(R,FDGAINTOL,FREQ,BLKOPT)
```

Description

`fdisspec` determines the strong structure matrix corresponding to the fault inputs of the internal form of a FDI filter or of a collection of internal forms of FDI filters.

Input data

`R` is a LTI system or a cell array of LTI systems.

If `R` is a LTI system representing the internal form of a FDI filter, then it is in a descriptor system state-space form

$$\begin{aligned} E_R\lambda x_R(t) &= A_R x_R(t) + B_{R_f} f(t) + B_{R_v} v(t), \\ r(t) &= C_R x_R(t) + D_{R_f} f(t) + D_{R_v} v(t), \end{aligned} \tag{103}$$

where $r(t) \in \mathbb{R}^q$ is the residual output, $f(t) \in \mathbb{R}^{m_f}$ is the fault input and $v(t)$ contains all additional inputs. Any of the input components $f(t)$ and $v(t)$ can be void. For the fault input $f(t)$ the input group `'faults'` has to be defined. If there is no input group `'faults'` defined, then, by default, all system inputs are considered faults and the auxiliary input is assumed void. The input-output form of `R` corresponding to (103) is

$$\mathbf{r}(\lambda) = R_f(\lambda)\mathbf{f}(\lambda) + R_v(\lambda)\mathbf{v}(\lambda), \tag{104}$$

where $R_f(\lambda)$ and $R_v(\lambda)$ are the transfer function matrices from the corresponding inputs. If `R` is specified in the input-output representation (104), then it is automatically converted to an equivalent minimal order state-space form as in (103).

If `R` is an $N \times 1$ cell array of LTI systems representing the internal forms of N FDI filters, then the i-th component system `R`$\{i\}$ is in the state-space form

$$\begin{aligned} E_R^{(i)}\lambda x_R^{(i)}(t) &= A_R^{(i)} x_R^{(i)}(t) + B_{R_f}^{(i)} f(t) + B_{R_v}^{(i)} v(t), \\ r^{(i)}(t) &= C_R^{(i)} x_R^{(i)}(t) + D_{R_f}^{(i)} f(t) + D_{R_v}^{(i)} v(t), \end{aligned} \tag{105}$$

where $r^{(i)}(t) \in \mathbb{R}^{q^{(i)}}$ is the i-th residual component, $f(t) \in \mathbb{R}^{m_f}$ is the fault input and $v(t)$ contains the rest of inputs. Any of the input components $f(t)$ and $v(t)$ can be void. For the fault input $f(t)$ the input group `'faults'` has to be defined, while $v(t)$ includes any other (not-relevant) system inputs. If there is no input group `'faults'` defined, then, by default, all system inputs are considered faults and the auxiliary input is assumed void. All component systems `R{`i`}`, $i = 1, \ldots, N$, must have the same number of fault inputs and the same sampling time. The input-output form of `R{`i`}` corresponding to (105) is

$$\mathbf{r}^{(i)}(\lambda) = R_f^{(i)}(\lambda)\mathbf{f}(\lambda) + R_v^{(i)}(\lambda)\mathbf{v}(\lambda), \tag{106}$$

where $R_f^{(i)}(\lambda)$ and $R_v^{(i)}(\lambda)$ are the transfer function matrices from the corresponding inputs. If `R{`i`}` is specified in the input-output representation (106), then it is automatically converted to an equivalent minimal order state-space form as in (105).

`FDGAINTOL` is a threshold for the magnitudes of the frequency-response gains of the transfer function matrix $R_f(\lambda)$ in the case of model (103), or of the transfer function matrices $R_f^{(i)}(\lambda)$ in the case of models (105). If `FDGAINTOL` $= 0$ or not specified at input, the default value `FDGAINTOL` $= 0.01$ is used.

`FREQ` is a real vector, which contains the real frequency values ω_k, $k = 1, \ldots, n_f$, to be used to check for the existence of zeros of the (rational) elements or columns of the transfer function matrix $R_f(\lambda)$ in the case of model (99), or of the transfer function matrices $R_f^{(i)}(\lambda)$ in the case of models (101). By default, `FREQ` $= 0$, if it is empty or not specified. For each real frequency ω_k contained in `FREQ`, there corresponds a complex frequency λ_k which is used to evaluate $R_f(\lambda_k)$, to check that the elements or columns of the respective transfer function matrices have λ_k as a zero. Depending on the system type, $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the sampling time of the system. The complex frequencies corresponding to the real frequencies specified in `FREQ` must be disjoint from the set of poles of $R_f(\lambda)$.

`BLKOPT` is a character variable to be set to `'block'` to specify the block-structure based evaluation option of the structure matrix.

Output data

`SMAT` is a logical array which contains the resulting structure matrix.

In the case `R` is a LTI system, `SMAT` is determined depending on the selected option `BLKOPT`, as follows:

- If `BLKOPT` is not specified (or empty), then `SMAT` is a $q \times m_f \times n_f$ logical array which contains in the k-th page `SMAT`$(:,:,k)$, the *strong* structure matrix corresponding to the presence or absence of zeros of the elements of $R_f(\lambda)$ in the complex frequency λ_k corresponding to the k-th real frequency ω_k contained in `FREQ` (see description of `FREQ`) (see also (18) for the definition of the strong structure matrix at a complex frequency λ_k). Accordingly, `SMAT`(i,j,k) = `true`, if the magnitude of the (i,j)-th element of $R_f(\lambda_k)$ is greater than or equal to `FDGAINTOL`. Otherwise, `SMAT`(i,j,k) = `false`.

- If `BLKOPT = 'block'` is specified, then `SMAT` is a $1 \times m_f \times n_f$ logical array which contains in the k-th page `SMAT`$(:,:,k)$, the *strong* structure (row) vector corresponding to the presence or absence of a zero in the columns of $R_f(\lambda)$ in the complex frequency λ_k corresponding to the k-th real frequency ω_k contained in `FREQ` (see description of `FREQ`). Accordingly, `SMAT`$(1,j,k) =$ `true`, if the norm of the j-th column of $R_f(\lambda_k)$ is greater than or equal to `FDGAINTOL` (i.e., $\big\|R_{f_j}(\lambda_k)\big\|_2 \geq$ `FDGAINTOL`). Otherwise, `SMAT`$(1,j,k) =$ `false`.

In the case `R` is an $N \times 1$ cell array of LTI systems, `SMAT` is an $N \times m_f \times n_f$ logical array which contains in the i-th row of the k-th page `SMAT`$(:,:,k)$, the *strong* structure (row) vector corresponding to the presence or absence of a zero in the columns of $R_f^{(i)}(\lambda)$ in the complex frequency λ_k corresponding to the k-th real frequency ω_k contained in `FREQ` (see description of `FREQ`) (see also (18) for the definition of the strong structure matrix at a complex frequency λ_k). Accordingly, `SMAT`$(i,j,k) =$ `true`, if the norm of the j-th column of $R_f^{(i)}(\lambda_k)$ is greater than or equal to `FDGAINTOL` (i.e., $\big\|R_{f_j}^{(i)}(\lambda_k)\big\|_2 \geq$ `FDGAINTOL`). Otherwise, `SMAT`$(i,j,k) =$ `false`. If `R`$\{i\}$ is empty, then all entries of the i-th row of `SMAT` are set to `false`.

`GAINS` is a real nonnegative array.

In the case `R` is a LTI system, `GAINS` is determined depending on the selected option `BLKOPT`, as follows:

- If `BLKOPT` is not specified (or empty), then `GAINS` is $q \times m_f$ matrix, whose (i,j)-th element contains the minimum value of the frequency-response gains of the (i,j)-th element of $R_f(\lambda)$ evaluated over all complex frequencies corresponding to `FREQ` (see description of `FREQ`). This value is a particular instance of the $\mathcal{H}_-$-index of the (i,j)-th element of $R_f(\lambda)$.
- If `BLKOPT = 'block'` is specified, then `GAINS` is a m_f-dimensional row vector, whose j-th element contains the minimum of the norms of the frequency responses of the j-th column of $R_f(\lambda)$ evaluated over all complex frequencies corresponding to `FREQ` (see description of `FREQ`). This value is a particular instance of the $\mathcal{H}_-$-index of the j-th column of $R_f(\lambda)$.

In the case `R` is an $N \times 1$ cell array of LTI systems, `GAINS` is an $N \times m_f$ matrix, whose (i,j)-th element contains the minimum of the norms of the frequency responses of the j-th column of $R_f^{(i)}(\lambda)$ evaluated over all complex frequencies corresponding to `FREQ` (see description of `FREQ`). If `R`$\{i\}$ is empty, then all entries of the i-th row of `GAINS` are set to zero.

Method

For the definition of the strong structure matrix at a given frequency, see Remark 3 of Section 2.5. For the case when `R` is a LTI system in a state-space form as in (103) and $R_f(\lambda)$ is the transfer function matrix from the fault inputs to the residual output as in the input-output model (104), the element-wise or column-wise (if `BLKOPT = 'block'`) evaluations of the $\mathcal{H}_-$-index on a discrete set of frequency values (see [16, Section 5.3]) are employed for each element or, respectively, each column of $R_f(\lambda)$, to determine the corresponding element of the structure matrix `SMAT` and associated `GAINS`. The resulting entries of `GAINS` correspond to an element-wise or column-wise evaluation of the $\mathcal{H}_-$-index on a discrete set of frequency values (see [16, Section 5.3]).

For the case when `R` is a cell array containing N LTI systems in state-space forms as in (105) and $R_f^{(i)}(\lambda)$ is the transfer function matrix of i-th LTI system `R`$\{i\}$ from the fault inputs to the i-th residual component as in the input-output models (106), the column-wise evaluations of the $\mathcal{H}_-$-index on a discrete set of frequency values (see [16, Section 5.3]) are employed for each $R_f^{(i)}(\lambda)$ to determine the corresponding i-row of the structure matrix `SMAT` and the associated i-th row of `GAINS`.

4.6.3 `fdifscond`

Syntax

```
FSCOND = fdifscond(R)
FSCOND = fdifscond(R,FREQ)
FSCOND = fdifscond(R,[],S)
FSCOND = fdifscond(R,FREQ,S)
[BETA,GAMMA] = fdifscond(...)
```

Description

`fdifscond` evaluates the fault sensitivity condition of the internal form of a FDI filter or the fault sensitivity conditions of the internal forms of a collection of FDI filters.

Input data

`R` is a LTI system or a cell array of LTI systems.

If `R` is a LTI system representing the internal form of a FDI filter, then it is in a descriptor system state-space form

$$\begin{aligned} E_R\lambda x_R(t) &= A_R x_R(t) + B_{R_f} f(t) + B_{R_v} v(t),\\ r(t) &= C_R x_R(t) + D_{R_f} f(t) + D_{R_v} v(t), \end{aligned} \tag{107}$$

where $r(t) \in \mathbb{R}^q$ is the residual output, $f(t) \in \mathbb{R}^{m_f}$ is the fault input, and $v(t)$ contains all additional inputs. For the fault input $f(t)$ the input group `'faults'` has to be defined. The input-output form of `R` corresponding to (107) is

$$\mathbf{r}(\lambda) = R_f(\lambda)\mathbf{f}(\lambda) + R_v(\lambda)\mathbf{v}(\lambda), \tag{108}$$

where $R_f(\lambda)$ and $R_v(\lambda)$ are the transfer function matrices from the corresponding inputs.

If `R` is an $N \times 1$ cell array of LTI systems representing the internal forms of N FDI filters, then the i-th component system `R`$\{i\}$ is in the state-space form

$$\begin{aligned} E_R^{(i)} \lambda x_R^{(i)}(t) &= A_R^{(i)} x_R^{(i)}(t) + B_{R_f}^{(i)} f(t) + B_{R_v}^{(i)} v(t), \\ r^{(i)}(t) &= C_R^{(i)} x_R^{(i)}(t) + D_{R_f}^{(i)} f(t) + D_{R_v}^{(i)} v(t), \end{aligned} \tag{109}$$

where $r^{(i)}(t) \in \mathbb{R}^{q^{(i)}}$ is the i-th residual component, $f(t) \in \mathbb{R}^{m_f}$ is the fault input and $v(t)$ contains the rest of inputs. For the fault input $f(t)$ the input group `'faults'` has to be defined. The input-output form of `R`$\{i\}$ corresponding to (109) is

$$\mathbf{r}^{(i)}(\lambda) = R_f^{(i)}(\lambda)\mathbf{f}(\lambda) + R_v^{(i)}(\lambda)\mathbf{v}(\lambda), \tag{110}$$

where $R_f^{(i)}(\lambda)$ and $R_v^{(i)}(\lambda)$ are the transfer function matrices from the corresponding inputs.

`FREQ` is a real vector, which contains the frequency values ω_k, $k = 1, \ldots, n_f$, to be used to evaluate the fault condition number of the transfer function matrix $R_f(\lambda)$ in (108), or of the transfer function matrices $R_f^{(i)}(\lambda)$ in (110). By default, `FREQ` is empty if it is not specified. For each real frequency ω_k contained in `FREQ`, there corresponds a complex frequency λ_k, which is used to define the complex set $\Omega = \{\lambda_1, \ldots, \lambda_{n_f}\}$. Depending on the system type, $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the sampling time of the system.

`S` is a $q \times m_f$ logical structure matrix if `R` is the internal form of a FDI filter with q residual outputs or is an $N \times m_f$ logical structure matrix if `R` is a collection of N internal forms of fault detection and isolation filters, where `R`$\{i\}$ is the internal form of the i-th filter.

Output data

`FSCOND` is the computed fault sensitivity condition, which is either a scalar or a vector, depending on the input variables.

In the case of calling `fdifscond` as

```
FSCOND = fdifscond(R)
```

then:

- If `R` is a LTI system, then `FSCOND` is the fault sensitivity condition computed as β/γ, where $\beta = \|R_f(\lambda)\|_{\infty-}$ and $\gamma = \max\limits_j \|R_{f_j}(\lambda)\|_\infty$;
- If `R` is a collection of N LTI systems, then `FSCOND` is an N-dimensional vector of fault sensitivity conditions, with `FSCOND`$\{i\}$, the i-th fault sensitivity condition, computed as β_i/γ_i, where $\beta_i = \|R_f^{(i)}(\lambda)\|_{\infty-}$ and $\gamma_i = \max\limits_j \|R_{f_j}^{(i)}(\lambda)\|_\infty$. If `R`$\{i\}$ is empty, then `FSCOND`$\{i\}$ is set to `NaN`.

In the case of calling `fdifscond` as

```
FSCOND = fdifscond(R,FREQ)
```

where `FREQ` is a nonempty vector of real frequencies, which define the set of complex frequencies Ω (see description of `FREQ`), then:

- If `R` is a LTI system, then `FSCOND` is the fault sensitivity condition computed as β/γ, where $\beta = \big\|R_f(\lambda)\big\|_{\Omega-}$ and $\gamma = \max\limits_j \big\{ \sup\limits_{\lambda_s \in \Omega} \big\|R_{f_j}(\lambda_s)\big\|_2 \big\}$;
- If `R` is a collection of N LTI systems, then `FSCOND` is an N-dimensional vector of fault sensitivity conditions, with `FSCOND`$\{i\}$, the i-th fault sensitivity condition, computed as β_i/γ_i, where $\beta_i = \big\|R_f^{(i)}(\lambda)\big\|_{\Omega-}$ and $\gamma_i = \max\limits_j \big\{ \sup\limits_{\lambda_s \in \Omega} \big\|R_{f_j}^{(i)}(\lambda_s)\big\|_2 \big\}$. If `R`$\{i\}$ is empty, then `FSCOND`$\{i\}$ is set to `NaN`.

In the case of calling `fdifscond` as

```
FSCOND = fdifscond(R,[],S)
```

then:

- If `R` is a LTI system with q residual outputs and `S` is a $q \times m_f$ structure matrix, then `FSCOND` is a q-dimensional vector of fault sensitivity conditions. `FSCOND`$\{i\}$ is the fault sensitivity condition of the i-th row $R_f^{(i)}(\lambda)$ of $R_f(\lambda)$, computed as β_i/γ_i, with $\beta_i = \big\|R_{f^{(i)}}^{(i)}(\lambda)\big\|_{\infty-}$ and $\gamma_i = \max\limits_j \big\|R_{f_j}^{(i)}(\lambda)\big\|_\infty$, where $R_{f^{(i)}}^{(i)}(\lambda)$ is formed from the elements of $R_f^{(i)}(\lambda)$ which correspond to `true` values in the i-th row of `S` and $R_{f_j}^{(i)}(\lambda)$ is the (i,j)-th element of $R_f(\lambda)$;
- If `R` is a collection of N LTI systems and `S` is an $N \times m_f$ structure matrix, then `FSCOND` is an N-dimensional vector of fault sensitivity conditions, with `FSCOND`$\{i\}$, the i-th fault sensitivity condition, computed as β_i/γ_i, with $\beta_i = \big\|R_{f^{(i)}}^{(i)}(\lambda)\big\|_{\infty-}$ and $\gamma_i = \max\limits_j \big\|R_{f_j}^{(i)}(\lambda)\big\|_\infty$, where $R_{f^{(i)}}^{(i)}(\lambda)$ is formed from the columns of $R_f^{(i)}(\lambda)$ which correspond to `true` values in the i-th row of `S` and $R_{f_j}^{(i)}(\lambda)$ is the j-th column of $R_f^{(i)}(\lambda)$. If `R`$\{i\}$ is empty, then `FSCOND`$\{i\}$ is set to `NaN`.

In the case of calling `fdifscond` as

```
FSCOND = fdifscond(R,FREQ,S)
```

where both `FREQ` and `S` are nonempty then:

- If `R` is a LTI system with q residual outputs and `S` is a $q \times m_f$ structure matrix, then `FSCOND` is a q-dimensional vector of fault sensitivity conditions. `FSCOND`$\{i\}$ is the fault sensitivity condition of the i-th row $R_f^{(i)}(\lambda)$ of $R_f(\lambda)$, computed as β_i/γ_i, with $\beta_i = \big\|R_{f^{(i)}}^{(i)}(\lambda)\big\|_{\Omega-}$ and $\gamma_i = \max\limits_j \big\{ \sup\limits_{\lambda_s \in \Omega} \big\|R_{f_j}^{(i)}(\lambda_s)\big\|_2 \big\}$, where $R_{f^{(i)}}^{(i)}(\lambda)$ is formed from the elements of $R_f^{(i)}(\lambda)$ which correspond to `true` values in the i-th row of `S` and $R_{f_j}^{(i)}(\lambda)$ is the (i,j)-th element of $R_f(\lambda)$;

- If `R` is a collection of N LTI systems and `S` is an $N \times m_f$ structure matrix, then `FSCOND` is an N-dimensional vector of fault sensitivity conditions. `FSCOND`$\{i\}$ is the fault sensitivity condition of the i-th system `R`$\{i\}$, computed as β_i/γ_i, with $\beta_i = \big\| R_{f^{(i)}}^{(i)}(\lambda) \big\|_{\Omega-}$ and $\gamma_i = \max_j \big\{ \sup_{\lambda_s \in \Omega} \big\| R_{f_j}^{(i)}(\lambda_s) \big\|_2 \big\}$, where $R_{f^{(i)}}^{(i)}(\lambda)$ is formed from the columns of $R_f^{(i)}(\lambda)$ which correspond to `true` values in the i-th row of `S` and $R_{f_j}^{(i)}(\lambda)$ is the j-th column of $R_f^{(i)}(\lambda)$. If `R`$\{i\}$ is empty, then `FSCOND`$\{i\}$ is set to `NaN`.

In the case of calling `fdifscond` as

```
[BETA,GAMMA] = fdifscond(...)
```

then:

If `BETA` and `GAMMA` are scalar values, then they contain the values of β and γ, respectively, whose ratio represents the fault sensitivity condition. If `BETA` and `GAMMA` are vectors, then `BETA`$\{i\}$ and `GAMMA`$\{i\}$ contain the values of β_i and γ_i, respectively, whose ratio represents the i-th fault sensitivity condition. If `R`$\{i\}$ is empty, then `BETA`$\{i\}$ and `GAMMA`$\{i\}$ are set to zero.

Method

The definitions of the fault sensitivity condition in terms of the $\mathcal{H}_{\infty -}$-index and its finite frequency counterpart, the $\mathcal{H}_{\Omega -}$-index, are given in Section 2.7.1.

4.6.4 `fdif2ngap`

Syntax

```
GAP = fdif2ngap(R)
GAP = fdif2ngap(R,FREQ)
GAP = fdif2ngap(R,[],S)
GAP = fdif2ngap(R,FREQ,S)
[BETA,GAMMA] = fdif2ngap(...)
```

Description

`fdif2ngap` evaluates the fault-to-noise gap of the internal form of a FDI filter or the fault-to-noise gaps of the internal forms of a collection of FDI filters.

Input data

`R` is a LTI system or a cell array of LTI systems.

If `R` is a LTI system representing the internal form of a FDI filter, then it is in a descriptor system state-space form

$$\begin{aligned} E_R\lambda x_R(t) &= A_Rx_R(t) + B_{R_f}f(t) + B_{R_w}w(t) + B_{R_v}v(t),\\ r(t) &= C_Rx_R(t) + D_{R_f}f(t) + D_{R_w}w(t) + D_{R_v}v(t), \end{aligned} \tag{111}$$

where $r(t) \in \mathbb{R}^q$ is the residual output, $f(t) \in \mathbb{R}^{m_f}$ is the fault input, $w(t) \in \mathbb{R}^{m_w}$ is the noise input and $v(t)$ is the auxiliary input. For the fault input $f(t)$ and noise input $w(t)$ the input groups `'faults'` and `'noise'`, respectively, have to be defined. The input-output form of `R` corresponding to (111) is

$$\mathbf{y}(\lambda) = G_f(\lambda)\mathbf{f}(\lambda) + G_w(\lambda)\mathbf{w}(\lambda) + G_v(\lambda)\mathbf{v}(\lambda), \tag{112}$$

where $R_f(\lambda)$, $R_w(\lambda)$ and $R_v(\lambda)$ are the transfer function matrices from the corresponding inputs.

If `R` is an $N \times 1$ cell array of LTI systems representing the internal forms of N FDI filters, then the i-th component system `R`$\{i\}$ is in the state-space form

$$\begin{aligned} E_R^{(i)}\lambda x_R^{(i)}(t) &= A_R^{(i)}x_R^{(i)}(t) + B_{R_f}^{(i)}f(t) + B_{R_w}^{(i)}w(t),\\ r^{(i)}(t) &= C_R^{(i)}x_R^{(i)}(t) + D_{R_f}^{(i)}f(t) + D_{R_w}^{(i)}w(t), \end{aligned} \tag{113}$$

where $r^{(i)}(t) \in \mathbb{R}^{q^{(i)}}$ is the i-th residual component, $f(t) \in \mathbb{R}^{m_f}$ is the fault input and $w(t)$ is the noise input. For the fault input $f(t)$ and noise input $w(t)$ the input groups `'faults'` and `'noise'`, respectively, have to be defined. The input-output form of `R`$\{i\}$ corresponding to (113) is

$$\mathbf{r}^{(i)}(\lambda) = R_f^{(i)}(\lambda)\mathbf{f}(\lambda) + R_w^{(i)}(\lambda)\mathbf{w}(\lambda), \tag{114}$$

where $R_f^{(i)}(\lambda)$ and $R_w^{(i)}(\lambda)$ are the transfer function matrices from the corresponding inputs.

`FREQ` is a real vector, which contains the frequency values ω_k, $k = 1, \ldots, n_f$, to be used to evaluate the gap between the transfer function matrices $R_f(\lambda)$ and $R_w(\lambda)$ in (112), or between the transfer function matrices $R_f^{(i)}(\lambda)$ and $R_w^{(i)}(\lambda)$ in (114). By default, `FREQ` is empty if it is not specified. For each real frequency ω_k contained in `FREQ`, there corresponds a complex frequency λ_k, which is used to define the complex set $\Omega = \{\lambda_1, \ldots, \lambda_{n_f}\}$. Depending on the system type, $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the sampling time of the system.

`S` is a $q \times m_f$ logical structure matrix if `R` is the internal form of fault detection filter with q residual outputs or is an $N \times m_f$ logical structure matrix if `R` is a collection of N internal forms of fault detection and isolation filters, where `R`$\{i\}$ is the internal form of the i-th filter.

Output data

`GAP` is the computed fault-to-noise gap, which is either a scalar or a vector, depending on the input variables.

In the case of calling `fdif2ngap` as

```
GAP = fdif2ngap(R)
```

then:

- If `R` is a LTI system, then `GAP` is the fault-to-noise gap computed as β/γ, where $\beta = \|R_f(\lambda)\|_{\infty-}$ and $\gamma = \|R_w(\lambda)\|_\infty$;
- If `R` is a collection of N LTI systems, then `GAP` is an N-dimensional vector, with `GAP`$\{i\}$, the i-th fault-to-noise gap, computed as β_i/γ_i, where $\beta_i = \|R_f^{(i)}(\lambda)\|_{\infty-}$ and $\gamma_i = \|R_w^{(i)}(\lambda)\|_\infty$. If `R`$\{i\}$ is empty, then `GAP`$\{i\}$ is set to `NaN`.

In the case of calling `fdif2ngap` as

```
GAP = fdif2ngap(R,FREQ)
```

where `FREQ` is a nonempty vector of real frequencies, which defines the set of complex frequencies Ω (see description of `FREQ`), then:

- If `R` is a LTI system, then `GAP` is the fault-to-noise gap computed as β/γ, where $\beta = \|R_f(\lambda)\|_{\Omega-}$ and $\gamma = \|R_w(\lambda)\|_\infty$;
- If `R` is a collection of N LTI systems, then `GAP` is an N-dimensional vector, with `GAP`$\{i\}$, the i-th fault-to-noise gap, computed as β_i/γ_i, where $\beta_i = \|R_f^{(i)}(\lambda)\|_{\Omega-}$ and $\gamma_i = \|R_w^{(i)}(\lambda)\|_\infty$. If `R`$\{i\}$ is empty, then `GAP`$\{i\}$ is set to `NaN`.

In the case of calling `fdif2ngap` as

```
GAP = fdif2ngap(R,[],S)
```

then:

- If `R` is a LTI system with q residual outputs and `S` is a $q \times m_f$ structure matrix, then `GAP` is a q-dimensional vector, whose i-th component `GAP`$\{i\}$ is the fault-to-noise gap between the i-th rows $R_f^{(i)}(\lambda)$ and $R_w^{(i)}(\lambda)$ of $R_f(\lambda)$ and $R_w(\lambda)$, respectively, computed as β_i/γ_i, with $\beta_i = \|R_{f(i)}^{(i)}(\lambda)\|_{\infty-}$ and $\gamma_i = \|\,[\,R_{\bar f(i)}^{(i)}(\lambda)\; R_w^{(i)}(\lambda)\,]\,\|_\infty$, where $R_{f(i)}^{(i)}(\lambda)$ and $R_{\bar f(i)}^{(i)}(\lambda)$ are formed from the elements of $R_f^{(i)}(\lambda)$ which correspond to the `true` and, respectively, `false` values, in the i-th row of `S`;

- If R is a collection of N LTI systems and S is an $N \times m_f$ structure matrix, then GAP is an N-dimensional vector, whose i-th component GAP$\{i\}$ is the i-th fault-to-noise gap, computed as β_i/γ_i, with $\beta_i = \|R^{(i)}_{f^{(i)}}(\lambda)\|_{\infty -}$ and $\gamma_i = \big\| \big[\, R^{(i)}_{\bar f^{(i)}}(\lambda) \;\; R^{(i)}_w(\lambda) \,\big] \big\|_\infty$, where $R^{(i)}_{f^{(i)}}(\lambda)$ and $R^{(i)}_{\bar f^{(i)}}(\lambda)$ are formed from the columns of $R^{(i)}_f(\lambda)$ which correspond to the `true` and, respectively, `false` values, in the i-th row of S. If R$\{i\}$ is empty, then GAP$\{i\}$ is set to NaN.

In the case of calling `fdif2ngap` as

```
GAP = fdif2ngap(R,FREQ,S)
```

where both FREQ and S are nonempty then:

- If R is a LTI system with q residual outputs and S is a $q \times m_f$ structure matrix, then GAP is a q-dimensional vector, whose i-th component GAP$\{i\}$ is the fault-to-noise gap between the i-th rows $R^{(i)}_f(\lambda)$ and $R^{(i)}_w(\lambda)$ of $R_f(\lambda)$ and $R_w(\lambda)$, respectively, computed as β_i/γ_i, with $\beta_i = \|R^{(i)}_{f^{(i)}}(\lambda)\|_{\Omega -}$ and $\gamma_i = \big\| \big[\, R^{(i)}_{\bar f^{(i)}}(\lambda) \; R^{(i)}_w(\lambda) \,\big] \big\|_\infty$, where $R^{(i)}_{f^{(i)}}(\lambda)$ and $R^{(i)}_{\bar f^{(i)}}(\lambda)$ are formed from the elements of $R^{(i)}_f(\lambda)$ which correspond to the `true` and, respectively, `false` values, in the i-th row of S;
- If R is a collection of N LTI systems and S is an $N \times m_f$ structure matrix, then GAP is an N-dimensional vector, whose i-th component GAP$\{i\}$ is the i-th fault-to-noise gap, computed as β_i/γ_i, with $\beta_i = \|R^{(i)}_{f^{(i)}}(\lambda)\|_{\Omega -}$ and $\gamma_i = \big\| \big[\, R^{(i)}_{\bar f^{(i)}}(\lambda) \; R^{(i)}_w(\lambda) \,\big] \big\|_\infty$, where $R^{(i)}_{f^{(i)}}(\lambda)$ and $R^{(i)}_{\bar f^{(i)}}(\lambda)$ are formed from the columns of $R^{(i)}_f(\lambda)$ which correspond to the `true` and, respectively, `false` values, in the i-th row of S. If R$\{i\}$ is empty, then GAP$\{i\}$ is set to NaN.

In the case of calling `fdif2ngap` as

```
[BETA,GAMMA] = fdif2ngap(...)
```

then:

If BETA and GAMMA are scalar values, then they contain the values of β and γ, respectively, whose ratio represents the fault-to-noise gap. If BETA and GAMMA are vectors, then BETA$\{i\}$ and GAMMA$\{i\}$ contain the values of β_i and γ_i, respectively, whose ratio represents the i-th fault-to-noise gap. If R$\{i\}$ is empty, then BETA$\{i\}$ and GAMMA$\{i\}$ are set to zero.

Method

The definitions of the fault-to-noise gap in terms of the $\mathcal{H}_{\infty -}$-index and its finite frequency counterpart, the $\mathcal{H}_{\Omega -}$-index, are given in Section 2.7.2.

4.6.5 `fdimmperf`

Syntax

```
GAMMA = fdimmperf(R)
GAMMA = fdimmperf(R,SYSR)
GAMMA = fdimmperf(R,SYSR,NRMFLAG)
GAMMA = fdimmperf(R,[],NRMFLAG)
GAMMA = fdimmperf(R,[],NRMFLAG,S)
```

Description

`fdimmperf` evaluates the model-matching performance of the internal form of a FDI filter or the model-matching performance of the internal forms of a collection of FDI filters.

Input data

`R` is a stable LTI system or a cell array of stable LTI systems.

If `R` is a LTI system representing the internal form of a FDI filter, then it is in a descriptor system state-space form

$$\begin{aligned} E_R\lambda x_R(t) &= A_R x_R(t) + B_{R_u}u(t) + B_{R_d}d(t) + B_{R_f}f(t) + B_{R_w}w(t), \\ r(t) &= C_R x_R(t) + D_{R_u}u(t) + D_{R_d}d(t) + D_{R_f}f(t) + D_{R_w}w(t), \end{aligned} \tag{115}$$

where $r(t)$ is a q-dimensional residual output vector and any of the inputs components $u(t)$, $d(t)$, $f(t)$ or $w(t)$ can be void. For the system `R`, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ must have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively. The input-output form of `R` corresponding to (115) is

$$\mathbf{r}(\lambda) = R_u(\lambda)\mathbf{u}(\lambda) + R_d(\lambda)\mathbf{d}(\lambda) + R_f(\lambda)\mathbf{f}(\lambda) + R_w(\lambda)\mathbf{w}(\lambda), \tag{116}$$

where $R_u(\lambda)$, $R_d(\lambda)$, $R_f(\lambda)$, and $R_w(\lambda)$ are the transfer function matrices from the control, disturbance, fault, and noise inputs, respectively. In the case of void inputs, the corresponding transfer function matrices are assumed to be zero.

If `R` is an $N \times 1$ cell array of LTI systems representing the internal forms of N FDI filters, then the i-th component system `R`$\{i\}$ is in the state-space form

$$\begin{aligned} E_R^{(i)}\lambda x_R^{(i)}(t) &= A_R^{(i)} x_R^{(i)}(t) + B_{R_u}^{(i)}u(t) + B_{R_d}^{(i)}d(t) + B_{R_f}^{(i)}f(t) + B_{R_w}^{(i)}w(t)\,, \\ r^{(i)}(t) &= C_R^{(i)} x_R^{(i)}(t) + D_{R_u}^{(i)}u(t) + D_{R_d}^{(i)}d(t) + D_{R_f}^{(i)}f(t) + D_{R_w}^{(i)}w(t)\,, \end{aligned} \tag{117}$$

where $r^{(i)}(t)$ is a q_i-dimensional output vector representing the i-th residual component and any of the inputs components $u(t)$, $d(t)$, $f(t)$ or $w(t)$ can be void. For each component system `R`$\{i\}$, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ must have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively. The input-output form of `R`$\{i\}$ corresponding to (117) is

$$\mathbf{r}^{(i)}(\lambda) = R_u^{(i)}(\lambda)\mathbf{u}(\lambda) + R_d^{(i)}(\lambda)\mathbf{d}(\lambda) + R_f^{(i)}(\lambda)\mathbf{f}(\lambda) + R_w^{(i)}(\lambda)\mathbf{w}(\lambda), \tag{118}$$

where $R_u^{(i)}(\lambda)$, $R_d^{(i)}(\lambda)$, $R_f^{(i)}(\lambda)$, and $R_w^{(i)}(\lambda)$ are the transfer function matrices from the control, disturbance, fault, and noise inputs, respectively, for the i-th component system.

`SYSR`, if nonempty, is a stable LTI system or a cell array of stable LTI systems.

If `SYSR` is a LTI system representing a reference model for the internal form of the FDI filter `R`, then it is in the state-space form

$$\begin{aligned} \lambda x_r(t) &= A_r x_r(t) + B_{ru}u(t) + B_{rd}d(t) + B_{rf}f(t) + B_{rw}w(t), \\ y_r(t) &= C_r x_r(t) + D_{ru}u(t) + D_{rd}d(t) + D_{rf}f(t) + D_{rw}w(t), \end{aligned} \tag{119}$$

where the $y_r(t)$ is a q-dimensional vector and any of the inputs components $u(t)$, $d(t)$, $f(t)$, or $w(t)$ can be void. For the system `SYSR`, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ must have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively. The input-output form corresponding to (119) is

$$\mathbf{y}_r(\lambda) = M_{ru}(\lambda)\mathbf{u}(\lambda) + M_{rd}(\lambda)\mathbf{d}(\lambda) + M_{rf}(\lambda)\mathbf{f}(\lambda) + M_{rw}(\lambda)\mathbf{w}(\lambda), \tag{120}$$

where $M_{ru}(\lambda)$, $M_{rd}(\lambda)$, $M_{rf}(\lambda)$, and $M_{rw}(\lambda)$ are the transfer function matrices from the control, disturbance, fault, and noise inputs, respectively. In the case of void inputs, the corresponding transfer function matrices are assumed to be zero.

If `SYSR` is an $N \times 1$ cell array of LTI systems representing a collection of N reference models for the internal forms of N FDI filters in the $N \times 1$ cell array `R`, then the i-th component system `SYSR`$\{i\}$ is in the state-space form

$$\begin{aligned} \lambda x_r^{(i)}(t) &= A_r^{(i)} x_r^{(i)}(t) + B_{ru}^{(i)}u(t) + B_{rd}^{(i)}d(t) + B_{rf}^{(i)}f(t) + B_{rw}^{(i)}w(t), \\ y_r^{(i)}(t) &= C_r^{(i)} x_r^{(i)}(t) + D_{ru}^{(i)}u(t) + D_{rd}^{(i)}d(t) + D_{rf}^{(i)}f(t) + D_{rw}^{(i)}w(t)\,, \end{aligned} \tag{121}$$

where $y_r^{(i)}(t)$ is a q_i-dimensional vector and any of the inputs components $u(t)$, $d(t)$, $f(t)$ or $w(t)$ can be void. For each component system `SYSR`$\{i\}$, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ must have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively. The input-output form of `SYSR`$\{i\}$ corresponding to (121) is

$$\mathbf{y}_r^{(i)}(\lambda) = M_{ru}^{(i)}(\lambda)\mathbf{u}(\lambda) + M_{rd}^{(i)}(\lambda)\mathbf{d}(\lambda) + M_{rf}^{(i)}(\lambda)\mathbf{f}(\lambda) + M_{rw}^{(i)}(\lambda)\mathbf{w}(\lambda), \tag{122}$$

where $M_{ru}^{(i)}(\lambda)$, $M_{rd}^{(i)}(\lambda)$, $M_{rf}^{(i)}(\lambda)$, and $M_{rw}^{(i)}(\lambda)$ are the transfer function matrices from the control, disturbance, fault, and noise inputs, respectively, for the i-th component system.

`NRMFLAG` specifies the used system norm and, if specified, must be either 2 for using the $\mathcal{H}_2$-norm or `Inf` for using the $\mathcal{H}_\infty$-norm. By default, `NRMFLAG` = `Inf` (if not specified).

`S` is a $q \times m_f$ logical structure matrix if `R` is a LTI system with q outputs or is an $N \times m_f$ logical structure matrix if `R` is a collection of N LTI systems.

Output data

`GAMMA` is the computed model-matching performance, depending on the input variables.

In the case of calling `fdimmperf` as

```
GAMMA = fdimmperf(R,SYSR,NRMFLAG)
```

with `NRMFLAG` $= \alpha$ ($\alpha = 2$ or $\alpha = \infty$), then:

- If `R` and `SYSR` are LTI systems having input-output descriptions of the form (116) and (120), respectively, then the model-matching performance `GAMMA` $= \gamma$ is computed as

$$\gamma = \big\|\big[\; R_u(\lambda)-M_{ru}(\lambda) \;\; R_d(\lambda)-M_{rd}(\lambda) \;\; R_f(\lambda)-M_{rf}(\lambda) \;\; R_w(\lambda)-M_{rw}(\lambda) \;\big]\big\|_\alpha .$$

- If `R` and `SYSR` are cell arrays of N LTI systems having input-output descriptions of the form (118) and (122), respectively, then `GAMMA` is an N-dimensional vector, whose i-th component `GAMMA`$(i) = \gamma_i$ is computed as

$$\gamma_i = \big\|\big[\; R_u^{(i)}(\lambda)-M_{ru}^{(i)}(\lambda) \;\; R_d^{(i)}(\lambda)-M_{rd}^{(i)}(\lambda) \;\; R_f^{(i)}(\lambda)-M_{rf}^{(i)}(\lambda) \;\; R_w^{(i)}(\lambda)-M_{rw}^{(i)}(\lambda) \;\big]\big\|_\alpha .$$

The call of `fdimmperf` as

```
GAMMA = fdimmperf(R,SYSR)
```

is equivalent to

```
GAMMA = fdimmperf(R,SYSR,Inf) .
```

In the case of calling `fdimmperf` as

```
GAMMA = fdimmperf(R,[],NRMFLAG)
```

then, for `NRMFLAG` $= \alpha$:

- If `R` is a LTI system having the input-output description of the form (116), then the model-matching performance `GAMMA` $= \gamma$ is computed as $\gamma = \big\|R_w(\lambda)\big\|_\alpha$;
- If `R` is a collection of N LTI systems having the input-output descriptions (118), then `GAMMA` is an N-dimensional vector, whose i-th component `GAMMA`$(i) = \gamma_i$ is computed as $\gamma_i = \big\|R_w^{(i)}(\lambda)\big\|_\alpha$.

The call of `fdimmperf` as

```
GAMMA = fdimmperf(R)
```

is equivalent to

```
GAMMA = fdimmperf(R,[],Inf) .
```

In the case of calling `fdimmperf` as

```
GAMMA = fdimmperf(R,[],NRMFLAG,S)
```

with `NRMFLAG` $= \alpha$, then:

- If `R` is a LTI system having the input-output description of the form (116) and `S` is a $q \times m_f$ structure matrix, then the model-matching performance `GAMMA` $= \gamma$ is computed as $\gamma = \big\| \big[\, \overline{R}_f(\lambda) \;\; R_w(\lambda) \,\big] \big\|_\alpha$, where $\overline{R}_f(\lambda)$ is a $q \times m_f$ transfer function matrix whose (i,j)-th element is 0 if $S_{ij} =$ `true` and is equal to the (i,j)-th element of $R_f(\lambda)$ if $S_{ij} =$ `false`.
- If `R` is a collection of N LTI systems having the input-output descriptions (118), then `GAMMA` is an N-dimensional vector, whose i-th component `GAMMA`$(i) = \gamma_i$ is computed as
$$\gamma_i = \big\| \big[\, \overline{R}_f^{(i)}(\lambda) \; R_w^{(i)}(\lambda) \,\big] \big\|_\alpha,$$
where $\overline{R}_f^{(i)}(\lambda)$ is a transfer function matrix whose j-th column is 0 if $S_{ij} =$ `true` and is equal to the j-th column of $R_f^{(i)}(\lambda)$ if $S_{ij} =$ `false`.

Method

The definitions of the model-matching performance are given in Section 2.7.3.

4.7 Functions for Performance Evaluation of Model Detection Filters

The functions for performance evaluation address the computation of the performance criteria of model detection filters defined in Section 3.6. All functions are fully compatible with the results computed by the synthesis functions of model detection filters described in Section 4.9.

4.7.1 `mdperf`

Syntax

```
[MDGAIN,FPEAK,P,RELGAIN] = mdperf(R,OPTIONS)
```

Description

`mdperf` evaluates the distance mapping performance of a collection of model detection filters.

Input data

`R` is an $N \times N$ cell array of filters, where the (i,j)-th filter `R`$\{i,j\}$, is the internal form of the i-th model detection filter acting on the j-th model. Each nonempty `R`$\{i,j\}$ has a standard state-space representation

$$\begin{aligned} \lambda x_R^{(i,j)}(t) &= A_R^{(i,j)} x_R^{(i,j)}(t) + B_{R_u}^{(i,j)} u(t) + B_{R_d}^{(i,j)} d^{(j)}(t) + B_{R_w}^{(i,j)} w^{(j)}(t), \\ r^{(i,j)}(t) &= C_R^{(i,j)} x_R^{(i,j)}(t) + D_{R_u}^{(i,j)} u(t) + D_{R_d}^{(i,j)} d^{(j)}(t) + D_{R_w}^{(i,j)} w^{(j)}(t), \end{aligned} \tag{123}$$

where $x_R^{(i,j)}(t)$ is the state vector of the (i,j)-th filter with the residual output $r^{(i,j)}(t)$, control input $u(t) \in \mathbb{R}^{m_u}$, disturbance input $d^{(j)}(t) \in \mathbb{R}^{m_d^{(j)}}$ and noise input $w^{(j)}(t) \in \mathbb{R}^{m_w^{(j)}}$, and where any of the inputs components $u(t)$, $d^{(j)}(t)$, or $w^{(j)}(t)$ can be void. The input groups for $u(t)$, $d^{(j)}(t)$, and $w^{(j)}(t)$ have the standard names `'controls'`, `'disturbances'`, and `'noise'`, respectively. If `OPTIONS.cdinp = true` (see below), then the same disturbance input d is assumed for all filters (i.e., $d^{(j)} = d$). The state-space form (123) corresponds to the input-output form

$$\mathbf{r}^{(i,j)}(\lambda) = R_u^{(i,j)}(\lambda)\mathbf{u}(\lambda) + R_d^{(i,j)}(\lambda)\mathbf{d}^{(j)}(\lambda) + R_w^{(i,j)}(\lambda)\mathbf{w}^{(j)}(\lambda)\,, \tag{124}$$

where $R_u^{(i,j)}(\lambda)$, $R_d^{(i,j)}(\lambda)$ and $R_w^{(i,j)}(\lambda)$ are the TFMs from the corresponding inputs to the residual output.

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

`OPTIONS` **fields**	**Description**
`MDSelect`	M-dimensional integer vector with increasing elements σ_i, $i = 1, \ldots, M$, containing the indices of the selected model detection filters for which the gains of the corresponding internal forms have to be evaluated (Default: $[\,1, \ldots, N\,]$)
`MDFreq`	real vector, which contains the frequency values ω_k, $k = 1, \ldots, n_f$, for which the point-wise gains have to be computed. For each real frequency ω_k, there corresponds a complex frequency λ_k which is used to evaluate the point-wise gain. Depending on the system type, $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the common sampling time of the component systems. (Default: `[ ]`)
`cdinp`	option to use both control and disturbance input channels to evaluate the distance mapping performance, as follows: `true` – use both control and disturbance input channels; `false` – use only the control input channels (default)
`MDIndex`	index ℓ of the ℓ-th smallest gains to be used to evaluate the relative gains to the second smallest gains (Default: $\ell = 3$)

Output data

`MDGAIN` is an $M \times N$ nonnegative matrix, whose (i,j)-th element `MDGAIN`(i,j) contains the computed peak gain for the selected input channels of the (σ_i, j)-th filter as follows:
– if `OPTIONS.MDFreq` is empty and `OPTIONS.cdinp = false` then

$$\texttt{MDGAIN}(i,j) = \big\| R_u^{(\sigma_i,j)}(\lambda) \big\|_\infty ;$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = false` then

$$\texttt{MDGAIN}(i,j) = \max_k \big\| R_u^{(\sigma_i,j)}(\lambda_k) \big\|_2 ;$$

– if `OPTIONS.MDFreq` is empty and `OPTIONS.cdinp = true` then

$$\texttt{MDGAIN}(i,j) = \big\| \big[\, R_u^{(\sigma_i,j)}(\lambda) \; R_d^{(\sigma_i,j)}(\lambda) \,\big] \big\|_\infty ;$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = true` then

$$\texttt{MDGAIN}(i,j) = \max_k \big\| \big[\, R_u^{(\sigma_i,j)}(\lambda_k) \; R_d^{(\sigma_i,j)}(\lambda_k) \,\big] \big\|_2 .$$

`FPEAK` is an $M \times N$ nonnegative matrix, whose (i,j)-th element `FPEAK`(i,j) contains the peak frequency (in rad/TimeUnit), where `MDGAIN`(i,j) is achieved.

`P` is an $M \times N$ integer matrix, whose i-th row contains the permutation to be applied to increasingly reorder the i-th row of `MDGAIN`.

`RELGAIN` is an M-dimensional vector, whose i-th element `RELGAIN`(i) contains the ratio of the second and ℓ-th smallest gains in the i-th row of `MDGAIN`, where $\ell =$ `OPTIONS.MDIndex`.

Method

The definition of the distance mapping performance of a set of model detection filters is given in Section 3.6.3.

4.7.2 `mdmatch`

Syntax

```
[MDGAIN,FPEAK,MIND] = mdmatch(Q,SYS,OPTIONS)
```

Description

`mdmatch` evaluates the distance matching performance of a collection of model detection filters acting on a given model and determines the index of the best matching component model.

Input data

`Q` is an $N \times 1$ cell array of stable model detection filters, where `Q`$\{i\}$ contains the i-th filter in a standard state-space representation

$$\begin{aligned} \lambda x_Q^{(i)}(t) &= A_Q^{(i)} x_Q^{(i)}(t) + B_{Q_y}^{(i)} y(t) + B_{Q_u}^{(i)} u(t), \\ r^{(i)}(t) &= C_Q^{(i)} x_Q^{(i)}(t) + D_{Q_y}^{(i)} y(t) + D_{Q_u}^{(i)} u(t), \end{aligned} \tag{125}$$

where $x_Q^{(i)}(t)$ is the state vector of the i-th filter with the residual signal $r^{(i)}(t)$ as output and the measured outputs $y(t)$ and control inputs $u(t)$ as inputs. The input groups for $y(t)$ and $u(t)$ have the standard names `'outputs'` and `'controls'`, respectively. The state-space form (125) corresponds to the input-output form

$$\mathbf{r}^{(i)}(\lambda) = Q^{(i)}(\lambda) \left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array} \right] . \tag{126}$$

`Q`$\{i\}$ may be empty.

`SYS` is a stable LTI model in the state-space form

$$\begin{array}{rcl} E\lambda x(t) &=& Ax(t) + B_u u(t) + B_d d(t) + B_w w(t)\,, \\ y(t) &=& Cx(t) + D_u u(t) + D_d d(t) + D_w w(t)\,, \end{array} \tag{127}$$

where $x(t) \in \mathbb{R}^n$ is the state vector of the system with control input $u(t) \in \mathbb{R}^{m_u}$, disturbance input $d(t) \in \mathbb{R}^{m_d}$ and noise input $w(t) \in \mathbb{R}^{m_w}$, and where any of the inputs components $u(t)$, $d(t)$, or $w(t)$ can be void. The input groups for $u(t)$, $d(t)$, and $w(t)$ have the standard names `'controls'`, `'disturbances'`, and `'noise'`, respectively. The state-space form (127) corresponds to the input-output form

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_w(\lambda)\mathbf{w}(\lambda), \tag{128}$$

where $G_u(\lambda)$, $G_d(\lambda)$ and $G_w(\lambda)$ are the TFMs from the corresponding inputs to the output.

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

`OPTIONS` **fields**	**Description**
`MDFreq`	real vector, which contains the frequency values ω_k, $k = 1, \ldots, n_f$, for which the point-wise gains have to be computed. For each real frequency ω_k, there corresponds a complex frequency λ_k which is used to evaluate the point-wise gain. Depending on the system type, $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the common sampling time of the component systems. (Default: `[ ]`)
`cdinp`	option to use both control and disturbance input channels to evaluate the distance matching performance, as follows: `true` – use both control and disturbance input channels; `false` – use only the control input channels (default)

Output data

`MDGAIN` is an N-dimensional column vector, whose i-th element `MDGAIN`(i) contains, for a nonempty filter `Q`$\{i\}$, the computed peak gain for the selected input channels of the i-th internal form as follows:

– if `OPTIONS.MDFreq` is empty and `OPTIONS.cdinp = false` then

$$\texttt{MDGAIN}(i) = \left\| Q^{(i)}(\lambda) \left[\begin{array}{c} G_u(\lambda) \\ I_{m_u} \end{array} \right] \right\|_\infty ;$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = false` then

$$\texttt{MDGAIN}(i) = \max_k \left\| Q^{(i)}(\lambda_k) \left[\begin{array}{c} G_u(\lambda_k) \\ I_{m_u} \end{array} \right] \right\|_\infty ;$$

– if `OPTIONS.MDFreq` is empty and `OPTIONS.cdinp = true` then

$$\texttt{MDGAIN}(i) = \left\| Q^{(i)}(\lambda) \left[\begin{array}{cc} G_u(\lambda) & G_d(\lambda) \\ I_{m_u} & 0 \end{array} \right] \right\|_\infty ;$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = true` then

$$\texttt{MDGAIN}(i) = \max_k \left\| Q^{(i)}(\lambda_k) \left[\begin{array}{cc} G_u(\lambda_k) & G_d(\lambda_k) \\ I_{m_u} & 0 \end{array} \right] \right\|_\infty .$$

`MDGAIN`$(i) = 0$ if `Q`$\{i\}$ is empty.

`FPEAK` is an N-dimensional vector, whose i-th element `FPEAK`(i) contains the peak frequency (in rad/TimeUnit), where `MDGAIN`(i) is achieved.

`MIND` is the index ℓ of the component of `MDGAIN` for which the minimum value of the peak gains is achieved. For a properly designed filter `Q`, this is also the index of the best matching model to the current model `SYS`.

Method

The definitions related to the model matching performance of a set of model detection filters are given in Section 3.6.4.

4.7.3 `mdgap`

Syntax

```
GAP = mdgap(R,OPTIONS)
[BETA,GAMMA] = mdgap(R,OPTIONS)
```

Description

`mdgap` computes the noise gaps of model detection filters.

Input data

`R` is an $N \times N$ cell array of filters, where the (i,j)-th filter `R`$\{i,j\}$, is the internal form of the i-th model detection filter acting on the j-th model. Each nonempty `R`$\{i,j\}$ has a standard state-space representation

$$\begin{aligned} \lambda x_R^{(i,j)}(t) &= A_R^{(i,j)} x_R^{(i,j)}(t) + B_{R_u}^{(i,j)} u(t) + B_{R_d}^{(i,j)} d^{(j)}(t) + B_{R_w}^{(i,j)} w^{(j)}(t), \\ r^{(i,j)}(t) &= C_R^{(i,j)} x_R^{(i,j)}(t) + D_{R_u}^{(i,j)} u(t) + D_{R_d}^{(i,j)} d^{(j)}(t) + D_{R_w}^{(i,j)} w^{(j)}(t) \end{aligned} \tag{129}$$

where $x_R^{(i,j)}(t)$ is the state vector of the (i,j)-th filter with the residual output $r^{(i,j)}(t)$, control input $u(t) \in \mathbb{R}^{m_u}$, disturbance input $d^{(j)}(t) \in \mathbb{R}^{m_d^{(j)}}$ and noise input $w^{(j)}(t) \in \mathbb{R}^{m_w^{(j)}}$, and where any of the inputs components $u(t)$, $d^{(j)}(t)$, or $w^{(j)}(t)$ can be void. The input groups for $u(t)$, $d^{(j)}(t)$, and $w^{(j)}(t)$ have the standard names `'controls'`, `'disturbances'`, and `'noise'`, respectively. If `OPTIONS.cdinp = true` (see below), then the same disturbance input d is assumed for all filters (i.e., $d^{(j)} = d$). The state-space form (129) corresponds to the input-output form

$$\mathbf{r}^{(i,j)}(\lambda) = R_u^{(i,j)}(\lambda)\mathbf{u}(\lambda) + R_d^{(i,j)}(\lambda)\mathbf{d}^{(j)}(\lambda) + R_w^{(i,j)}(\lambda)\mathbf{w}^{(j)}(\lambda)\,, \tag{130}$$

where $R_u^{(i,j)}(\lambda)$, $R_d^{(i,j)}(\lambda)$ and $R_w^{(i,j)}(\lambda)$ are the TFMs from the corresponding inputs to the residual output.

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

`OPTIONS` **fields**	**Description**
`MDSelect`	M-dimensional integer vector with increasing elements σ_i, $i = 1, \ldots, M$, containing the indices of the selected model detection filters for which the gaps of the corresponding internal forms have to be evaluated (Default: $[1, \ldots, N]$)
`MDFreq`	real vector, which contains the frequency values ω_k, $k = 1, \ldots, n_f$, for which the point-wise gains have to be computed. For each real frequency ω_k, there corresponds a complex frequency λ_k which is used to evaluate the point-wise gain. Depending on the system type, $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the common sampling time of the component systems. (Default: `[ ]`)
`cdinp`	option to use both control and disturbance input channels to evaluate the noise gaps, as follows: `true` – use both control and disturbance input channels; `false` – use only the control input channels (default)

Output data

In the case of calling `mdgap` as

```
GAP = mdgap(R,OPTIONS)
```

then `GAP` is an M-dimensional vector, whose i-th element `GAP`(i) contains the computed noise gap for the selected input channels of the (σ_i, j)-th filter as follows:
– if `OPTIONS.MDFreq` is empty and `OPTIONS.cdinp = false` then

$$\texttt{GAP}(i) = \min_{j\neq\sigma_i} \big\|R_u^{(\sigma_i,j)}(\lambda)\big\|_\infty / \big\|R_w^{(\sigma_i,\sigma_i)}(\lambda)\big\|_\infty;$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = false` then

$$\texttt{GAP}(i) = \min_{j\neq\sigma_i} \max_k \big\|R_u^{(\sigma_i,j)}(\lambda_k)\big\|_\infty / \big\|R_w^{(\sigma_i,\sigma_i)}(\lambda)\big\|_\infty;$$

– if `OPTIONS.MDFreq` is empty and `OPTIONS.cdinp = true` then

$$\texttt{GAP}(i) = \min_{j\neq\sigma_i} \big\|\big[\, R_u^{(\sigma_i,j)}(\lambda)\ R_d^{(\sigma_i,j)}(\lambda)\,\big]\big\|_\infty / \big\|R_w^{(\sigma_i,\sigma_i)}(\lambda)\big\|_\infty;$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = true` then

$$\texttt{GAP}(i) = \min_{j\neq\sigma_i} \max_k \big\|\big[\, R_u^{(\sigma_i,j)}(\lambda_k)\ R_d^{(\sigma_i,j)}(\lambda_k)\,\big]\big\|_\infty / \big\|R_w^{(\sigma_i,\sigma_i)}(\lambda)\big\|_\infty.$$

In the case of calling `mdgap` as

```
[BETA,GAMMA] = mdgap(R,OPTIONS)
```

then `BETA` and `GAMMA` are M-dimensional vector, whose i-th elements `BETA`(i) and `GAMMA`(i) contain the values whose ratio represents the noise gaps for the selected input channels of the (σ_i, j)-th filter as follows:
– if `OPTIONS.MDFreq` is empty and `OPTIONS.cdinp = false` then

$$\texttt{BETA}(i) = \min_{j\neq\sigma_i} \big\|R_u^{(\sigma_i,j)}(\lambda)\big\|_\infty, \quad \texttt{GAMMA}(i) = \big\|R_w^{(\sigma_i,\sigma_i)}(\lambda)\big\|_\infty;$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = false` then

$$\texttt{BETA}(i) = \min_{j\neq\sigma_i} \max_k \big\|R_u^{(\sigma_i,j)}(\lambda_k)\big\|_2, \quad \texttt{GAMMA}(i) = \big\|R_w^{(\sigma_i,\sigma_i)}(\lambda)\big\|_\infty;$$

– if `OPTIONS.MDFreq` is empty and `OPTIONS.cdinp = true` then

$$\texttt{BETA}(i) = \min_{j\neq\sigma_i} \big\|\big[\, R_u^{(\sigma_i,j)}(\lambda)\ R_d^{(\sigma_i,j)}(\lambda)\,\big]\big\|_\infty, \quad \texttt{GAMMA}(i) = \big\|R_w^{(\sigma_i,\sigma_i)}(\lambda)\big\|_\infty;$$

– if `OPTIONS.MDFreq` is nonempty and `OPTIONS.cdinp = true` then

$$\texttt{BETA}(i) = \min_{j\neq\sigma_i} \max_k \big\|\big[\, R_u^{(\sigma_i,j)}(\lambda_k)\ R_d^{(\sigma_i,j)}(\lambda_k)\,\big]\big\|_2, \quad \texttt{GAMMA}(i) = \big\|R_w^{(\sigma_i,\sigma_i)}(\lambda)\big\|_\infty.$$

Method

The definition of the noise gap of a set of model detection filters is given in Section 3.6.5.

4.8 Functions for the Synthesis of FDI Filters

4.8.1 `efdsyn`

Syntax

`[Q,R,INFO] = efdsyn(SYSF,OPTIONS)`

Description

`efdsyn` solves the *exact fault detection problem* (EFDP) (see Section 2.6.1), for a given LTI system `SYSF` with additive faults. Two stable and proper filters, `Q` and `R`, are computed, where `Q` contains the fault detection filter representing the solution of the EFDP, and `R` contains its internal form.

Input data

`SYSF` is a LTI system in the state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_u u(t) + B_d d(t) + B_f f(t) + B_w w(t) + B_v v(t),\\ y(t) &= Cx(t) + D_u u(t) + D_d d(t) + D_f f(t) + D_w w(t) + B_v v(t), \end{aligned} \tag{131}$$

where any of the inputs components $u(t)$, $d(t)$, $f(t)$, $w(t)$ or $v(t)$ can be void. The auxiliary input signal $v(t)$ can be used for convenience. For the system `SYSF`, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively. For the auxiliary input $v(t)$ the standard input group `'aux'` can be used.

`OPTIONS` is a MATLAB structure used to specify various synthesis options and has the following fields:

`OPTIONS` **fields**	**Description**
`tol`	relative tolerance for rank computations (Default: internally computed)
`tolmin`	absolute tolerance for observability tests (Default: internally computed)
`FDTol`	threshold for fault detectability checks (Default: 10^{-4})
`FDGainTol`	threshold for strong fault detectability checks (Default: 10^{-2})
`rdim`	desired number q of residual outputs for `Q` and `R` (Default: `[ ]`, in which case: if `OPTIONS.HDesign` is empty, then $q = 1$, if `OPTIONS.minimal` = `true`, or $q = p - r_d$, if `OPTIONS.minimal` = `false` (see **Method**); if `OPTIONS.HDesign` is nonempty, then q is the row dimension of the design matrix H contained in `OPTIONS.HDesign`)
`FDFreq`	vector of real frequency values for strong detectability checks (Default: `[ ]`)

`smarg`	stability margin for the poles of the filters `Q` and `R` (Default: `-sqrt(eps)` for a continuous-time system `SYSF`; `1-sqrt(eps)` for a discrete-time system `SYSF`).
`sdeg`	prescribed stability degree for the poles of the filters `Q` and `R` (Default: -0.05 for a continuous-time system `SYSF`; 0.95 for a discrete-time system `SYSF`).
`poles`	complex vector containing a complex conjugate set of desired poles (within the stability domain) to be assigned for the filters `Q` and `R` (Default: `[ ]`)
`nullspace`	option to use a specific proper nullspace basis `true` – use a minimal proper basis (default); `false` – use a full-order observer based basis (see **Method**). *Note:* This option can only be used if no disturbance inputs are present in (131) and E is invertible.
`simple`	option to employ a simple proper basis for filter synthesis `true` – use a simple basis; `false` – use a non-simple basis (default)
`minimal`	option to perform a least order filter synthesis `true` – perform least order synthesis (default); `false` – perform full order synthesis.
`tcond`	maximum alowed condition number of the employed non-orthogonal transformations (Default: 10^4).
`HDesign`	full row rank design matrix H to build `OPTIONS.rdim` linear combinations of the left nullspace basis vectors (see **Method**) (Default: `[ ]`)

Output data

`Q` is the resulting fault detection filter in a standard state-space form

$$\begin{aligned} \lambda x_Q(t) &= A_Q x_Q(t) + B_{Q_y} y(t) + B_{Q_u} u(t), \\ r(t) &= C_Q x_Q(t) + D_{Q_y} y(t) + D_{Q_u} u(t), \end{aligned} \tag{132}$$

where the residual signal $r(t)$ is a q-dimensional vector. The resulting value of q depends on the selected options `OPTIONS.rdim` and `OPTIONS.minimal` (see **Method**). For the system object `Q`, two input groups `'outputs'` and `'controls'` are defined for $y(t)$ and $u(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r(t)$.

`R` is the resulting internal form of the fault detection filter and has a standard state-space representation of the form

$$\begin{aligned} \lambda x_R(t) &= A_Q x_R(t) + B_{R_f} f(t) + B_{R_w} w(t) + B_{R_v} v(t), \\ r(t) &= C_Q x_R(t) + D_{R_f} f(t) + D_{R_w} w(t) + D_{R_v} v(t). \end{aligned} \tag{133}$$

The input groups `'faults'`, `'noise'` and `'aux'` are defined for $f(t)$, $w(t)$, and $v(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r(t)$. Notice that the realizations of `Q` and `R` share the matrices A_Q and C_Q.

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

`INFO` **fields**	**Description**
`tcond`	maximum of the condition numbers of the employed non-orthogonal transformation matrices; a warning is issued if `INFO.tcond` $\geq$ `OPTIONS.tcond`.
`degs`	if `OPTIONS.simple` = `true`, the orders of the basis vectors of the employed simple nullspace basis; if `OPTIONS.simple` = `false`, the degrees of the basis vectors of an equivalent polynomial nullspace basis. `INFO.degs = [ ]` if `OPTIONS.nullspace = false` is used.
`S`	binary structure matrix corresponding to $H\overline{G}_f(\lambda)$ (see **Method**)
`HDesign`	design matrix H employed for the synthesis of the fault detection filter (see **Method**)

Method

The function `efdsyn` implements an extension of the **Procedure EFD** from [16, Sect. 5.2], which relies on the nullspace-based synthesis method proposed in [5]. In what follows, we succinctly present this extended procedure, in terms of the input-output descriptions. Full details of the employed state-space based computational algorithms are given in [16][Chapter 7].

Let assume the system `SYSF` in (131) has the equivalent input-output form

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_f(\lambda)\mathbf{f}(\lambda) + G_w(\lambda)\mathbf{w}(\lambda) + G_v(\lambda)\mathbf{v}(\lambda), \tag{134}$$

where the vectors y, u, d, f, w and v have dimensions p, m_u, m_d, m_f, m_w and m_v, respectively. The resulting fault detection filter in (132) has the input-output form

$$\mathbf{r}(\lambda) = Q(\lambda)\left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array}\right], \tag{135}$$

where the residual vector $r(t)$ has q components.

The synthesis method which underlies **Procedure EFD**, essentially determines the filter $Q(\lambda)$ as a stable rational left annihilator of

$$G(\lambda) := \left[\begin{array}{cc} G_u(\lambda) & G_d(\lambda) \\ I_{m_u} & 0 \end{array}\right], \tag{136}$$

such that $R_{f_j}(\lambda) \neq 0$, for $j = 1, \ldots, m_f$, where

$$R(\lambda) := \left[\begin{array}{c|c|c} R_f(\lambda) & R_w(\lambda) & R_v(\lambda) \end{array}\right] := Q(\lambda)\left[\begin{array}{c|c|c} G_f(\lambda) & G_w(\lambda) & G_v(\lambda) \\ 0 & 0 & 0 \end{array}\right]. \tag{137}$$

The resulting internal form of the fault detection filter (135) is

$$\mathbf{r}(\lambda) = R(\lambda)\begin{bmatrix} \mathbf{f}(\lambda) \\ \mathbf{w}(\lambda) \\ \mathbf{v}(\lambda) \end{bmatrix} = R_f(\lambda)\mathbf{f}(\lambda) + R_w(\lambda)\mathbf{w}(\lambda) + R_v(\lambda)\mathbf{v}(\lambda)\,, \tag{138}$$

with $R(\lambda)$, defined in (137), stable.

The filter $Q(\lambda)$ is determined in the product form

$$Q(\lambda) = Q_3(\lambda)Q_2(\lambda)Q_1(\lambda), \tag{139}$$

where the factors are determined as follows:

(a) $Q_1(\lambda) = N_l(\lambda)$, with $N_l(\lambda)$ a $\big(p - r_d\big) \times (p + m_u)$ proper rational left nullspace basis satisfying $N_l(\lambda)G(\lambda) = 0$, with $r_d := \operatorname{rank} G_d(\lambda)$;

(b) $Q_2(\lambda)$ is an admissible factor (i.e., guaranteeing complete fault detectability) to perform least order synthesis;

(c) $Q_3(\lambda)$ is a stable invertible factor determined such that $Q(\lambda)$ and the associated $R(\lambda)$ in (137) have a desired dynamics.

The computations of individual factors depend on the user's options and specific choices are discussed in what follows.

Computation of $Q_1(\lambda)$

If `OPTIONS.nullspace = true` or $m_d > 0$ or E is singular, then $N_l(\lambda)$ is determined as a minimal proper nullspace basis. In this case, if `OPTIONS.simple = true`, then $N_l(\lambda)$ is determined as a simple rational basis and the orders of the basis vectors are provided in `INFO.degs`. If `OPTIONS.simple = false`, then $N_l(\lambda)$ is determined as a proper rational basis and `INFO.degs` contains the degrees of the basis vectors of an equivalent polynomial nullspace basis (see [16, Section 9.1.3] for definitions). A stable basis is determined if `OPTIONS.FDfreq` is not empty.

If `OPTIONS.nullspace = false`, $m_d = 0$ and E is invertible, then $N_l(\lambda) = [\, I_p \;\; -G_u(\lambda)\,]$ is used, which corresponds to a full-order Luenberger observer. If `OPTIONS.FDfreq` is not empty and the system (131) is unstable, then $\widetilde{N}_l(\lambda) = M(\lambda)N_l(\lambda)$ is used instead $N_l(\lambda)$, where $M(\lambda)$ and $\widetilde{N}_l(\lambda)$ are the stable factors of a left coprime factorization $N_l(\lambda) = M^{-1}(\lambda)\widetilde{N}_l(\lambda)$.

To check the solvability of the EFDP, the transfer function matrix $\overline{G}_f(\lambda) := Q_1(\lambda)\left[\begin{smallmatrix} G_f(\lambda) \\ 0 \end{smallmatrix}\right]$ and the structure matrix $S_{H\overline{G}_f}$ of $H\overline{G}_f(\lambda)$ are determined, where H is the design matrix specified in a nonempty `OPTIONS.HDesign` (otherwise $H = I_{p-r_d}$ is used). The EFDP is solvable provided $S_{H\overline{G}_f}$ has all its columns nonzero. The resulted $S_{H\overline{G}_f}$ is provided in `INFO.S`.

Computation of $Q_2(\lambda)$

For a nonempty `OPTIONS.rdim`, the resulting dimension q of the residual vector $r(t)$ is $q = \min(\texttt{OPTIONS.rdim}, p - r_d)$, where $r_d = \operatorname{rank} G_d(\lambda)$. If `OPTIONS.rdim` is empty, then a default value of q is used (see the description of `OPTIONS.rdim`).

If `OPTIONS.minimal = false`, then $Q_2(\lambda) = H$, where H is a suitable $q \times \big(p - r_d\big)$ full row rank design matrix. H is set as follows. If `OPTIONS.HDesign` is nonempty, then $H =$ `OPTIONS.HDesign`. If `OPTIONS.HDesign` is empty, then the matrix H is chosen to build q linear combinations of the $p - r_d$ left nullspace basis vectors, such that $HQ_1(\lambda)$ has full row rank. If $q = p - r_d$ then the choice $H = I_{p-r_d}$ is used, otherwise H is chosen a randomly generated $q \times \big(p - r_d\big)$ real matrix.

If `OPTIONS.minimal = true`, then $Q_2(\lambda)$ is a $q \times \big(p - r_d\big)$ transfer function matrix, with q chosen as above. $Q_2(\lambda)$ is determined in the form

$$Q_2(\lambda) = H + Y_2(\lambda)\,,$$

such that $Q_2(\lambda)Q_1(\lambda)$ $\big(= HN_l(\lambda) + Y_2(\lambda)N_l(\lambda)\big)$ and $Y_2(\lambda)$ are the least order solution of a left minimal cover problem [15]. If `OPTIONS.HDesign` is nonempty, then $H =$ `OPTIONS.HDesign`, and if `OPTIONS.HDesign` is empty, then a suitable randomly generated H is employed (see above).

The structure field `INFO.HDesign` contains the employed value of the design matrix H.

Computation of $Q_3(\lambda)$

$Q_3(\lambda)$ is a stable invertible transfer function matrix determined such that $Q(\lambda)$ in (139) and the associated $R(\lambda)$ in (137) have a desired dynamics (specified via `OPTIONS.sdeg` and `OPTIONS.poles`).

Example

Example 7. This is Example 5.4 from the book [16] of an unstable continuous-time system with the TFMs

$$G_u(s) = \begin{bmatrix} \dfrac{s+1}{s-2} \\[2mm] \dfrac{s+2}{s-3} \end{bmatrix}, \quad G_d(s) = \begin{bmatrix} \dfrac{s-1}{s+2} \\[2mm] 0 \end{bmatrix}, \quad G_f(s) = \begin{bmatrix} \dfrac{s+1}{s-2} & 0 \\[2mm] \dfrac{s+2}{s-3} & 1 \end{bmatrix}, \quad G_w(s) = 0, \quad G_v(s) = 0,$$

where the fault input f_1 corresponds to an additive actuator fault, while the fault input f_2 describes an additive sensor fault in the second output y_2. The TFM $G_d(s)$ is non-minimum phase, having an unstable zero at 1.

We want to design a least order fault detection filter $Q(s)$ with scalar output, and a stability degree of -3 for the poles, which fulfills:

– the decoupling condition: $Q(s)\left[\begin{smallmatrix} G_u(s) & G_d(s) \\ I_{m_u} & 0 \end{smallmatrix}\right] = 0$;

– the fault detectability condition: $R_{f_j}(\lambda) \neq 0, \quad j = 1, \ldots, m_f$.

The results computed with the following script are

$$Q(s) = \begin{bmatrix} 0 & \dfrac{s-3}{s+3} & -\dfrac{s+2}{s+3} \end{bmatrix}, \quad R_f(s) = \begin{bmatrix} \dfrac{s+2}{s+3} & \dfrac{s-3}{s+3} \end{bmatrix}.$$

```
% Example - Solution of an exact fault detection problem (EFDP)

s = tf('s'); % define the Laplace variable s
% define Gu(s) and Gd(s)
Gu = [(s+1)/(s-2); (s+2)/(s-3)];     % enter Gu(s)
Gd = [(s-1)/(s+2); 0];               % enter Gd(s)
p = 2; mu = 1; md = 1; mf = 2;       % set dimensions

% setup the synthesis model with faults
sysf = fdimodset(ss([Gu Gd]),struct('c',1,'d',2,'f',1,'fs',2));

% call of EFDSYN with the options for stability degree -3 and the synthesis
% of a scalar output filter
[Q,R] = efdsyn(sysf,struct('sdeg',-3,'rdim',1));

% display the implementation form Q and the internal form Rf of the
% resulting fault detection filter
tf(Q), tf(R)

% check synthesis conditions: Q[Gu Gd;I 0] = 0 and Q[Gf; 0] = R
syse = [sysf;eye(mu,mu+md+mf)];  % form Ge = [Gu Gd Gf;I 0 0];
norm_Ru_Rd = norm(Q*syse(:,{'controls','disturbances'}),inf)
norm_rez = norm(Q*syse(:,'faults')-R,inf)

% check weak and strong fault detectability
S_weak = fditspec(R)
[S_strong,abs_dcgains] = fdisspec(R)

% determine the fault sensitivity condition
FSCOND = fdifscond(R,0)

% evaluate step responses
set(R,'InputName',{'f_1','f_2'},'OutputName','r');
step(R);
title('Step responses from the fault inputs'), ylabel('')
```

4.8.2 `afdsyn`

Syntax

```
[Q,R,INFO] = afdsyn(SYSF,OPTIONS)
```

Description

`afdsyn` solves the *approximate fault detection problem* (AFDP) (see Section 2.6.2), for a given LTI system `SYSF` with additive faults. Two stable and proper filters, `Q` and `R`, are computed, where `Q` contains the fault detection filter representing the solution of the AFDP, and `R` contains its internal form.

Input data

`SYSF` is a LTI system in the state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_u u(t) + B_d d(t) + B_f f(t) + B_w w(t) + B_v v(t), \\ y(t) &= Cx(t) + D_u u(t) + D_d d(t) + D_f f(t) + D_w w(t) + B_v v(t), \end{aligned} \tag{140}$$

where any of the inputs components $u(t)$, $d(t)$, $f(t)$, $w(t)$ or $v(t)$ can be void. The auxiliary input signal $v(t)$ can be used for convenience. For the system `SYSF`, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively. For the auxiliary input $v(t)$ the standard input group `'aux'` can be used.

`OPTIONS` is a MATLAB structure used to specify various synthesis options and has the following fields:

`OPTIONS` **fields**	**Description**
`tol`	relative tolerance for rank computations (Default: internally computed)
`tolmin`	absolute tolerance for observability tests (Default: internally computed)
`FDTol`	threshold for fault detectability checks (Default: 10^{-4})
`FDGainTol`	threshold for strong fault detectability checks (Default: 10^{-2})

`rdim`	desired number q of residual outputs for `Q` and `R` (Default: `[ ]`, in which case $q = q_1 + q_2$, with q_1 and q_2 selected as follows: if `OPTIONS.HDesign` is empty, then $q_1 = \min(1, r_w)$ if `OPTIONS.minimal` = `true`, or $q_1 = r_w$, if `OPTIONS.minimal` = `false` (see **Method**); if `OPTIONS.HDesign` is nonempty, then q_1 is the row dimension of the design matrix H_1 contained in `OPTIONS.HDesign` if `OPTIONS.HDesign2` is empty, then $q_2 = 1 - \min(1, r_w)$ if `OPTIONS.minimal` = `true`, or $q_2 = p - r_d - r_w$, if `OPTIONS.minimal` = `false` (see **Method**); if `OPTIONS.HDesign2` is nonempty, then q_2 is the row dimension of the design matrix H_2 contained in `OPTIONS.HDesign2`)
`FDFreq`	vector of real frequency values for strong detectability checks (Default: `[ ]`)
`smarg`	stability margin for the poles of the filters `Q` and `R` (Default: `-sqrt(eps)` for a continuous-time system `SYSF`; `1-sqrt(eps)` for a discrete-time system `SYSF`).
`sdeg`	prescribed stability degree for the poles of the filters `Q` and `R` (Default: -0.05 for a continuous-time system `SYSF`; 0.95 for a discrete-time system `SYSF`).
`poles`	complex vector containing a complex conjugate set of desired poles (within the stability domain) to be assigned for the filters `Q` and `R` (Default: `[ ]`)
`nullspace`	option to use a specific proper nullspace basis `true` – use a minimal proper basis (default); `false` – use a full-order observer based basis (see **Method**). *Note:* This option can be only used if no disturbance inputs are present in (140) and E is invertible.
`simple`	option to employ a simple proper basis for filter synthesis `true` – use a simple basis; `false` – use a non-simple basis (default)
`minimal`	option to perform a least order filter synthesis `true` – perform least order synthesis (default); `false` – perform full order synthesis.
`exact`	option to perform exact filter synthesis `true` – perform exact synthesis (i.e., no optimization performed); `false` – perform approximate synthesis (default).
`tcond`	maximum alowed condition number of the employed non-orthogonal transformations (Default: 10^4).

`freq`	complex frequency value to be employed to check the full row rank admissibility condition (see **Method**) (Default: `[ ]`, i.e., a randomly generated frequency).
`HDesign`	design matrix H_1, with full row rank q_1, to build q_1 linear combinations of the left nullspace basis vectors of $G_1(\lambda) := \left[\begin{smallmatrix} G_u(\lambda) & G_d(\lambda) \\ I & 0 \end{smallmatrix}\right]$; H_1 is used for the synthesis of the filter components $Q^{(1)}(\lambda)$ and $R^{(1)}(\lambda)$ (see **Method**) (Default: `[ ]`)
`HDesign2`	design matrix H_2, with full row rank q_2, to build q_2 linear combinations of the left nullspace basis vectors of $G_2(\lambda) := \left[\begin{smallmatrix} G_u(\lambda) & G_d(\lambda) & G_w(\lambda) \\ I & 0 & 0 \end{smallmatrix}\right]$; H_2 is used for the synthesis of the filter components $Q^{(2)}(\lambda)$ and $R^{(2)}(\lambda)$ (see **Method**) (Default: `[ ]`)
`gamma`	upper bound on the resulting $\|R_w(\lambda)\|_\infty$ (see **Method**) (Default: 1)
`epsreg`	regularization parameter (Default: 0.1)
`sdegzer`	prescribed stability degree for zeros shifting (Default: -0.05 for a continuous-time system `SYSF`; 0.95 for a discrete-time system `SYSF`).
`nonstd`	option to handle nonstandard optimization problems (see **Method**) 1 – use the quasi-co-outer–co-inner factorization (default); 2 – use the modified co-outer–co-inner factorization with the regularization parameter `OPTIONS.epsreg`; 3 – use the Wiener-Hopf type co-outer–co-inner factorization. 4 – use the Wiener-Hopf type co-outer-co-inner factorization with zero shifting of the non-minimum phase factor using the stabilization parameter `OPTIONS.sdegzer` 5 – use the Wiener-Hopf type co-outer-co-inner factorization with the regularization of the non-minimum phase factor using the regularization parameter `OPTIONS.epsreg`

Output data

`Q` is the resulting fault detection filter in a standard state-space form

$$\begin{aligned} \lambda x_Q(t) &= A_Q x_Q(t) + B_{Q_y} y(t) + B_{Q_u} u(t), \\ r(t) &= C_Q x_Q(t) + D_{Q_y} y(t) + D_{Q_u} u(t), \end{aligned} \tag{141}$$

where the residual signal $r(t)$ is a q-dimensional vector. The resulting value of q depends on the selected options `OPTIONS.rdim` and `OPTIONS.minimal` (see **Method**). For the system object `Q`, two input groups `'outputs'` and `'controls'` are defined for $y(t)$ and $u(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r(t)$.

`R` is the resulting internal form of the fault detection filter and has a standard state-space representation of the form

$$\begin{aligned} \lambda x_R(t) &= A_Q x_R(t) + B_{R_f} f(t) + B_{R_w} w(t) + B_{R_v} v(t), \\ r(t) &= C_Q x_R(t) + D_{R_f} f(t) + D_{R_w} w(t) + D_{R_v} v(t). \end{aligned} \tag{142}$$

The input groups `'faults'`, `'noise'` and `'aux'` are defined for $f(t)$, $w(t)$, and $v(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r(t)$. Notice that the realizations of `Q` and `R` share the matrices A_Q and C_Q.

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

`INFO` **fields**	**Description**
`tcond`	maximum of the condition numbers of the employed non-orthogonal transformation matrices; a warning is issued if `INFO.tcond` $\geq$ `OPTIONS.tcond`.
`degs`	If `OPTIONS.nullspace` = `true`, then: if `OPTIONS.simple` = `true`, the orders of the basis vectors of the employed simple left nullspace basis of $G_1(\lambda)$; if `OPTIONS.simple` = `false`, the degrees of the basis vectors of an equivalent polynomial nullspace basis (see **Method**). If `OPTIONS.nullspace = false` then `INFO.degs = [ ]`.
`degs2`	if `OPTIONS.simple` = `true`, the orders of the basis vectors of the employed simple left nullspace basis of $\overline{G}_w(\lambda)$; if `OPTIONS.simple` = `false`, the degrees of the basis vectors of an equivalent polynomial nullspace basis (see **Method**)
`S`	binary structure matrix S_1 corresponding to $H_1\overline{G}_f(\lambda)$ (see **Method**)
`S2`	binary structure matrix S_2 corresponding to $H_2\overline{G}_f^{(2)}(\lambda)$ (see **Method**)
`HDesign`	design matrix H_1 employed for the synthesis of the fault detection filter (see **Method**)
`HDesign2`	design matrix H_2 employed for the synthesis of the fault detection filter (see **Method**)
`freq`	complex frequency value employed to check the full row rank admissibility condition (see **Method**)
`gap`	achieved gap $\|R_f(\lambda)\|_{\infty -}/\|R_w(\lambda)\|_\infty$, where the $\mathcal{H}_-$-index is computed over the whole frequency range, if `OPTIONS.FDFreq` is empty, or over the frequency values contained in `OPTIONS.FDFreq`. (see **Method**)

Method

The function `afdsyn` implements an extension of the **Procedure AFD** from [16, Sect. 5.3] as proposed in *Remark 5.10* in [16], which relies on the nullspace-based synthesis method proposed in [9], with extensions discussed in [1]. In what follows, we discuss succinctly the main steps of this extended procedure, in terms of the input-output descriptions. Full details of the employed state-space based computational algorithms are given in [16][Chapter 7].

Let assume the system `SYSF` in (140) has the equivalent input-output form

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_f(\lambda)\mathbf{f}(\lambda) + G_w(\lambda)\mathbf{w}(\lambda) + G_v(\lambda)\mathbf{v}(\lambda), \tag{143}$$

where the vectors y, u, d, f, w and v have dimensions p, m_u, m_d, m_f, m_w and m_v, respectively. The resulting fault detection filter in (141) has the input-output form

$$\mathbf{r}(\lambda) = Q(\lambda)\left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array}\right], \tag{144}$$

where the residual vector $r(t)$ has q components.

The implemented synthesis method essentially determines the filter $Q(\lambda)$ as a stable rational left annihilator of

$$G_1(\lambda) := \left[\begin{array}{cc} G_u(\lambda) & G_d(\lambda) \\ I_{m_u} & 0 \end{array}\right], \tag{145}$$

such that $R_{f_j}(\lambda) \neq 0$, for $j = 1, \ldots, m_f$, where

$$R(\lambda) := \left[\begin{array}{c|c|c} R_f(\lambda) & R_w(\lambda) & R_v(\lambda) \end{array}\right] := Q(\lambda)\left[\begin{array}{c|c|c} G_f(\lambda) & G_w(\lambda) & G_v(\lambda) \\ 0 & 0 & 0 \end{array}\right]. \tag{146}$$

The resulting internal form of the fault detection filter (144) is

$$\mathbf{r}(\lambda) = R(\lambda)\left[\begin{array}{c} \mathbf{f}(\lambda) \\ \mathbf{w}(\lambda) \\ \mathbf{v}(\lambda) \end{array}\right] = R_f(\lambda)\mathbf{f}(\lambda) + R_w(\lambda)\mathbf{w}(\lambda) + R_v(\lambda)\mathbf{v}(\lambda)\,, \tag{147}$$

with $R(\lambda)$, defined in (146), stable.

An initial synthesis step implements the nullspace based synthesis approach and determines $Q(\lambda)$ in the product form $Q(\lambda) = \overline{Q}_1(\lambda)Q_1(\lambda)$, where $Q_1(\lambda)$ is a proper left nullspace basis of $G_1(\lambda)$ in (145). This step ensures the decoupling of control and disturbance inputs in the residual (as is apparent in (147)) and allows to determine of $\overline{Q}_1(\lambda)$ by solving an AFDP for the reduced system

$$\overline{\mathbf{y}}(\lambda) := \overline{G}_f(\lambda)\mathbf{f}(\lambda) + \overline{G}_w(\lambda)\mathbf{w}(\lambda) + \overline{G}_v(\lambda)\mathbf{v}(\lambda)\,, \tag{148}$$

where

$$\left[\begin{array}{c|c|c} \overline{G}_f(\lambda) & \overline{G}_w(\lambda) & \overline{G}_v(\lambda) \end{array}\right] = Q_1(\lambda)\left[\begin{array}{c|c|c} G_f(\lambda) & G_w(\lambda) & G_v(\lambda) \\ 0 & 0 & 0 \end{array}\right], \tag{149}$$

to obtain

$$\mathbf{r}(\lambda) = \overline{Q}_1(\lambda)\overline{\mathbf{y}}(\lambda). \tag{150}$$

If $r_d := \operatorname{rank} G_d(\lambda)$, then $Q_1(\lambda)$ has $p - r_d$ rows, which are the basis vectors of the left nullspace of $G_1(\lambda)$. Let $r_w = \operatorname{rank} \overline{G}_w(\lambda)$, which satisfies $0 \leq r_w \leq p - r_d$. If $r_w = 0$, then we have to solve an EFDP for the reduced system (148) with $\overline{G}_w(\lambda) = 0$ (e.g., by using **Procedure EFD** in [16]), while for $r_w > 0$ and $q \leq r_w$ we have to solve an AFDP, for which **Procedure AFD** in [16] can be applied. In the case $0 < r_w < q \leq p - r_d$, we can apply the approach suggested in *Remark 5.10* in [16]), to determine $\overline{Q}_1(\lambda)$ in the row partitioned form

$$\overline{Q}_1(\lambda) = \left[\begin{array}{c} \overline{Q}_1^{(1)}(\lambda) \\ \overline{Q}_1^{(2)}(\lambda) \end{array}\right],$$

where $\overline{Q}_1^{(1)}(\lambda)$ has $q_1 = r_w$ rows and can be determined by solving an AFDP for the reduced system (148), while $\overline{Q}_1^{(2)}(\lambda)$ has $q_2 = q - r_w$ rows and can be determined by solving an EFDP for the same reduced system (148), but with the noise inputs redefined as disturbances and a part of the fault inputs (those which become undetectable due to the decoupling of noise inputs) redefined as auxiliary inputs. More precisely, let $Q_2^{(2)}(\lambda)$ be a left nullspace basis of $\overline{G}_w(\lambda)$ and let $\overline{Q}_1^{(2)}(\lambda) = \overline{Q}_2^{(2)}(\lambda)Q_2^{(2)}(\lambda)$. Then, a second reduced system is obtained as

$$\overline{\mathbf{y}}^{(2)}(\lambda) := \overline{G}_f^{(2)}(\lambda)\mathbf{f}(\lambda) + \overline{G}_v^{(2)}(\lambda)\mathbf{v}(\lambda)\,, \tag{151}$$

where

$$\left[\ \overline{G}_f^{(2)}(\lambda)\ \middle|\ \overline{G}_v^{(2)}(\lambda)\ \right] = Q_2^{(2)}(\lambda)\left[\ \overline{G}_f(\lambda)\ \middle|\ \overline{G}_v(\lambda)\ \right].$$

To determine $\overline{Q}_2^{(2)}(\lambda)$, we first determine S_2, the structure matrix of $\overline{G}_f^{(2)}(\lambda)$ and separate the fault components in two parts: $f^{(1)}$, which correspond to nonzero columns in S_2 and $f^{(2)}$, which correspond to zero columns in S_2. If we denote $\overline{G}_{f^{(1)}}^{(2)}(\lambda)$ and $\overline{G}_{f^{(2)}}^{(2)}(\lambda)$ the columns of $\overline{G}_f^{(2)}(\lambda)$ corresponding to $f^{(1)}$ and $f^{(2)}$, respectively, we can rewrite the reduced system (151) as

$$\overline{\mathbf{y}}^{(2)}(\lambda) := \overline{G}_{f^{(1)}}^{(2)}(\lambda)\mathbf{f}^{(1)}(\lambda) + \overline{G}_{f^{(2)}}^{(2)}(\lambda)\mathbf{f}^{(2)}(\lambda) + \overline{G}_v^{(2)}(\lambda)\mathbf{v}(\lambda)\,. \tag{152}$$

The filter component $\overline{Q}_2^{(2)}(\lambda)$ can be determined by solving an EFDP for the reduced system (152), with $f^{(1)}$ as fault inputs and $f^{(2)}$ and $v(t)$ as auxiliary inputs.

To check the solvability of the AFDP, let S_1 be the structure matrix of $\overline{G}_f(\lambda)$. According to Corollary 5.4 of [16], the AFDP is solvable if and only if $\overline{G}_{f_j}(\lambda) \neq 0$ for $j = 1, \ldots, m_f$, or equivalently all columns of S_1 are nonzero. More generally, if H_1 is the design matrix specified in a nonempty `OPTIONS.HDesign` (otherwise $H_1 = I_{p-r_d}$ is used) and H_2 is the design matrix specified in a nonempty `OPTIONS.HDesign2` (otherwise $H_2 = I_{p-r_d-r_w}$ is used), then let S_1 be the structure matrix of $H_1\overline{G}_f(\lambda)$ and let S_2 be the structure matrix of $H_2\overline{G}_f^{(2)}(\lambda)$. It follows that the AFDP is solvable if $S = \left[\begin{smallmatrix} S_1 \\ S_2 \end{smallmatrix}\right]$ has all its columns nonzero.

The solution of the AFDP for the reduced system (148) to determine $\overline{Q}_1^{(1)}(\lambda)$ involves the solution of a $\mathcal{H}_{\infty-}/\mathcal{H}_\infty$ optimization problem

$$\beta = \max_{\overline{Q}_1^{(1)}(\lambda)} \left\{ \left\| R_f^{(1)}(\lambda) \right\|_{\infty-} \ \middle|\ \left\| R_w^{(1)}(\lambda) \right\|_\infty \leq \gamma \right\}, \tag{153}$$

where $\left[\, R_f^{(1)}(\lambda)\ R_w^{(1)}(\lambda)\,\right] := \overline{Q}_1^{(1)}(\lambda)\left[\,\overline{G}_f(\lambda)\ \overline{G}_w(\lambda)\,\right]$ and γ is a given upper bound on the resulting $\left\|R_w^{(1)}(\lambda)\right\|_\infty$ (specified via `OPTIONS.gamma`). The $\mathcal{H}_{\infty-}$-index $\|\cdot\|_{\infty-}$ characterizes the complete fault detectability property of the system (143) and is evaluated, for `OPTIONS.FDFreq` empty, as

$$\|R_f^{(1)}(\lambda)\|_{\infty-} := \min_{1\leq j\leq m_f} \|R_{f_j}^{(1)}(\lambda)\|_\infty, \tag{154}$$

while for `OPTIONS.FDFreq` containing a nonempty set of real frequencies, which define a complex frequency domain Ω, the $\mathcal{H}_{\Omega-}$-index

$$\|R_f^{(1)}(\lambda)\|_{\Omega-} := \min_{1\leq j\leq m_f} \left\{ \inf_{\lambda_s\in\Omega} \left\|R_{f_j}^{(1)}(\lambda_s)\right\|_2 \right\} \tag{155}$$

is used instead. The value of the achieved fault-to-noise gap $\eta := \beta/\gamma$ is provided in `INFO.gap` and represents a measure of the noise attenuation quality of the fault detection filter. For $\gamma = 0$, the exact solution of an EFDP with $w \equiv 0$ is targeted and the corresponding gap $\eta = \infty$.

In general, the filter $Q(\lambda)$ and its corresponding internal form $R(\lambda)$ are determined in the partitioned forms

$$Q(\lambda) = \left[\begin{array}{c} Q^{(1)}(\lambda) \\ Q^{(2)}(\lambda) \end{array}\right] = \left[\begin{array}{c} \overline{Q}_1^{(1)}(\lambda) \\ \overline{Q}_2^{(2)}(\lambda)Q_2^{(2)}(\lambda) \end{array}\right] Q_1(\lambda), \quad R(\lambda) = \left[\begin{array}{c} R^{(1)}(\lambda) \\ R^{(2)}(\lambda) \end{array}\right], \tag{156}$$

where the filters $Q^{(1)}(\lambda)$ and $R^{(1)}(\lambda)$ with q_1 residual outputs are a solution of an AFDP, while $Q^{(2)}(\lambda)$ and $R^{(2)}(\lambda)$ with q_2 residual outputs are a solution of an EFDP. In what follows, we first describe the determination of $Q_1(\lambda)$ and $Q_2^{(2)}(\lambda)$, and discuss the verification of the solvability conditions. Then we discuss the determination of the remaining factors $\overline{Q}_1^{(1)}(\lambda)$ and $\overline{Q}_2^{(2)}(\lambda)$.

Computation of $Q_1(\lambda)$

$Q_1(\lambda) = N_l(\lambda)$, with $N_l(\lambda)$ a $\big(p - r_d\big) \times (p + m_u)$ proper rational left nullspace basis satisfying $N_l(\lambda)G_1(\lambda) = 0$, where $r_d := \operatorname{rank} G_d(\lambda)$.

If `OPTIONS.nullspace = true` or $m_d > 0$ or E is singular, then $N_l(\lambda)$ is determined as a minimal proper nullspace basis. In this case, if `OPTIONS.simple = true`, then $N_l(\lambda)$ is determined as a simple rational basis and the orders of the basis vectors are provided in `INFO.degs`. If `OPTIONS.simple = false`, then $N_l(\lambda)$ is determined as a proper rational basis and `INFO.degs` contains the degrees of the basis vectors of an equivalent polynomial nullspace basis. A stable basis is determined if `OPTIONS.FDfreq` is not empty.

If `OPTIONS.nullspace = false`, $m_d = 0$ and E is invertible, then $N_l(\lambda) = [\, I_p \;\; -G_u(\lambda)\,]$ is used, which corresponds to a full-order Luenberger observer. If `OPTIONS.FDfreq` is not empty and the system (140) is unstable, then $\widetilde{N}_l(\lambda) = M(\lambda)N_l(\lambda)$ is used instead $N_l(\lambda)$, where $M(\lambda)$ and $\widetilde{N}_l(\lambda)$ are the stable factors of a left coprime factorization $N_l(\lambda) = M^{-1}(\lambda)\widetilde{N}_l(\lambda)$. In this case `INFO.degs = [ ]`.

Computation of $Q_2^{(2)}(\lambda)$

In the case when $r_w < p - r_d$, $Q_2^{(2)}(\lambda) = \overline{N}_{l,w}(\lambda)$, with $\overline{N}_{l,w}(\lambda)$ a $\big(p - r_d - r_w\big) \times (p - r_d)$ proper left nullspace basis satisfying $\overline{N}_{l,w}(\lambda)\overline{G}_w(\lambda) = 0$, where $r_w := \operatorname{rank} \overline{G}_w(\lambda)$. It follows, that $\overline{N}_{l,w}(\lambda)N_l(\lambda)$ is a proper left nullspace basis of

$$G_2(\lambda) := \left[\begin{array}{ccc} G_u(\lambda) & G_d(\lambda) & G_w(\lambda) \\ I_{m_u} & 0 & 0 \end{array}\right]. \tag{157}$$

If `OPTIONS.simple = true`, then $\overline{N}_{l,w}(\lambda)N_l(\lambda)$ is determined as a simple rational basis and the orders of the basis vectors are provided in `INFO.degs2`. If `OPTIONS.simple = false`, then $\overline{N}_{l,w}(\lambda)N_l(\lambda)$ is determined as a proper rational basis and `INFO.degs2` contains the degrees of the basis vectors of an equivalent polynomial nullspace basis. A stable basis is determined if `OPTIONS.FDfreq` is not empty.

Checking the solvability conditions of the AFDP

Let $\overline{G}_f(\lambda) := Q_1(\lambda)\left[\begin{smallmatrix} G_f(\lambda) \\ 0 \end{smallmatrix}\right]$ and $\overline{G}_w(\lambda) := Q_1(\lambda)\left[\begin{smallmatrix} G_w(\lambda) \\ 0 \end{smallmatrix}\right]$ be the TFMs of the reduced model (148) and let $\overline{G}_f^{(2)}(\lambda) = Q_2^{(2)}(\lambda)\overline{G}_f(\lambda)$ be the TFM from the fault inputs in the reduced model (151). To check the solvability of the AFDP, the structure matrices S_1 of $H_1\overline{G}_f(\lambda)$ and S_2 of $H_2\overline{G}_f^{(2)}(\lambda)$ are determined, where H_1 is a full row rank design matrix with $q_1 \leq p - r_d$ rows specified in a nonempty `OPTIONS.HDesign` (otherwise $H_1 = I_{p-r_d}$ is used) and H_2 is a full row rank design matrix specified with $q_2 \leq p - r_d - r_w$ rows in a nonempty `OPTIONS.HDesign2` (otherwise $H_2 = I_{p-r_d-r_w}$ is used). The AFDP is solvable if $S = \left[\begin{smallmatrix} S_1 \\ S_2 \end{smallmatrix}\right]$ has all its columns nonzero. The resulted S_1 and S_2 are provided in `INFO.S` and `INFO.S2`, respectively.

Computation of $\overline{Q}_1^{(1)}(\lambda)$

If $r_w > 0$, the filter $\overline{Q}_1^{(1)}(\lambda)$ is determined in the product form

$$\overline{Q}_1^{(1)}(\lambda) = Q_4^{(1)}(\lambda)Q_3^{(1)}(\lambda)Q_2^{(1)}(\lambda), \tag{158}$$

where the factors are determined as follows:

(a) $Q_2^{(1)}(\lambda)$ is an admissible regularization factor;

(b) $Q_3^{(1)}(\lambda)$ represents an optimal choice which maximizes the gap η;

(c) $Q_4^{(1)}(\lambda)$ is a stable invertible factor determined such that $Q^{(1)}(\lambda)$ and $R^{(1)}(\lambda)$ have a desired dynamics.

The computations of individual factors depend on the user's options and specific choices are discussed in what follows.

Computation of $Q_2^{(1)}(\lambda)$

Let q = `OPTIONS.rdim` if `OPTIONS.rdim` is nonempty. If `OPTIONS.rdim` is empty, q_1 is set to a default value as follows: if `OPTIONS.HDesign` is empty, then $q_1 = 1$ if `OPTIONS.minimal` = `true`, or $q_1 = r_w$, if `OPTIONS.minimal` = `false`. If `OPTIONS.HDesign` is nonempty, then q_1 is the row dimension of the full row rank design matrix H_1 contained in `OPTIONS.HDesign`. To be admissible, H_1 must also fulfill $\operatorname{rank} H_1\overline{G}_w(\lambda) = q_1$.

If `OPTIONS.minimal = false`, then $Q_2^{(1)}(\lambda) = H_1$, where H_1 is a suitable $q_1 \times \big(p - r_d\big)$ full row rank design matrix. If both `OPTIONS.HDesign` and `OPTIONS.HDesign2` are empty, then H_1 is chosen to build $q_1 = \min(q, r_w)$ linear combinations of the $p - r_d$ left nullspace basis vectors, such that $\operatorname{rank} H_1\overline{G}_w(\lambda) = q_1$, and, additionally, $H_1\overline{G}_f(\lambda)$ has the same nonzero columns as S_1. If `OPTIONS.HDesign` is empty but `OPTIONS.HDesign2` is nonempty, then $q_1 = q - q_2$. If $q_1 = p - r_d$ then the choice $H_1 = I_{p-r_d}$ is used, otherwise H_1 is chosen a randomly generated $q_1 \times \big(p - r_d\big)$ real matrix.

If `OPTIONS.minimal = true`, then $Q_2^{(1)}(\lambda)$ is a $q_1 \times \big(p - r_d\big)$ transfer function matrix, with q_1 chosen as above. $Q_2^{(1)}(\lambda)$ is determined as

$$Q_2^{(1)}(\lambda) = \widetilde{Q}_2(\lambda)\,,$$

where $\widetilde{Q}_2(\lambda) := H_1 + Y_2(\lambda)$, $\widetilde{Q}_2(\lambda)Q_1(\lambda)$ $\big(= H_1 N_l(\lambda) + Y_2(\lambda)N_l(\lambda)\big)$ and $Y_2(\lambda)$ are the least order solution of a left minimal cover problem [15]. If `OPTIONS.HDesign` is nonempty, then $H_1 =$ `OPTIONS.HDesign`, and if `OPTIONS.HDesign` is empty, then q_1 is chosen as above and a suitable randomly generated H_1 is employed (see above). To be admissible, $\widetilde{Q}_2(\lambda)$ must fulfill $\operatorname{rank} \widetilde{Q}_2(\lambda)\overline{G}_w(\lambda) = q_1$ and, additionally, $\widetilde{Q}_2(\lambda)\overline{G}_f(\lambda)$ has the same nonzero columns as S_1. The above rank condition is checked as

$$\operatorname{rank} \widetilde{G}_w(\lambda_s) = q_1,$$

where $\widetilde{G}_w(\lambda) := \widetilde{Q}_2(\lambda)\overline{G}_w(\lambda)$ and λ_s is a suitable frequency value, which can be specified via the `OPTIONS.freq`. In the case when `OPTIONS.freq` is empty, the employed frequency value λ_s is provided in `INFO.freq`.

Computation of $Q_3^{(1)}(\lambda)$

Let redefine $\widetilde{G}_w(\lambda) := Q_2^{(1)}(\lambda)\overline{G}_w(\lambda)$ and compute the quasi-co-outer–co-inner factorization of $\widetilde{G}_w(\lambda)$ as

$$\widetilde{G}_w(\lambda) = R_{wo}(\lambda)R_{wi}(\lambda),$$

where $R_{wo}(\lambda)$ is an invertible quasi-co-outer factor and $R_{wi}(\lambda)$ is a (full row rank) co-inner factor. In the *standard case* $R_{wo}(\lambda)$ is outer (i.e., has no zeros on the boundary of the stability domain $\partial\mathbb{C}_s$) and we choose $Q_3^{(1)}(\lambda) = R_{wo}^{-1}(\lambda)$. This is an optimal choice which ensures that the optimal gap η is achieved.

In the *non-standard case* $R_{wo}(\lambda)$ is only quasi-outer and thus, has zeros on the boundary of the stability domain $\partial\mathbb{C}_s$. Depending on the selected option to handle nonstandard optimization problems `OPTIONS.nonstd`, several choices are possible for $Q_3^{(1)}(\lambda)$ in this case:

- If `OPTIONS.nonstd` $= 1$, then $Q_3^{(1)}(\lambda) = R_{wo}^{-1}(\lambda)$ is used.
- If `OPTIONS.nonstd` $= 2$, then a modified co-outer–co-inner factorization of $[\, R_{wo}(\lambda)\ \epsilon I\,]$ is computed, whose co-outer factor $R_{wo,\epsilon}(\lambda)$ satisfies

  $$R_{wo,\epsilon}(\lambda)\big(R_{wo,\epsilon}(\lambda)\big)^{\sim} = \epsilon^2 I + R_{wo}(\lambda)\big(R_{wo}(\lambda)\big)^{\sim}.$$

  Then $Q_3^{(1)}(\lambda) = R_{wo,\epsilon}^{-1}(\lambda)$ is used. The value of the regularization parameter ϵ is specified via `OPTIONS.epsreg`.
- If `OPTIONS.nonstd` $= 3$, then a Wiener-Hopf type co-outer–co-inner factorization is computed in the form

  $$\widetilde{G}_w(\lambda) = R_{wo}(\lambda)R_{wb}(\lambda)R_{wi}(\lambda), \tag{159}$$

  where $R_{wo}(\lambda)$ is co-outer, $R_{wi}(\lambda)$ is co-inner, and $R_{wb}(\lambda)$ is a square stable factor whose zeros are precisely the zeros of $\widetilde{G}_w(\lambda)$ in $\partial\mathbb{C}_s$. $Q_3^{(1)}(\lambda)$ is determined as before $Q_3^{(1)}(\lambda) = R_{wo}^{-1}(\lambda)$.

- If `OPTIONS.nonstd` $= 4$, then the Wiener-Hopf type co-outer–co-inner factorization (159) is computed and $Q_3^{(1)}(\lambda)$ is determined as $Q_3^{(1)}(\lambda) = R_{wb}^{-1}(\tilde{\lambda})R_{wo}^{-1}(\lambda)$, where $\tilde{\lambda}$ is a small perturbation of λ to move all zeros of $R_{wb}(\lambda)$ into the stable domain. In the continuous-time case $\tilde{s} = \frac{s-\beta_z}{1-\beta_z s}$, while in the discrete-time case $\tilde{z} = z/\beta_z$, where the zero shifting parameter β_z is the prescribed stability degree for the zeros specified in `OPTIONS.sdegzer`. For the evaluation of $R_{wb}(\tilde{\lambda})$, a suitable bilinear transformation is performed.

- If `OPTIONS.nonstd` $= 5$, then the Wiener-Hopf type co-outer–co-inner factorization (159) is computed and $Q_3^{(1)}(\lambda)$ is determined as $Q_3^{(1)}(\lambda) = R_{wb,\epsilon}^{-1}(\lambda)R_{wo}^{-1}(\lambda)$, where $R_{wb,\epsilon}(\lambda)$ is the co-outer factor of the co-outer–co-inner factorization of $\left[\, R_{wb}(\lambda)\ \epsilon I \,\right]$ and satisfies

$$R_{wb,\epsilon}(\lambda)\big(R_{wb,\epsilon}(\lambda)\big)^{\sim} = \epsilon^2 I + R_{wb}(\lambda)\big(R_{wb}(\lambda)\big)^{\sim}.$$

  The value of the regularization parameter ϵ is specified via `OPTIONS.epsreg`.

A typical feature of the non-standard case is that, with the exception of using the option `OPTIONS.nonstd` $= 3$, all other choices of `OPTIONS.nonstd` lead to a poor dynamical performance of the resulting filter, albeit an arbitrary large gap η can be occasionally achieved.

Computation of $Q_4^{(1)}(\lambda)$

In the standard case, $Q_4^{(1)}(\lambda) = I$. In the non-standard case, $Q_4^{(1)}(\lambda)$ is a stable invertible transfer function matrix determined such that $Q^{(1)}(\lambda)$ and $R^{(1)}(\lambda)$ in (156) have a desired dynamics (specified via `OPTIONS.sdeg` and `OPTIONS.poles`).

Computation of $\overline{Q}^{(2)}(\lambda)$

If $0 \leq r_w < p - r_d$ or `OPTIONS.exact = true`, then the filter $Q^{(2)}(\lambda)$ in (156) is determined with $\overline{Q}^{(2)}(\lambda)$ in the product form

$$\overline{Q}^{(2)}(\lambda) = Q_4^{(2)}(\lambda)Q_3^{(2)}(\lambda), \tag{160}$$

where the factors are determined as follows:

(a) $Q_3^{(2)}(\lambda)$ is an admissible regularization factor;

(b) $Q_4^{(2)}(\lambda)$ is a stable invertible factor determined such that $Q^{(2)}(\lambda)$ and $R^{(2)}(\lambda)$ in (156) have a desired dynamics.

The computations of individual factors depend on the user's options and specific choices are discussed in what follows.

Computation of $Q_3^{(2)}(\lambda)$

Let q = `OPTIONS.rdim` if `OPTIONS.rdim` is nonempty. If `OPTIONS.rdim` is empty, q_2 is set to a default value as follows: if `OPTIONS.HDesign2` is empty, then $q_2 = 1 - \min(1, r_w)$ if `OPTIONS.minimal` $=$ `true`, or $q_2 = p - r_d - r_w$, if `OPTIONS.minimal` $=$ `false`. In the case when `OPTIONS.HDesign2` is nonempty, then q_2 is the row dimension of the full row rank design matrix H_2 contained in `OPTIONS.HDesign2`.

If `OPTIONS.minimal = false`, then $Q_3^{(2)}(\lambda) = H_2$, where H_2 is a suitable $q_2 \times \big(p - r_d - r_w\big)$ full row rank design matrix. If both `OPTIONS.HDesign` and `OPTIONS.HDesign2` are empty, then H_2 is chosen to build $q_2 = \min(q, r_w)$ linear combinations of the $p - r_d - r_w$ left nullspace basis vectors, such that $H_2\overline{G}_f^{(2)}(\lambda)$ has the same nonzero columns as S_2. If `OPTIONS.HDesign2` is empty but `OPTIONS.HDesign` is nonempty, then $q_2 = q - q_1$. If $q_2 = p - r_d - r_w$ then the choice $H_2 = I_{p-r_d-r_w}$ is used, otherwise H_2 is chosen a randomly generated $q_2 \times \big(p - r_d - r_w\big)$ real matrix.

If `OPTIONS.minimal = true`, then $Q_3^{(2)}(\lambda)$ is a $q_2 \times \big(p - r_d - r_w\big)$ transfer function matrix, with q_2 chosen as above. $Q_3^{(2)}(\lambda)$ is determined as

$$Q_3^{(2)}(\lambda) = \widetilde{Q}_3(\lambda)\,,$$

where $\widetilde{Q}_3(\lambda) := H_2 + Y_3(\lambda)$, $\widetilde{Q}_3(\lambda)Q_2^{(2)}(\lambda)$ $\big(= H_2\overline{N}_{l,w}(\lambda) + Y_3(\lambda)\overline{N}_{l,w}(\lambda)\big)$ and $Y_3(\lambda)$ are the least order solution of a left minimal cover problem [15]. If `OPTIONS.HDesign2` is nonempty, then $H_2 =$ `OPTIONS.HDesign`, and if `OPTIONS.HDesign2` is empty, then q_2 is chosen as above and a suitable randomly generated H_2 is employed (see above). To be admissible, $\widetilde{Q}_2(\lambda)$ must ensure that $\widetilde{Q}_3(\lambda)\overline{G}_f^{(2)}(\lambda)$ has the same nonzero columns as S_2.

Computation of $Q_4^{(2)}(\lambda)$

$Q_4^{(2)}(\lambda)$ is a stable invertible transfer function matrix determined such that $Q^{(2)}(\lambda)$ and $R^{(2)}(\lambda)$ in (156) have a desired dynamics (specified via `OPTIONS.sdeg` and `OPTIONS.poles`).

Example

Example 8. This is Example 5.3 from the book [16], with the disturbance redefined as a noise input and by adding a sensor fault for the first output measurement. The TFMs of the system are:

$$G_u(s) = \begin{bmatrix} \dfrac{s+1}{s+2} \\[2mm] \dfrac{s+2}{s+3} \end{bmatrix}, \quad G_d(s) = 0, \quad G_f(s) = [\, G_u(s)\; I\,], \quad G_w(s) = \begin{bmatrix} \dfrac{s-1}{s+2} \\[2mm] 0 \end{bmatrix},$$

where the fault input f_1 corresponds to an additive actuator fault, while the fault inputs f_2 and f_3 describe additive sensor faults in the outputs y_1 and y_2, respectively. The transfer function matrix $G_w(s)$ is non-minimum phase, having an unstable zero at 1. Interestingly, the EFDP formulated with $G_d(s) = G_w(s)$ is not solvable.

We want to design a least order fault detection filter $Q(s)$, which fulfills:

– the decoupling condition: $Q(s)\left[\begin{smallmatrix} G_u(s) & G_d(s) \\ I_{mu} & 0 \end{smallmatrix}\right] = 0$;

– the fault detectability condition: $\|R_f(s)\|_{\infty-} > 0$;

– the maximization of the noise attenuation gap: $\eta := \|R_f(s)\|_{\infty-}/\|R_w(s)\|_\infty = \max$.

The results computed with the following script are

$$Q(s) = \left[\frac{s+2}{s+1} \; \frac{s+3}{s+1} \; -\frac{2s+3}{s+1} \right], \quad R_f(s) = \left[\frac{2s+3}{s+1} \; \frac{s+2}{s+1} \; \frac{s+3}{s+1} \right], \quad R_w(s) = \frac{s-1}{s+1}.$$

```
% Example - Solution of an approximate fault detection problem (AFDP)

% Example 5.3 of (V,2017) (modified)
s = tf('s'); % define the Laplace variable s
Gu = [(s+1)/(s+2); (s+2)/(s+3)];  % enter Gu(s)
Gw = [(s-1)/(s+2); 0];            % enter Gw(s)

% build model with additive faults with Gf = [Gu eye(p)];
sysf = fdimodset(ss([Gu Gw]),struct('c',1,'n',2,'f',1,'fs',1:2));
mu = 1; mw = 1; p = 2; mf = mu+p; % set dimensions


% perform synthesis with  AFDSYN, using default options
[Q,R,info] = afdsyn(sysf);  % R(s) = [Rf(s) Rw(s)]

% display the implementation form Q(s) and the internal forms Rf(s) and Rw(s)
% of resulting fault detection filter
tf(Q), tf(R(:,'faults')),   tf(R(:,'noise'))

% display the resulting gap and fault condition number
info.gap
FSCOND = fdifscond(R)

% check results: R(s) := Q(s)*Ge(s) = [0 Rf(s) Rw(s)],
% with Ge(s) = [Gu(s) Gf(s) Gw(s); I 0 0]
syse = [sysf;eye(mu,mu+mf+mw)]; % form Ge(s)
norm_Ru = norm(Q*syse(:,'controls'),inf)
norm_rez = norm(Q*syse(:,{'faults','noise'})-R,inf)
gap = fdif2ngap(R,0)           % check gap

% check strong fault detectability
S_strong = fdisspec(R(:,'faults'))

% simulate step responses from fault and noise inputs
inpnames = {'f_1','f_2','f_3','w'};
set(R,'InputName',inpnames,'OutputName','r');
step(R); ylabel('')
title('Step responses from fault and noise inputs')
```

4.8.3 `efdisyn`

Syntax

```
[Q,R,INFO] = efdisyn(SYSF,OPTIONS)
```

Description

`efdisyn` solves the *exact fault detection and isolation problem* (EFDIP) (see Section 2.6.3), for a given LTI system `SYSF` with additive faults and a given structure matrix S_{FDI} (specified via the `OPTIONS` structure). Two banks of stable and proper filters are computed in the n_b-dimensional cell arrays `Q` and `R`, where n_b is the number of specifications contained in S_{FDI} (i.e., the number of rows of the structure matrix S_{FDI}). `Q{i}` contains the i-th fault detection filter (12) in the overall solution (11) of the EFDIP and `R{i}` contains its internal form.

Input data

`SYSF` is a LTI system in the state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_u u(t) + B_d d(t) + B_f f(t) + B_w w(t),\\ y(t) &= Cx(t) + D_u u(t) + D_d d(t) + D_f f(t) + D_w w(t), \end{aligned} \tag{161}$$

where any of the inputs components $u(t)$, $d(t)$, $f(t)$, or $w(t)$ can be void. For the system `SYSF`, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively.

`OPTIONS` is a MATLAB structure used to specify various synthesis options and has the following fields:

`OPTIONS` **fields**	**Description**
`SFDI`	the desired structure matrix S_{FDI} to solve the EFDIP (Default: `[1 ... 1]`, i.e., solve an exact fault detection problem)
`tol`	relative tolerance for rank computations (Default: internally computed)
`tolmin`	absolute tolerance for observability tests (Default: internally computed)
`FDTol`	threshold for fault detectability checks (Default: 0.0001)
`FDGainTol`	threshold for strong fault detectability checks (Default: 0.01)

`rdim`	vector, whose i-th component q_i, specifies the desired number of residual outputs for the i-th component filters `Q`$\{i\}$ and `R`$\{i\}$; if `OPTIONS.rdim` is a scalar q, then a vector with all components $q_i = q$ is assumed. (Default: `[ ]`, in which case: – if `OPTIONS.HDesign`$\{i\}$ is empty, then $q_i = 1$, if `OPTIONS.minimal` = `true`, or q_i is the dimension of the left nullspace which underlies the synthesis of `Q`$\{i\}$ and `R`$\{i\}$, if `OPTIONS.minimal` = `false`; – if `OPTIONS.HDesign`$\{i\}$ is nonempty, then q_i is the row dimension of the design matrix contained in `OPTIONS.HDesign`$\{i\}$.)
`FDFreq`	vector of real frequency values for strong detectability checks (Default: `[ ]`)
`smarg`	stability margin for the poles of the component filters `Q`$\{i\}$ and `R`$\{i\}$ (Default: `-sqrt(eps)` for a continuous-time system `SYSF`; `1-sqrt(eps)` for a discrete-time system `SYSF`).
`sdeg`	prescribed stability degree for the poles of the component filters `Q`$\{i\}$ and `R`$\{i\}$ (Default: -0.05 for a continuous-time system `SYSF`; 0.95 for a discrete-time system `SYSF`).
`poles`	complex vector containing a complex conjugate set of desired poles (within the stability domain) to be assigned for the component filters `Q`$\{i\}$ and `R`$\{i\}$ (Default: `[ ]`)
`nullspace`	option to use a specific proper nullspace basis to be employed at the initial reduction step `true` – use a minimal proper basis (default); `false` – use a full-order observer based basis (see **Method**). *Note:* This option can only be used if no disturbance inputs are present in (161) and E is invertible.
`simple`	option to employ simple proper bases for the synthesis of the component filters `Q`$\{i\}$ and `R`$\{i\}$ `true` – use simple bases; `false` – use non-simple bases (default)
`minimal`	option to perform least order synthesis of the component filters `Q`$\{i\}$ and `R`$\{i\}$ `true` – perform least order synthesis (default); `false` – perform full order synthesis
`tcond`	maximum alowed condition number of the employed non-orthogonal transformations (Default: 10^4).
`FDSelect`	integer vector with increasing elements containing the indices of the desired filters to be designed (Default: $[1, \ldots, n_b]$)

`HDesign`	n_b-dimensional cell array; `OPTIONS.HDesign`$\{i\}$, if not empty, is a full row rank design matrix employed for the synthesis of the i-th fault detection filter (Default: `[ ]`)

Output data

`Q` an n_b-dimensional cell array, with `Q`$\{i\}$ containing the resulting i-th filter in a standard state-space representation

$$\begin{aligned} \lambda x_Q^{(i)}(t) &= A_Q^{(i)} x_Q^{(i)}(t) + B_{Q_y}^{(i)} y(t) + B_{Q_u}^{(i)} u(t), \\ r^{(i)}(t) &= C_Q^{(i)} x_Q^{(i)}(t) + D_{Q_y}^{(i)} y(t) + D_{Q_u}^{(i)} u(t), \end{aligned}$$

where the residual signal $r^{(i)}(t)$ is a q_i-dimensional vector. For each system object `Q`$\{i\}$, two input groups `'outputs'` and `'controls'` are defined for $y(t)$ and $u(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r^{(i)}(t)$. `Q`$\{i\}$ is empty if the index i is not selected in `OPTIONS.FDSelect`.

`R` an n_b-dimensional cell array, with `R`$\{i\}$ containing the resulting internal form of the i-th filter in a standard state-space representation

$$\begin{aligned} \lambda x_R^{(i)}(t) &= A_Q^{(i)} x_R^{(i)}(t) + B_{R_f}^{(i)} f(t) + B_{R_w}^{(i)} w(t), \\ r^{(i)}(t) &= C_Q^{(i)} x_R^{(i)}(t) + D_{R_f}^{(i)} f(t) + D_{R_w}^{(i)} w(t). \end{aligned}$$

The input groups `'faults'` and `'noise'` are defined for $f(t)$, and $w(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r^{(i)}(t)$. Note that the realizations of `Q`$\{i\}$ and `R`$\{i\}$ share the matrices $A_Q^{(i)}$ and $C_Q^{(i)}$. `R`$\{i\}$ is empty if the index i is not selected in `OPTIONS.FDSelect`.

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

`INFO` **fields**	**Description**
`tcond`	n_b-dimensional vector; `INFO.tcond`(i) contains the maximum of the condition numbers of the employed non-orthogonal transformation matrices to determine the i-th filter component `Q`$\{i\}$; a warning is issued if any `INFO.tcond`$(i) \geq$ `OPTIONS.tcond`.
`degs`	n_b-dimensional cell array; if `OPTIONS.simple = true`, `INFO.degs`$\{i\}$ contains the orders of the basis vectors of the employed simple nullspace basis for the synthesis of the i-th filter component `Q`$\{i\}$; if `OPTIONS.simple = false`, `INFO.degs`$\{i\}$ contains the degrees of the basis vectors of an equivalent polynomial nullspace basis
`HDesign`	n_b-dimensional cell array; `INFO.HDesign`$\{i\}$ is the i-th design matrix actually employed for the synthesis of the i-th fault detection filter `Q`$\{i\}$. `INFO.HDesin`$\{i\}$ is empty if the index i is not selected in `OPTIONS.FDSelect`.

Method

The **Procedure EFDI** from [16, Sect. 5.4] is implemented, which relies on the nullspace-based synthesis method proposed in [7]. This method essentially determines each filter $Q^{(i)}(\lambda)$ and its internal form $R^{(i)}(\lambda)$, by solving a suitably formulated EFDP for a reduced system without control inputs, and with redefined disturbance and fault inputs. For this purpose, the function `efdisyn` calls internally the function `efdsyn` to solve a suitably formulated EFDP for each specification (i.e., row) contained in the structure matrix S_{FDI}.

If the faulty system `SYSF` has the input-output form

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_f(\lambda)\mathbf{f}(\lambda) + G_w(\lambda)\mathbf{w}(\lambda) \tag{162}$$

and the i-th fault detection filter $Q^{(i)}(\lambda)$ contained in `Q`$\{i\}$ has the input-output form

$$\mathbf{r}^{(i)}(\lambda) = Q^{(i)}(\lambda)\left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array}\right], \tag{163}$$

then, taking into account the decoupling conditions (19), the resulting internal form of the i-th fault detection filter $R^{(i)}(\lambda)$, contained in `R`$\{i\}$, is

$$\mathbf{r}^{(i)}(\lambda) = R^{(i)}(\lambda)\left[\begin{array}{c} \mathbf{f}(\lambda) \\ \mathbf{w}(\lambda) \end{array}\right] = R_f^{(i)}(\lambda)\mathbf{f}(\lambda) + R_w^{(i)}(\lambda)\mathbf{w}(\lambda), \tag{164}$$

with $R^{(i)}(\lambda) = [\, R_f^{(i)}(\lambda)\; R_w^{(i)}(\lambda)\,]$ defined as

$$\left[\begin{array}{c|c} R_f^{(i)}(\lambda) & R_w^{(i)}(\lambda) \end{array}\right] := Q^{(i)}(\lambda)\left[\begin{array}{c|c} G_f(\lambda) & G_w(\lambda) \\ 0 & 0 \end{array}\right]. \tag{165}$$

In accordance with (23), the structure (row) vector corresponding to the zero and nonzero columns of the transfer function matrix $R_f^{(i)}(\lambda)$ is equal to the i-th row of the specified S_{FDI}.

Each filter $Q^{(i)}(\lambda)$ is determined in the product form

$$Q^{(i)}(\lambda) = \overline{Q}_1^{(i)}(\lambda)Q_1(\lambda), \tag{166}$$

where the factors are determined as follows:

(a) $Q_1(\lambda) = N_l(\lambda)$, with $N_l(\lambda)$ a $(p-r_d)\times(p+m_u)$ proper rational left nullspace basis satisfying $N_l(\lambda)\left[\begin{smallmatrix} G_u(\lambda) & G_d(\lambda) \\ I_{m_u} & 0 \end{smallmatrix}\right] = 0$, with $r_d := \operatorname{rank} G_d(\lambda)$;

(b) $\overline{Q}_1^{(i)}(\lambda)$ is the solution of a suitably formulated EFDP to achieve the specification contained in the i-th row of S_{FDI}. The function `efdsyn` is called for this purpose and internally uses a design matrix $H^{(i)}$, which can be specified in `OPTIONS.HDesign`$\{i\}$ (see **Method** for `efdsyn`). The actually employed design matrix is returned in `INFO.HDesign`$\{i\}$.

The computations to determine the individual factors $\overline{Q}^{(i)}(\lambda)$ depend on the user's options (see **Method** for the function `efdsyn`). In what follows, we only discuss shortly the computation of $Q_1(\lambda)$, performed only once as an initial reduction step.

If `OPTIONS.nullspace = true` or $m_d > 0$ or E is singular, then $N_l(\lambda)$ is determined as a minimal proper nullspace basis. In this case, only orthogonal transformations are performed at this computational step.

If `OPTIONS.nullspace = false`, $m_d = 0$ and E is invertible, then $N_l(\lambda) = [\, I_p \;\; -G_u(\lambda)\,]$ is used, which corresponds to a full-order Luenberger observer. This option involves no numerical computations.

Examples

Example 9. This is *Example* 5.10 from the book [16], which considers a continuous-time system with triplex sensor redundancy on its measured scalar output, which we denote, respectively, by y_1, y_2 and y_3. Each output is related to the control and disturbance inputs by the input-output relation

$$\mathbf{y}_i(s) = G_u(s)\mathbf{u}(s) + G_d(s)\mathbf{d}(s), \quad i = 1,2,3,$$

where $G_u(s)$ and $G_d(s)$ are $1 \times m_u$ and $1 \times m_d$ TFMs, respectively. We assume all three outputs are susceptible to additive sensor faults. Thus, the input-output model of the system with additive faults has the form

$$\mathbf{y}(s) := \begin{bmatrix} \mathbf{y}_1(s) \\ \mathbf{y}_2(s) \\ \mathbf{y}_3(s) \end{bmatrix} = \begin{bmatrix} G_u(s) \\ G_u(s) \\ G_u(s) \end{bmatrix} \mathbf{u}(s) + \begin{bmatrix} G_d(s) \\ G_d(s) \\ G_d(s) \end{bmatrix} \mathbf{d}(s) + \begin{bmatrix} \mathbf{f}_1(s) \\ \mathbf{f}_2(s) \\ \mathbf{f}_3(s) \end{bmatrix}.$$

The maximal achievable structure matrix is

$$S_{max} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

If we assume that no simultaneous sensor faults occur, then we can target to solve an EFDIP for the structure matrix

$$S_{FDI} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix},$$

where the columns of S_{FDI} codify the desired fault signatures.

The resulting least order overall FDI filter has the generic form (i.e, independent of the numbers of control and disturbance inputs)

$$Q(s) = \begin{bmatrix} Q^{(1)}(s) \\ Q^{(2)}(s) \\ Q^{(3)}(s) \end{bmatrix} = \begin{bmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & -1 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{167}$$

and the corresponding overall internal form is

$$R_f(s) = \begin{bmatrix} R_f^{(1)}(s) \\ R_f^{(2)}(s) \\ R_f^{(3)}(s) \end{bmatrix} = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \tag{168}$$

```
% Example - Solution of an EFDIP

p = 3; mf = 3;   % enter output and fault vector dimensions
% generate random dimensions for system order and input vectors
rng('default')
nu = floor(1+4*rand); mu = floor(1+4*rand);
nd = floor(1+4*rand); md = floor(1+4*rand);
% define random Gu(s) and Gd(s) with triplex sensor redundancy
% and Gf(s) = I for triplex sensor faults
Gu = ones(3,1)*rss(nu,1,mu); % enter Gu(s) in state-space form
Gd = ones(3,1)*rss(nd,1,md); % enter Gd(s) in state-space form

% build synthesis model with sensor faults
sysf = fdimodset([Gu Gd],struct('c',1:mu,'d',mu+(1:md),'fs',1:3));

SFDI = [ 0 1 1; 1 0 1; 1 1 0] > 0;  % enter structure matrix

% set options for least order synthesis with EFDISYN
options = struct('tol',1.e-7,'sdeg',-1,'rdim',1,'SFDI',SFDI);
[Qt,Rft] = efdisyn(sysf,options);

% normalize Q and Rf to match example
scale = sign([ Rft{1}.d(1,2) Rft{2}.d(1,3) Rft{3}.d(1,1)]);
for i = 1:3, Qt{i} = scale(i)*Qt{i}; Rft{i} = scale(i)*Rft{i}; end
Q = vertcat(Qt{:});  Rf = vertcat(Rft{:});
Q = set(Q,'InputName',['y1';'y2';'y3';strseq('u',1:mu)],...
          'OutputName',['r1';'r2';'r3'])
Rf = set(Rf,'InputName',['f1';'f2';'f3'],'OutputName',['r1';'r2';'r3'])

% check synthesis conditions: Q*[Gu Gd;I 0] = 0 and Q*[Gf; 0] = Rf
syse = [sysf;eye(mu,mu+md+mf)];  % form Ge = [Gu Gd Gf;I 0 0];
norm_Ru_Rd = norm(Q*syse(:,{'controls','disturbances'}),inf)
norm_rez = norm(Q*syse(:,'faults')-Rf,inf)

% check strong fault detectability
[S_strong,abs_dcgains] = fdisspec(Rft)

% determine achieved fault sensitivity conditions
FSCOND = fdifscond(Rft,0,SFDI)

% evaluate step responses
set(Rf,'InputName',strseq('f_',1:mf),'OutputName',strseq('r_',1:size(SFDI,1)));
step(Rf);
title('Step responses from the fault inputs')
ylabel('Residuals')
```

Example 10. This is the example of [20], already considered in Example 4. Using `efdisyn`, we can easily determine a bank of least order fault detection filters, which achieve the computed maximal weak structure matrix S_{weak}. With the default least order synthesis option, we obtain a bank of 18 filters, each one of order one or two. The overall filters $Q(s)$ and $R_f(s)$ obtained by stacking the 18 component filters have state-space realizations of order 32, which are usually non-minimal. Typically, minimal realizations of orders about 20 can be computed for each of these filters. The bank of 12 component filters ensuring strong fault detection can easily be picked-out from the computed filters.

In this example, we show that using the pole assignment feature, the overall filters $Q(s)$ and $R_f(s)$ can be determined with minimal realizations of order 6, which is probably the least achievable global order. To arrive to this order, we enforce the same dynamics for all component filters by assigning, for example, all poles of the component filters to lie in the set $\{-1, -2\}$. The resulting least order of the overall filter $Q(s)$ can be easily read-out from the plot of its Hankel-singular values shown in Fig. 5. The synthesis procedure also ensures that $R_f(s)$, and even of the joint overall filter $[\, Q(s)\ R_f(s)\,]$ have minimal realizations of order 6!. It is also straightforward to check that the resulting weak structure matrix of $R_f(s)$ and S_{weak} coincide.

```
% Example of Yuan et al. IJC (1997)
rng('default');  % make results reproducible
p = 3; mu = 1; mf = 8;
A = [ -1 1 0 0; 1 -2 1 0; 0 1 -2 1; 0 0 1 -2  ];
Bu = [1 0 0 0]';
Bf = [ 1 0 0 0 1 0 0 0; 0 1 0 0 -1 1 0 0; 0 0 1 0 0 -1 1 0; 0 0 0 1 0 0 -1 1];
C = [ 1 0 0 0; 0 0 1 0; 0 0 0 1];
Du = zeros(p,mu); Df = zeros(p,mf);
% setup the model with additive faults
sysf = fdimodset(ss(A,[Bu Bf],C,[Du Df]),struct('c',1:mu,'f',mu+(1:mf)));

% compute the achievable weak specifications
opt = struct('tol',1.e-7,'FDTol',1.e-5);
S_weak = fdigenspec(sysf,opt);

% set options for least order synthesis with pole assignment
options = struct('tol',1.e-7,'sdeg',-5,'smarg',-5,'poles',[-1 -2],...
                    'FDTol',0.0001,'rdim',1,'simple',false,'SFDI',S_weak);
[Q,Rf] = efdisyn(sysf,options);

% minimal order of the overall filter is 6!
hsvd(vertcat(Q{:})) % only the first 6 Hankel singular values are nonzero

% check that the achieved structure matrix is the desired one
isequal(S_weak,fditspec(Rf))
```

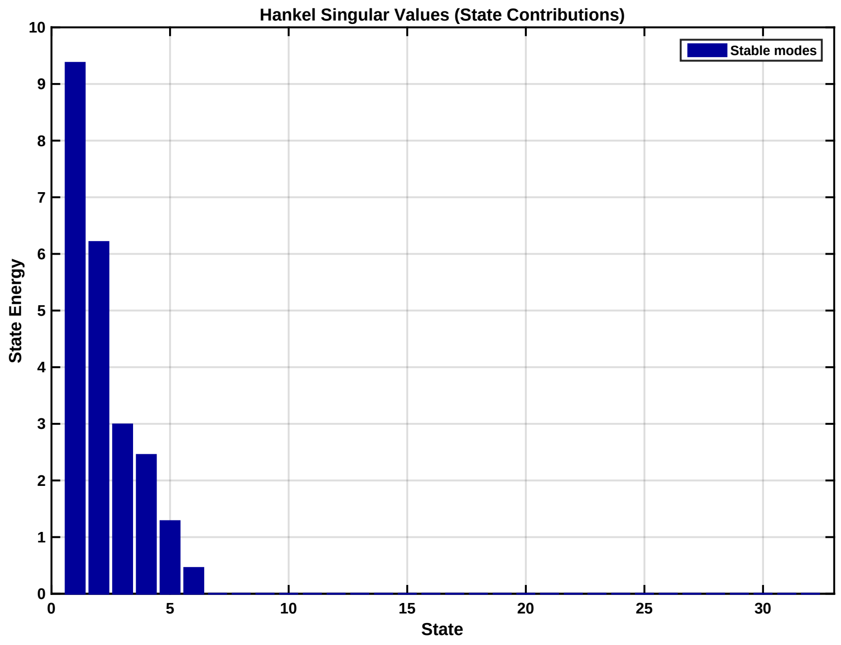

Figure 5: Hankel-singular values of the overall filter $Q(s)$.

4.8.4 `afdisyn`

Syntax

```
[Q,R,INFO] = afdisyn(SYSF,OPTIONS)
```

Description

`afdisyn` solves the *approximate fault detection and isolation problem* (AFDIP) (see Section 2.6.4), for a given LTI system `SYSF` with additive faults and a given structure matrix S_{FDI} (specified via the `OPTIONS` structure). Two banks of stable and proper filters are computed in the n_b-dimensional cell arrays `Q` and `R`, where n_b is the number of specifications contained in S_{FDI} (i.e., the number of rows of the structure matrix S_{FDI}). `Q{i}` contains the i-th fault detection filter (12) in the overall solution (11) of the AFDIP and `R{i}` contains its internal form.

Input data

`SYSF` is a LTI system in the state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_u u(t) + B_d d(t) + B_f f(t) + B_w w(t), \\ y(t) &= Cx(t) + D_u u(t) + D_d d(t) + D_f f(t) + D_w w(t), \end{aligned} \qquad (169)$$

where any of the inputs components $u(t)$, $d(t)$, $f(t)$, or $w(t)$ can be void. For the system `SYSF`, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively.

`OPTIONS` is a MATLAB structure used to specify various synthesis options and has the following fields:

`OPTIONS` **fields**	**Description**
`SFDI`	the desired structure matrix S_{FDI} with n_b rows to solve the AFDIP (Default: `[1...1]`, i.e., solve an approximate fault detection problem)
`tol`	relative tolerance for rank computations (Default: internally computed)
`tolmin`	absolute tolerance for observability tests (Default: internally computed)
`FDTol`	threshold for fault detectability checks (Default: 0.0001)
`FDGainTol`	threshold for strong fault detectability checks (Default: 0.01)
`rdim`	vector, whose i-th component q_i, specifies the desired number of residual outputs for the i-th component filters `Q`$\{i\}$ and `R`$\{i\}$; if `OPTIONS.rdim` is a scalar q, then a vector with all components $q_i = q$ is assumed. (Default: `[ ]`, in which case $q_i = q_{i,1} + q_{i,2}$, with $q_{i,1}$ and $q_{i,2}$ selected taking into account $r_w^{(i)}$, the rank of the transfer function matrix from the noise input to the reduced system output employed to determine `Q`$\{i\}$, as follows: if `OPTIONS.HDesign`$\{i\}$ is empty, then $q_{i,1} = \min\big(1, r_w^{(i)}\big)$, if `OPTIONS.minimal` = `true`, or $q_{i,1} = r_w^{(i)}$ if `OPTIONS.minimal` = `false`; if `OPTIONS.HDesign`$\{i\}$ is nonempty, then $q_{i,1}$ is the row dimension of the design matrix contained in `OPTIONS.HDesign`$\{i\}$; if `OPTIONS.HDesign2`$\{i\}$ is empty, then $q_{i,2} = 1 - \min\big(1, r_w^{(i)}\big)$, if `OPTIONS.minimal` = `true`, or $q_{i,2}$ is set to its maximum achievable value, if `OPTIONS.minimal` = `false`; if `OPTIONS.HDesign2`$\{i\}$ is nonempty, then $q_{i,2}$ is the row dimension of the design matrix contained in `OPTIONS.HDesign2`$\{i\}$.)
`FDFreq`	vector of real frequency values for strong detectability checks (Default: `[ ]`)
`smarg`	stability margin for the poles of the component filters `Q`$\{i\}$ and `R`$\{i\}$ (Default: `-sqrt(eps)` for a continuous-time system `SYSF`; `1-sqrt(eps)` for a discrete-time system `SYSF`).

`sdeg`	prescribed stability degree for the poles of the component filters `Q`$\{i\}$ and `R`$\{i\}$ (Default: -0.05 for a continuous-time system `SYSF`; 0.95 for a discrete-time system `SYSF`).
`poles`	complex vector containing a complex conjugate set of desired poles (within the stability domain) to be assigned for the component filters `Q`$\{i\}$ and `R`$\{i\}$ (Default: `[ ]`)
`nullspace`	option to use a specific proper nullspace basis to be employed at the initial reduction step `true` – use a minimal proper basis (default); `false` – use a full-order observer based basis (see **Method**). *Note:* This option can only be used if no disturbance inputs are present in (169) and E is invertible.
`simple`	option to employ simple proper bases for the synthesis of the component filters `Q`$\{i\}$ and `R`$\{i\}$ `true` – use simple bases; `false` – use non-simple bases (default)
`minimal`	option to perform least order synthesis of the component filters `true` – perform least order synthesis (default); `false` – perform full order synthesis
`exact`	option to perform exact filter synthesis `true` – perform exact synthesis (i.e., no optimization performed); `false` – perform approximate synthesis (default).
`freq`	complex frequency value to be employed to check the full row rank admissibility conditions within the function `afdsyn` (see **Method** for `afdsyn`) (Default:`[ ]`, i.e., a randomly generated frequency).
`tcond`	maximum alowed condition number of the employed non-orthogonal transformations (Default: 10^4).
`FDSelect`	integer vector with increasing elements containing the indices of the desired filters to be designed (Default: $[1,\ldots,n_b]$)
`HDesign`	n_b-dimensional cell array; `OPTIONS.HDesign`$\{i\}$, if not empty, is a full row rank design matrix $H_1^{(i)}$ employed for the synthesis of the i-th filter components $Q^{(1,i)}(\lambda)$ and $R^{(1,i)}(\lambda)$ (see **Method**) (Default: `[ ]`)
`HDesign2`	n_b-dimensional cell array; `OPTIONS.HDesign2`$\{i\}$, if not empty, is a full row rank design matrix $H_2^{(i)}$ employed for the synthesis of the i-th filter components $Q^{(2,i)}(\lambda)$ and $R^{(2,i)}(\lambda)$ (see **Method**) (Default: `[ ]`)
`gamma`	upper bound on the resulting $\\|R_w^{(i)}(\lambda)\\|_\infty$ (see **Method**) (Default: 1)
`epsreg`	regularization parameter used in `afdsyn` (Default: 0.1)

`sdegzer`	prescribed stability degree for zeros shifting (Default: -0.05 for a continuous-time system `SYSF`; 0.95 for a discrete-time system `SYSF`).
`nonstd`	option to handle nonstandard optimization problems used in `afdsyn`: 1 – use the quasi-co-outer–co-inner factorization (default); 2 – use the modified co-outer–co-inner factorization with the regularization parameter `OPTIONS.epsreg`; 3 – use the Wiener-Hopf type co-outer–co-inner factorization. 4 – use the Wiener-Hopf type co-outer-co-inner factorization with zero shifting of the non-minimum phase factor using the stabilization parameter `OPTIONS.sdegzer` 5 – use the Wiener-Hopf type co-outer-co-inner factorization with the regularization of the non-minimum phase factor using the regularization parameter `OPTIONS.epsreg`

Output data

`Q` is an n_b-dimensional cell array, with `Q`$\{i\}$ containing the resulting i-th filter in a standard state-space representation

$$\begin{aligned}\lambda x_Q^{(i)}(t) &= A_Q^{(i)} x_Q^{(i)}(t) + B_{Q_y}^{(i)} y(t) + B_{Q_u}^{(i)} u(t),\\ r^{(i)}(t) &= C_Q^{(i)} x_Q^{(i)}(t) + D_{Q_y}^{(i)} y(t) + D_{Q_u}^{(i)} u(t),\end{aligned}$$

where the residual signal $r^{(i)}(t)$ is a q_i-dimensional vector. For each system object `Q`$\{i\}$, two input groups `'outputs'` and `'controls'` are defined for $y(t)$ and $u(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r^{(i)}(t)$. `Q`$\{i\}$ is empty if the index i is not selected in `OPTIONS.FDSelect`.

`R` is an n_b-dimensional cell array, with `R`$\{i\}$ containing the resulting internal form of the i-th filter in a standard state-space representation

$$\begin{aligned}\lambda x_R^{(i)}(t) &= A_Q^{(i)} x_R^{(i)}(t) + B_{R_f}^{(i)} f(t) + B_{R_w}^{(i)} w(t),\\ r^{(i)}(t) &= C_Q^{(i)} x_R^{(i)}(t) + D_{R_f}^{(i)} f(t) + D_{R_w}^{(i)} w(t).\end{aligned}$$

The input groups `'faults'` and `'noise'` are defined for $f(t)$, and $w(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r^{(i)}(t)$. Note that the realizations of `Q`$\{i\}$ and `R`$\{i\}$ share the matrices $A_Q^{(i)}$ and $C_Q^{(i)}$. `R`$\{i\}$ is empty if the index i is not selected in `OPTIONS.FDSelect`.

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

`INFO` **fields**	**Description**
`tcond`	n_b-dimensional vector; `INFO.tcond`(i) contains the maximum of the condition numbers of the employed non-orthogonal transformation matrices to determine the i-th filter component `Q`$\{i\}$; a warning is issued if any `INFO.tcond`$(i) \geq$ `OPTIONS.tcond`.
`HDesign`	n_b-dimensional cell array; `INFO.HDesign`$\{i\}$ is the i-th design matrix actually employed for the synthesis of the filter component $Q_1^{(i)}$ of the i-th fault detection filter `Q`$\{i\}$ (see **Method**). `INFO.HDesign`$\{i\}$ is empty if the index i is not selected in `OPTIONS.FDSelect`.
`HDesign2`	n_b-dimensional cell array; `INFO.HDesign2`$\{i\}$ is the i-th design matrix actually employed for the synthesis of the filter component $Q_2^{(i)}$ of the i-th fault detection filter `Q`$\{i\}$ (see **Method**). `INFO.HDesign2`$\{i\}$ is empty if the index i is not selected in `OPTIONS.FDSelect`.
`freq`	complex frequency value employed to check the full row rank admissibility conditions within the function `afdsyn`
`gap`	n_b-dimensional vector; `INFO.gap`(i) contains the i-th gap η_i resulted by calling `afdsyn` to determine `Q`$\{i\}$ (see **Method**).

Method

The **Procedure AFDI** from [16, Sect. 5.4] is implemented, which relies on the optimization-based synthesis method proposed in [9]. Each filter $Q^{(i)}(\lambda)$ and its internal form $R^{(i)}(\lambda)$, are determined by solving a suitably formulated AFDP for a reduced system without control inputs, and with redefined disturbance and fault inputs. For this purpose, the function `afdisyn` calls internally the function `afdsyn` to solve a suitably formulated AFDP for each specification (i.e., row) contained in the structure matrix S_{FDI}.

If the faulty system `SYSF` has the input-output form

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_f(\lambda)\mathbf{f}(\lambda) + G_w(\lambda)\mathbf{w}(\lambda) \tag{170}$$

and the i-th fault detection filter $Q^{(i)}(\lambda)$ contained in `Q`$\{i\}$ has the input-output form

$$\mathbf{r}^{(i)}(\lambda) = Q^{(i)}(\lambda)\left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array}\right], \tag{171}$$

then, taking into account the decoupling conditions (19), the resulting internal form of the i-th fault detection filter $R^{(i)}(\lambda)$, contained in `R`$\{i\}$, is

$$\mathbf{r}^{(i)}(\lambda) = R^{(i)}(\lambda)\left[\begin{array}{c} \mathbf{f}(\lambda) \\ \mathbf{w}(\lambda) \end{array}\right] = R_f^{(i)}(\lambda)\mathbf{f}(\lambda) + R_w^{(i)}(\lambda)\mathbf{w}(\lambda)\,, \tag{172}$$

with $R^{(i)}(\lambda) = [\, R_f^{(i)}(\lambda)\; R_w^{(i)}(\lambda)\,]$ defined as

$$\left[\begin{array}{c|c} R_f^{(i)}(\lambda) & R_w^{(i)}(\lambda) \end{array}\right] := Q^{(i)}(\lambda)\left[\begin{array}{c|c} G_f(\lambda) & G_w(\lambda) \\ 0 & 0 \end{array}\right]. \tag{173}$$

When determining $Q^{(i)}(\lambda)$ it is first attempted to solve the *strict* AFDIP (26), such that the structure vector corresponding to the zero and nonzero columns of $R_f^{(i)}(\lambda)$ is equal to the i-th row of the specified S_{FDI}. If this is not achievable, then the *soft* AFDIP (25) is attempted to be solved.

The achieved gap η_i is computed as $\eta_i = \big\|\overline{R}_f^{(i)}(\lambda)\big\|_{\infty-}/\big\|\big[\widetilde{R}_f^{(i)}(\lambda)\; R_w^{(i)}(\lambda)\big]\big\|_{\infty}$, where $\overline{R}_f^{(i)}(\lambda)$ is formed from the columns of $R_f^{(i)}(\lambda)$ corresponding to nonzero entries in the i-th row of S_{FDI} and $\widetilde{R}_f^{(i)}(\lambda)$ is formed from the columns of $R_f^{(i)}(\lambda)$ corresponding to zero entries in the i-th row of S_{FDI}. If `OPTIONS.FDFreq` is nonempty, then $\big\|\overline{R}_f^{(i)}(\lambda)\big\|_{\infty-}$ is only evaluated over the frequency values contained in `OPTIONS.FDFreq`. The achieved gaps are returned in `INFO.gap`.

Each filter $Q^{(i)}(\lambda)$ is determined in the product form

$$Q^{(i)}(\lambda) = \overline{Q}_1^{(i)}(\lambda)Q_1(\lambda), \tag{174}$$

where the factors are determined as follows:

(a) $Q_1(\lambda) = N_l(\lambda)$, with $N_l(\lambda)$ a $\big(p-r_d\big)\times(p+m_u)$ proper rational left nullspace basis satisfying $N_l(\lambda)\left[\begin{smallmatrix} G_u(\lambda) & G_d(\lambda) \\ I_{m_u} & 0 \end{smallmatrix}\right] = 0$, with $r_d := \operatorname{rank} G_d(\lambda)$;

(b) $\overline{Q}_1^{(i)}(\lambda)$ is the solution of a suitably formulated AFDP to achieve the specification contained in the i-th row of S_{FDI}. The function `afdsyn` is called for this purpose and internally uses the design matrices $H_1^{(i)}$, which can be specified in `OPTIONS.HDesign{`i`}`, and $H_2^{(i)}$, which can be specified in `OPTIONS.HDesign2{`i`}` (see **Method** for `afdsyn`). The actually employed design matrices are returned in `INFO.HDesign{`i`}` and `INFO.HDesign2{`i`}`, respectively.

Each factor $\overline{Q}_1^{(i)}(\lambda)$ is determined in the partitioned form

$$\overline{Q}_1^{(i)}(\lambda) = \left[\begin{array}{c} \overline{Q}_1^{(i,1)}(\lambda) \\ \overline{Q}_1^{(i,2)}(\lambda) \end{array}\right],$$

where the computation of individual factors $\overline{Q}_1^{(i,1)}(\lambda)$ and $\overline{Q}_1^{(i,2)}(\lambda)$ depends on the user's options (see **Method** for the function `afdsyn`). In what follows, we only discuss shortly the computation of $Q_1(\lambda)$, performed only once as an initial reduction step.

If `OPTIONS.nullspace = true` or $m_d > 0$ or E is singular, then $N_l(\lambda)$ is determined as a minimal proper nullspace basis. In this case, only orthogonal transformations are performed at this computational step.

If `OPTIONS.nullspace = false`, $m_d = 0$ and E is invertible, then $N_l(\lambda) = [\, I_p \;\; -G_u(\lambda)\,]$ is used, which corresponds to a full-order Luenberger observer. This option involves no numerical computations.

Example

Example 11. This is *Example* 5.3 from the book [16], with the disturbances redefined as noise inputs and by adding a sensor fault for the first output measurement. The TFMs of the system are:

$$G_u(s) = \begin{bmatrix} \dfrac{s+1}{s+2} \\ \dfrac{s+2}{s+3} \end{bmatrix}, \quad G_d(s) = 0, \quad G_f(s) = [\, G_u(s)\; I\,], \quad G_w(s) = \begin{bmatrix} \dfrac{s-1}{s+2} \\ 0 \end{bmatrix},$$

where the fault input f_1 corresponds to an additive actuator fault, while the fault inputs f_2 and f_3 describe additive sensor faults in the outputs y_1 and y_2, respectively. The transfer function matrix $G_w(s)$ is non-minimum phase, having an unstable zero at 1. Interestingly, the EFDIP formulated with $G_d(s) = G_w(s)$ is not solvable.

The maximal achievable structure matrix (for the EFDIP) is

$$S_{max} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} .$$

We assume that no simultaneous faults occur, and thus we can target to solve an AFDIP for the structure matrix

$$S_{FDI} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} ,$$

where the columns of S_{FDI} codify the desired fault signatures.

We want to design a bank of three detection filters $Q^{(i)}(s)$, $i = 1, 2, 3$ which fulfill for each i:

– the decoupling condition: $Q^{(i)}(s) \left[\begin{smallmatrix} G_u(s) & G_d(s) \\ I_{m_u} & 0 \end{smallmatrix}\right] = 0$;

– the fault isolability condition: $S_{R_f^{(i)}}$ is equal to the i-th row of S_{FDI};

– the maximization of the noise attenuation gap: $\eta_i := \|\overline{R}_f^{(i)}(s)\|_{\infty -}/\|R_w^{(i)}(s)\|_\infty = \max$, where $\overline{R}_f^{(i)}(s)$ is formed from the columns of $R_f^{(i)}(s)$ corresponding to nonzero entries in the i-th row of S_{FDI}.

The results computed with the following script are:

– $\eta_1 = 1.5$ with

$$Q^{(1)}(s) = \begin{bmatrix} \dfrac{s+2}{s+1} & -\dfrac{s+3}{s+2} & 0 \end{bmatrix}, \quad R_f^{(1)}(s) = \begin{bmatrix} 0 & \dfrac{s+2}{s+1} & -\dfrac{s+3}{s+2} \end{bmatrix}, \quad R_w^{(1)}(s) = \frac{s-1}{s+1} \,;$$

– $\eta_2 = \infty$ with

$$Q^{(2)}(s) = \begin{bmatrix} 0 & 1 & -\dfrac{s+2}{s+3} \end{bmatrix}, \quad R_f^{(2)}(s) = \begin{bmatrix} \dfrac{s+2}{s+3} & 0 & 1 \end{bmatrix}, \quad R_w^{(2)}(s) = 0 \,;$$

– $\eta_3 = 1$ with

$$Q^{(3)}(s) = \begin{bmatrix} \dfrac{s+2}{s+1} & 0 & -1 \end{bmatrix}, \quad R_f^{(3)}(s) = \begin{bmatrix} 1 & \dfrac{s+2}{s+1} & 0 \end{bmatrix}, \quad R_w^{(3)}(s) = \frac{s-1}{s+1} \,.$$

```
% Example - Solution of an approximate fault detection problem (AFDIP)

% Example 5.3 of (V,2017) (modified)
s = tf('s'); % define the Laplace variable s
mu = 1; mw = 1; p = 2; mf = mu+p; % set dimensions
Gu = [(s+1)/(s+2); (s+2)/(s+3)];  % enter Gu(s)
Gw = [(s-1)/(s+2); 0];            % enter Gw(s)

% build the model with additive faults having Gf(s) = [Gu(s) eye(p)];
sysf = fdimodset(ss([Gu Gw]),struct('c',1,'f',1,'fs',1:2,'n',2));

% select SFDI
S = fdigenspec(sysf); SFDI = S(sum(S,2)==2,:)
nb = size(SFDI,1);

% perform synthesis with AFDISYN
options = struct('tol',1.e-7,'smarg',-3,...
                 'sdeg',-3,'SFDI',SFDI);
[Q,R,info] = afdisyn(sysf,options);

% display the implementation form Q{i}(s) and the internal forms
% Rf{i}(s) and Rw{i}(s) of the resulting fault detection filters
minreal(tf(Q{1})), minreal(tf(R{1}(:,'faults'))), minreal(tf(R{1}(:,'noise')))
minreal(tf(Q{2})), minreal(tf(R{2}(:,'faults'))), tf(R{2}(:,'noise'))
minreal(tf(Q{3})), minreal(tf(R{3}(:,'faults'))), tf(R{3}(:,'noise'))

% check the resulting gaps
format short e
info.gap
gap = fdif2ngap(R,[],SFDI)

% simulate step responses from fault and noise inputs
inpnames = {'f_1','f_2','f_3','w'};
outnames = {'r_1','r_2','r_3'};
Rtot = vertcat(R{:});
set(Rtot,'InputName',inpnames,'OutputName',outnames);
step(Rtot); ylabel('Residuals')
title('Step responses from fault and noise inputs')
```

4.8.5 `emmsyn`

Syntax

```
[Q,R,INFO] = emmsyn(SYSF,SYSR,OPTIONS)
```

Description

`emmsyn` solves the *exact model matching problem* (EMMP) (as formulated in the more general forms (29) in Remark 7 or (36) in Remark 9; see Section 2.6.5), for a given LTI system `SYSF` with additive faults and a given stable reference filter or a bank of stable reference filters contained in `SYSR`. If `SYSR` is a LTI system, two stable and proper filters, `Q` and `R`, are computed, where `Q` contains the fault detection and isolation filter representing the solution of the EMMP in (29), and `R` contains its internal form. If `SYSR` is a $n_b \times 1$ cell array of LTI systems, then `Q` and `R` are $n_b \times 1$ cell arrays, whose i-th components `Q`$\{i\}$ and `R`$\{i\}$, are, respectively, the implementation form and its internal form of the solution of the corresponding i-th EMMP in (36).

Input data

`SYSF` is a LTI system in the state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_u u(t) + B_d d(t) + B_f f(t) + B_w w(t), \\ y(t) &= Cx(t) + D_u u(t) + D_d d(t) + D_f f(t) + D_w w(t), \end{aligned} \tag{175}$$

where any of the inputs components $u(t)$, $d(t)$, $f(t)$, or $w(t)$ can be void. For the system `SYSF`, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively.

`SYSR` is a stable LTI system or a cell array of stable LTI systems.

If `SYSR` is a LTI system, then it is in the state-space form

$$\begin{aligned} \lambda x_r(t) &= A_r x_r(t) + B_{ru} u(t) + B_{rd} d(t) + B_{rf} f(t) + B_{rw} w(t), \\ y_r(t) &= C_r x_r(t) + D_{ru} u(t) + D_{rd} d(t) + D_{rf} f(t) + D_{rw} w(t), \end{aligned} \tag{176}$$

where the reference model output $y_r(t)$ is a q-dimensional vector and any of the inputs components $u(t)$, $d(t)$, $f(t)$, or $w(t)$ can be void. For the system `SYSR`, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively.

If `SYSR` is a $n_b \times 1$ cell array of LTI systems, then the i-th component proper and stable system `SYSR`$\{i\}$ is in the state-space form

$$\begin{aligned} \lambda x_r^{(i)}(t) &= A_r^{(i)} x_r^{(i)}(t) + B_{ru}^{(i)} u(t) + B_{rd}^{(i)} d(t) + B_{rf}^{(i)} f(t) + B_{rw}^{(i)} w(t), \\ y_r^{(i)}(t) &= C_r^{(i)} x_r^{(i)}(t) + D_{ru}^{(i)} u(t) + D_{rd}^{(i)} d(t) + D_{rf}^{(i)} f(t) + D_{rw}^{(i)} w(t), \end{aligned} \tag{177}$$

where $y_r^{(i)}(t) \in \mathbb{R}^{q^{(i)}}$ is the i-th reference model output and any of the input components $u(t)$, $d(t)$, $f(t)$ and $w(t)$ can be void. For each system `SYSR`$\{i\}$, the input groups for $u(t)$,

$d(t)$, $f(t)$, and $w(t)$ have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively.

`OPTIONS` is a MATLAB structure used to specify various synthesis options and has the following fields:

`OPTIONS` **fields**	**Description**
`tol`	relative tolerance for rank computations (Default: internally computed)
`tolmin`	absolute tolerance for observability tests (Default: internally computed)
`smarg`	stability margin for the poles of the filters `Q` and `R` (Default: `-sqrt(eps)` for a continuous-time system `SYSF`; `1-sqrt(eps)` for a discrete-time system `SYSF`).
`sdeg`	prescribed stability degree for the poles of the filters `Q` and `R` (Default: -0.05 for a continuous-time system `SYSF`; 0.95 for a discrete-time system `SYSF`).
`poles`	complex vector containing a complex conjugate set of desired poles (within the stability domain) to be assigned for the filters `Q` and `R` (Default: `[ ]`)
`simple`	option to employ a simple proper basis for synthesis `true` – use a simple basis; `false` – use a non-simple basis (default)
`minimal`	option to perform least order synthesis of the filter `Q` `true` – perform least order synthesis (default); `false` – perform full order synthesis
`regmin`	option to perform regularization selecting a least order left annihilator `true` – perform least order selection (default); `false` – no least order selection performed
`tcond`	maximum allowed condition number of the employed non-orthogonal transformations (Default: 10^4).
`normalize`	option for the normalization of the diagonal elements of the updating matrix $M(\lambda)$ (if `SYSR` is a LTI system) or of the updating matrices $M^{(i)}(\lambda)$ (if `SYSR` is a cell array of LTI systems): `'gain'` – scale with the gains of the zero-pole-gain representation (default) `'dcgain'` – scale with the DC-gains `'infnorm'` – scale with the values of infinity-norms
`freq`	complex frequency value to be employed to check the left-invertibility-based solvability condition (see **Method**) (Default: `[ ]`, i.e., a randomly generated frequency).

`HDesign`	full row rank design matrix H employed for the synthesis of the filter `Q` (see **Method**) (Default: `[ ]`) If `SYSR` contains n_b filters, then separate design matrices can be specified for each filter, as follows: `OPTIONS.HDesign` is an n_b-dimensional cell array, where each `OPTIONS.HDesign`$\{i\}$ is a full row rank design matrix H_i employed for the synthesis of the i-th filter `Q`$\{i\}$ (Default: `[ ]`) *Note.* This option can be only used in conjunction with the "no least order synthesis" option: `OPTIONS.minimal = false`.

Output data

`Q` is a LTI system or a cell array of LTI filters containing the resulting fault detection filter or the bank of resulting filters, respectively.

If `SYSR` is a LTI system, then `Q` is in a standard state-space representation

$$\begin{aligned} \lambda x_Q(t) &= A_Q x_Q(t) + B_{Q_y} y(t) + B_{Q_u} u(t), \\ r(t) &= C_Q x_Q(t) + D_{Q_y} y(t) + D_{Q_u} u(t), \end{aligned} \tag{178}$$

where the residual signal $r(t)$ is a q-dimensional vector. For the system object `Q`, two input groups `'outputs'` and `'controls'` are defined for $y(t)$ and $u(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r(t)$.

If `SYSR` is a n_b-dimensional cell array of LTI systems, then `Q` is an n_b-dimensional cell array, with `Q`$\{i\}$ containing the resulting i-th filter in a standard state-space representation

$$\begin{aligned} \lambda x_Q^{(i)}(t) &= A_Q^{(i)} x_Q^{(i)}(t) + B_{Q_y}^{(i)} y(t) + B_{Q_u}^{(i)} u(t), \\ r^{(i)}(t) &= C_Q^{(i)} x_Q^{(i)}(t) + D_{Q_y}^{(i)} y(t) + D_{Q_u}^{(i)} u(t), \end{aligned} \tag{179}$$

where the residual signal $r^{(i)}(t)$ is a q_i-dimensional vector. For each system object `Q`$\{i\}$, two input groups `'outputs'` and `'controls'` are defined for $y(t)$ and $u(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r^{(i)}(t)$. `Q`$\{i\}$ is empty if `SYSR`$\{i\}$ is empty.

`R` is a LTI system or a cell array of LTI systems containing the internal form of the resulting fault detection filter or the bank of internal forms of the resulting filters, respectively.

If `SYSR` is a LTI system, then `R` is in a standard state-space representation

$$\begin{aligned} \lambda x_R(t) &= A_R x_R(t) + B_{R_u} u(t) + B_{R_d} d(t) + B_{R_f} f(t) + B_{R_w} w(t), \\ r(t) &= C_R x_R(t) + D_{R_u} u(t) + D_{R_d} d(t) + D_{R_f} f(t) + D_{R_w} w(t) \end{aligned} \tag{180}$$

with the same input groups defined as for `SYSR` and the output group `'residuals'` defined for the residual signal $r(t)$.

If SYSR is a n_b-dimensional cell array of LTI systems, then R an n_b-dimensional cell array, with R$\{i\}$ containing the resulting internal form of the i-th filter in a standard state-space representation

$$\begin{aligned} \lambda x_R^{(i)}(t) &= A_R^{(i)} x_R^{(i)}(t) + B_{R_u}^{(i)} u(t) + B_{R_d}^{(i)} d(t) + B_{R_f}^{(i)} f(t) + B_{R_w}^{(i)} w(t), \\ r^{(i)}(t) &= C_R^{(i)} x_R^{(i)}(t) + D_{R_u}^{(i)} u(t) + D_{R_d}^{(i)} d(t) + D_{R_f}^{(i)} f(t) + D_{R_w}^{(i)} w(t). \end{aligned} \tag{181}$$

The same input groups are defined as for SYSR$\{i\}$ and the output group 'residuals' is defined for the residual signal $r^{(i)}(t)$. R$\{i\}$ is empty if SYSR$\{i\}$ is empty.

INFO is a MATLAB structure containing additional information as follows:

INFO **fields**	**Description**
tcond	the maximum of the condition numbers of the employed non-orthogonal transformation matrices; a warning is issued if INFO.tcond $\geq$ OPTIONS.tcond.
degs	the left Kronecker indices of $G(\lambda) := \left[\begin{smallmatrix} G_u(\lambda) & G_d(\lambda) \\ I & 0 \end{smallmatrix}\right]$ (see **Method**); also the increasingly ordered degrees of a left minimal polynomial nullspace basis of $G(\lambda)$; (INFO.degs = [ ] if no explicit left nullspace basis is computed)
M	If SYSR contains a LTI filter, then INFO.M is the state-space realization of the employed stable, diagonal and invertible updating matrix $M(\lambda)$ (see **Method**). If SYSR contains n_b filters, then INFO.M is an $n_b \times 1$ cell array, with INFO.M$\{i\}$ containing the employed stable, diagonal and invertible updating matrix $M^{(i)}(\lambda)$ (see **Method**).
freq	complex frequency value employed to check the left invertibility condition; INFO.freq = [ ] if no frequency-based left invertibility check was performed.
HDesign	If SYSR contains a LTI filter, then INFO.HDesign contains the design matrix H employed for the synthesis of the fault detection filter Q. If SYSR contains n_b filters, then INFO.HDesign is a an n_b-dimensional cell array, with INFO.HDesign$\{i\}$ containing the design matrix H_i employed for the synthesis of the filter Q$\{i\}$. INFO.HDesign = [ ] if no design matrix was explicitly involved in the filter synthesis.

Method

Two cases are considered. The first case is when SYSR contains a single LTI reference model. The computational approach employed to address this case is discussed in details. The second case is when SYSR contains a collection of reference models. The approach used for a single reference model forms the basis to address the second case, therefore we limit our discussion to the mathematical formulation and associated notations.

Case 1: SYSR contains a single LTI reference model

Extensions of the **Procedure EMM** and **Procedure EMMS** from [16, Sect. 5.6] are implemented in the function `emmsyn`. The **Procedure EMM** relies on the model-matching synthesis method proposed in [6], while **Procedure EMMS** uses the inversion-based method proposed in [14] in conjunction with the nullspace method. The **Procedure EMM** is employed to solve the general EMMP (see below), while **Procedure EMMS** is employed to solve the more particular (but more practice relevant) strong exact fault detection and isolation problem (strong EFDIP).

Assume that the system SYSF has the input-output form

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_f(\lambda)\mathbf{f}(\lambda) + G_w(\lambda)\mathbf{w}(\lambda) \tag{182}$$

and the reference model SYSR has the input-output form

$$\mathbf{y}_r(\lambda) = M_{ru}(\lambda)\mathbf{u}(\lambda) + M_{rd}(\lambda)\mathbf{d}(\lambda) + M_{rf}(\lambda)\mathbf{f}(\lambda) + M_{rw}(\lambda)\mathbf{w}(\lambda), \tag{183}$$

where the vectors y, u, d, f, w and y_r have dimensions p, m_u, m_d, m_f, m_w and q, respectively. The resulting fault detection filter in (178) has the input-output form

$$\mathbf{r}(\lambda) = Q(\lambda)\left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array}\right], \tag{184}$$

where the resulting dimension of the residual vector r is q.

The function `emmsyn` determines $Q(\lambda)$ by solving the general exact model-matching problem

$$Q(\lambda)\left[\begin{array}{c|c|c|c} G_u(\lambda) & G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ I_{m_u} & 0 & 0 & 0 \end{array}\right] = M(\lambda)\left[\, M_{ru}(\lambda) \mid M_{rd}(\lambda) \mid M_{rf}(\lambda) \mid M_{rw}(\lambda) \,\right], \tag{185}$$

where $M(\lambda)$ is a stable, diagonal and invertible transfer function matrix chosen such that the resulting solution $Q(\lambda)$ of the linear rational matrix equation (185) is stable and proper. The resulting internal form $R(\lambda)$, contained in R, is computed as

$$R(\lambda) := \left[\, R_u(\lambda)\,|\,R_d(\lambda)\,|\,R_f(\lambda)\,|\,R_w(\lambda) \,\right] := M(\lambda)\left[\, M_{ru}(\lambda)\,|\,M_{rd}(\lambda)\,|\,M_{rf}(\lambda)\,|\,M_{rw}(\lambda) \,\right]. \tag{186}$$

Two cases are separately addressed, depending on the presence or absence of $M_{rw}(\lambda)$ in the reference model (183).

In the first case, when $M_{rw}(\lambda)$ is present, the general EMMP (185) is solved, with $M_{ru}(\lambda)$, or $M_{rd}(\lambda)$, or both of them, explicitly set to zero if not present in the reference model. In the second case, when $M_{rw}(\lambda)$ is not present, then $Q(\lambda)$ is determined by solving the EMMP

$$Q(\lambda)\left[\begin{array}{c|c|c} G_u(\lambda) & G_d(\lambda) & G_f(\lambda) \\ I_{m_u} & 0 & 0 \end{array}\right] = M(\lambda)\left[\, M_{ru}(\lambda) \mid M_{rd}(\lambda) \mid M_{rf}(\lambda) \,\right] \tag{187}$$

with $M_{ru}(\lambda)$, or $M_{rd}(\lambda)$, or both of them, explicitly set to zero if not present in the reference model. In this case, if $G_w(\lambda)$ is present in the plant model (182), then $R_w(\lambda)$ is explicitly computed as

$$R_w(\lambda) = Q(\lambda)\left[\begin{array}{c} G_w(\lambda) \\ 0 \end{array}\right].$$

The particular EMMP formulated in (27) (see Section 2.6.5) corresponds to solve the EMMP (187) with $M_{ru}(\lambda) = 0$, $M_{rd}(\lambda) = 0$.

If `OPTIONS.minimal = true`, then a least order solution $Q(\lambda)$ is determined in the form $Q(\lambda) = Q_2(\lambda)Q_1(\lambda)$, where $Q_1(\lambda)$ is a least McMillan degree solution of the linear rational matrix equation

$$Q_1(\lambda)\left[\begin{array}{c|c|c|c} G_u(\lambda) & G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ I_{m_u} & 0 & 0 & 0 \end{array}\right] = \left[\, M_{ru}(\lambda) \mid M_{rd}(\lambda) \mid M_{rf}(\lambda) \mid M_{rw}(\lambda) \,\right] \qquad (188)$$

and the diagonal updating factor $Q_2(\lambda) := M(\lambda)$ is determined to ensure that $Q(\lambda)$ is proper and stable.

If `OPTIONS.minimal = false` and either $M_{ru}(\lambda)$ or $M_{rd}(\lambda)$, or both, are present in the reference model (183), then $Q(\lambda)$ is determined in the form $Q(\lambda) = Q_2(\lambda)Q_1(\lambda)$, where $Q_1(\lambda)$ is a particular solution of the linear rational equation (188) and the diagonal updating factor $Q_2(\lambda) := M(\lambda)$ is determined to ensure that $Q(\lambda)$ is proper and stable.

If `OPTIONS.minimal = false` and if both $M_{ru}(\lambda)$ and $M_{rd}(\lambda)$ are absent in the reference model (183), then the nullspace method is employed as the first computational step of solving the EMMP (see **Procedure EMM** in [16]). The strong EFDIP arises if additionally $M_{rf}(\lambda)$ is diagonal and invertible, in which case, an extension of the **Procedure EMMS** in [16] is employed. In fact, this procedure works for arbitrary invertible $M_{rf}(\lambda)$ and this case was considered for the implementation of `emmsyn`. The solution of a fault estimation problem can be targeted by choosing $M_{rf}(\lambda) = I_{m_f}$ and checking that the resulting $M(\lambda) = I_{m_f}$. Recall that $M(\lambda)$ is provided in `INFO.M`. In what follows, we give some details of the implemented synthesis approach employed if $M_{ru}(\lambda)$, $M_{rd}(\lambda)$ and $M_{rw}(\lambda)$ are not present in reference model.

If `OPTIONS.regmin = false`, then $Q(\lambda)$ is determined in the form

$$Q(\lambda) = M(\lambda)Q_2(\lambda)HN_l(\lambda),$$

where: $N_l(\lambda)$ is a $\big(p - r_d\big) \times (p + m_u)$ rational left nullspace basis satisfying

$$N_l(\lambda)\left[\begin{array}{cc} G_u(\lambda) & G_d(\lambda) \\ I_{m_u} & 0 \end{array}\right] = 0,$$

with $r_d := \operatorname{rank} G_d(\lambda)$; H is a suitable full row rank design matrix used to build q linear combinations of the $p - r_d$ left nullspace basis vectors (q is the number of outputs of `SYSR`); $Q_2(\lambda)$ is the solution of $Q_2(\lambda)H\overline{G}_f(\lambda) = M_{rf}(\lambda)$, where

$$\overline{G}_f(\lambda) = N_l(\lambda)\left[\begin{array}{c} G_f(\lambda) \\ 0 \end{array}\right];$$

and $M(\lambda)$ is a stable invertible transfer function matrix determined such that $Q(\lambda)$ and the corresponding $R(\lambda)$ in (186) have desired dynamics (specified via `OPTIONS.sdeg` and `OPTIONS.poles`). The internal form of the filter $Q(\lambda)$ is obtained as

$$R(\lambda) = M(\lambda)Q_2(\lambda)H\overline{G}(\lambda),$$

where

$$\overline{G}(\lambda) := \big[\, \overline{G}_f(\lambda) \mid \overline{G}_w(\lambda) \,\big] = N_l(\lambda) \left[\begin{array}{c|c} G_f(\lambda) & G_w(\lambda) \\ 0 & 0 \end{array} \right].$$

The solvability condition of the strong EFDIP is verified by checking the left invertibility condition

$$\operatorname{rank} H\overline{G}_f(\lambda_s) = m_f, \tag{189}$$

where λ_s is a suitable frequency value, which can be specified via the `OPTIONS.freq`. The design parameter matrix H is set as follows: if `OPTIONS.HDesign` is nonempty, then H = `OPTIONS.HDesign`; if `OPTIONS.HDesign = [ ]`, then $H = I_{p-r_d}$, if $q = p - r_d$, or H is a randomly generated $q \times \big(p - r_d\big)$ real matrix, if $q < p - r_d$. If `OPTIONS.simple = true`, then $N_l(\lambda)$ is determined as a simple rational basis. The orders of the basis vectors are provided in `INFO.degs`. These are also the degrees of the basis vectors of an equivalent polynomial nullspace basis.

If `OPTIONS.regmin = true`, then $[\, Q(\lambda) \;\, R_f(\lambda) \,]$ has the least McMillan degree, with $Q(\lambda)$ having q outputs. $Q(\lambda)$ and $R_f(\lambda)$ are determined in the form

$$[\, Q(\lambda) \mid R_f(\lambda) \,] = M(\lambda)Q_2(\lambda)\big[\, \overline{Q}(\lambda) \mid \overline{R}_f(\lambda) \,\big],$$

where

$$\big[\, \overline{Q}(\lambda) \mid \overline{R}_f(\lambda) \,\big] = H\big[\, N_l(\lambda) \mid \overline{G}_f(\lambda) \,\big] + Y(\lambda)\big[\, N_l(\lambda) \mid \overline{G}_f(\lambda) \,\big]$$

with $[\, \overline{Q}(\lambda) \;\, \overline{R}_f(\lambda) \,]$ and $Y(\lambda)$ the least order solution of a left minimal cover problem [15]; $Q_2(\lambda)$ is the solution of $Q_2(\lambda)\overline{R}_f(\lambda) = M_{rf}(\lambda)$; and $M(\lambda)$ is a stable invertible transfer function matrix determined such that $[\, Q(\lambda) \;\, R(\lambda) \,]$ has a desired dynamics. If `OPTIONS.HDesign` is nonempty, then H = `OPTIONS.HDesign`, and if `OPTIONS.HDesign = [ ]`, then a suitable randomly generated H is employed, which fulfills the left invertibility condition (189).

The actually employed design matrix H is provided in `INFO.HDesign`, and `INFO.HDesign = [ ]` if the solution of the EMMP is obtained by directly solving (185).

Case 2: `SYSR` contains a collection of reference models

If `SYSR` contains n_b reference models with the i-th model having the input-output form

$$\mathbf{y}_r^{(i)}(\lambda) = M_{ru}^{(i)}(\lambda)\mathbf{u}(\lambda) + M_{rd}^{(i)}(\lambda)\mathbf{d}(\lambda) + M_{rf}^{(i)}(\lambda)\mathbf{f}(\lambda) + M_{rw}^{(i)}(\lambda)\mathbf{w}(\lambda), \tag{190}$$

where $y_r^{(i)}$ has dimension $q^{(i)}$, then the resulting `Q` contains n_b filters, with the i-th filter in (179) having the input-output form

$$\mathbf{r}^{(i)}(\lambda) = Q^{(i)}(\lambda) \begin{bmatrix} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{bmatrix}, \tag{191}$$

where the resulting dimension of the residual vector $r^{(i)}$ is $q^{(i)}$. $Q^{(i)}(\lambda)$ is determined by solving the general exact model-matching problem

$$Q^{(i)}(\lambda)\left[\begin{array}{c|c|c|c} G_u(\lambda) & G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ I_{m_u} & 0 & 0 & 0 \end{array}\right] = M^{(i)}(\lambda)\big[\, M_{ru}^{(i)}(\lambda) \mid M_{rd}^{(i)}(\lambda) \mid M_{rf}^{(i)}(\lambda) \mid M_{rw}^{(i)}(\lambda)\,\big], \quad (192)$$

where $M^{(i)}(\lambda)$ is a stable, diagonal and invertible transfer function matrix chosen such that the resulting solution $Q^{(i)}(\lambda)$ of the linear rational matrix equation (192) is stable and proper. The resulting internal form of the i-th filter $R^{(i)}(\lambda)$ in (181), contained in the i-th cell of `R`, is computed as

$$R^{(i)}(\lambda) := \big[\, R_u^{(i)}(\lambda) \,|\, R_d^{(i)}(\lambda) \,|\, R_f^{(i)}(\lambda) \,|\, R_w^{(i)}(\lambda)\,\big] := M^{(i)}(\lambda)\big[\, M_{ru}^{(i)}(\lambda) \,|\, M_{rd}^{(i)}(\lambda) \,|\, M_{rf}^{(i)}(\lambda) \,|\, M_{rw}^{(i)}(\lambda)\,\big]. \quad (193)$$

For the computation of $Q^{(i)}(\lambda)$ and $R^{(i)}(\lambda)$, for $i = 1,\ldots,n_b$, the previously sketched computational approaches are employed for all filters, taking into account the selected synthesis options in the `OPTION` structure. The resulting `INFO.M` and `OPTIONS.HDesign` are $n_b \times 1$ cell arrays, and `INFO.M`$\{i\}$ the state-space realization of $M^{(i)}(\lambda)$, while `INFO.HDesign`$\{i\}$ contains the design matrix H_i employed for the synthesis of the i-th filter. `INFO.HDesign = [ ]` if no design matrices have been involved.

Examples

Example 12. This is *Example* 5.12 from the book [16] and was used in *Example* 9, to solve an EFDIP for a system with triplex sensor redundancy. To solve the same problem by solving an EMMP, we use the resulting $R_f(s)$ to define the reference model

$$M_{rf}(s) := R_f(s) = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}.$$

The resulting least order filter $Q(s)$, determined by employing `emmsyn` with **Procedure EMM**, has the generic form

$$Q(s) = \begin{bmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & -1 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

```
% Example - Solution of an EMMP

p = 3; mf = 3;    % enter output and fault vector dimensions
% generate random dimensions for system order and input vectors
rng('default')
nu = floor(1+4*rand); mu = floor(1+4*rand);
nd = floor(1+4*rand); md = floor(1+4*rand);
% define random Gu(s) and Gd(s) with triplex sensor redundancy
% and Gf(s) = I for triplex sensor faults
Gu = ones(3,1)*rss(nu,1,mu); % enter Gu(s) in state-space form
```

```
Gd = ones(3,1)*rss(nd,1,md); % enter Gd(s) in state-space form

% build synthesis model with sensor faults
sysf = fdimodset([Gu Gd],struct('c',1:mu,'d',mu+(1:md),'fs',1:3));

% enter reference model for the TFM from faults to residual
Mr = fdimodset(ss([ 0 1 -1; -1 0 1; 1 -1 0]),struct('f',1:mf));

% solve an exact model-matching problem using EMMSYN
[Q,R,info] = emmsyn(sysf,Mr);

% determine achieved fault sensitivity conditions
FSCOND = fdifscond(R,0,fdisspec(Mr))

% check the synthesis: Q*Ge = M*Me and R = M*Mr, where
% Ge = [Gu Gd Gf; I 0 0] and Me = [0 0 Mr ].
Ge = [sysf; eye(mu,mu+md+mf)]; Me = [zeros(p,mu+md) Mr];
norm(gminreal(Q*Ge)-info.M*Me,inf)
norm(R-info.M*Mr,inf)
```

Example 13. This is *Example* 5.13 from the book [16], with a continuous-time system with additive actuator faults, having the transfer-function matrices

$$G_u(s) = \begin{bmatrix} \dfrac{s}{s^2+3s+2} & \dfrac{1}{s+2} \\ \dfrac{s}{s+1} & 0 \\ 0 & \dfrac{1}{s+2} \end{bmatrix}, \quad G_d(s) = 0, \quad G_f(s) = G_u(s), \quad G_w(s) = 0\,.$$

We want to solve an EMMP with the reference model

$$M_{rf}(s) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

which is equivalent to solve a strong EFDIP with the structure matrix

$$S_{FDI} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

A least order stable filter $Q(s)$, with poles assigned to -1, has been determined by employing **emmsyn** with **Procedure EMMS** of [16] (which is employed if `OPTIONS.minimal = false`). The resulting $Q(s)$ is

$$Q(s) = \begin{bmatrix} 0 & 1 & 0 & -\dfrac{s}{s+1} & 0 \\ 0 & 0 & \dfrac{s+2}{s+1} & 0 & -\dfrac{1}{s+1} \end{bmatrix}$$

and has the McMillan degree equal to 2. The resulting updating factor is

$$M(s) = \left[\begin{array}{cc} \dfrac{s}{s+1} & 0 \\ 0 & \dfrac{1}{s+1} \end{array} \right]$$

and has McMillan degree 2. The presence of the zero at $s = 0$ in $R(s) := M(s)M_r(s)$ is unavoidable for the existence of a stable solution. It follows, that while a constant actuator fault f_2 is strongly detectable, a constant actuator fault f_1 is only detectable during transients.

```
% Example - Solution of a strong EFDIP as an EMMP

% define s as an improper transfer function
s = tf('s');
% enter Gu(s)
Gu = [s/(s^2+3*s+2) 1/(s+2);
     s/(s+1) 0;
      0 1/(s+2)];
[p,mu] = size(Gu); mf = mu;

% build model with faults
sysf = fdimodset(ss(Gu),struct('c',1:mu,'f',1:mu));

% define Mr(s)
Mr = fdimodset(ss(eye(2)),struct('f',1:mf));

% solve a strong EFDIP using EMMSYN (for an invertible reference model)
opts_emmsyn = struct('tol',1.e-7,'sdeg',-1,'minimal',false);
[Q,R,info] = emmsyn(sysf,Mr,opts_emmsyn);

% check solution
G = [sysf;eye(mu,mu+mf)];
norm(Q*G-info.M*[zeros(mf,mu) Mr],inf)

% display results
minreal(tf(Q)), tf(info.M)
```

4.8.6 ammsyn

Syntax

```
[Q,R,INFO] = ammsyn(SYSF,SYSR,OPTIONS)
```

Description

ammsyn solves the *approximate model matching problem* (AMMP) (as formulated in the more general forms (39) in Remark 10 or (41) in Remark 11; see Section 2.6.6), for a given LTI system

SYSF with additive faults and a given stable reference filter or a bank of stable reference filters contained in SYSR. If SYSR is a LTI system, two stable and proper filters, Q and R, are computed, where Q contains the fault detection and isolation filter representing the solution of the AMMP in (39), and R contains its internal form. If SYSR is a $n_b \times 1$ cell array of LTI systems, then Q and R are $n_b \times 1$ cell arrays, whose i-th components Q$\{i\}$ and R$\{i\}$, are, respectively, the implementation form and its internal form of the solution of the corresponding i-th AMMP in (41).

Input data

SYSF is a LTI system in the state-space form

$$\begin{aligned} E\lambda x(t) &= Ax(t) + B_u u(t) + B_d d(t) + B_f f(t) + B_w w(t), \\ y(t) &= Cx(t) + D_u u(t) + D_d d(t) + D_f f(t) + D_w w(t), \end{aligned} \tag{194}$$

where any of the inputs components $u(t)$, $d(t)$, $f(t)$ or $w(t)$ can be void. For the system SYSF, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ must have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively.

SYSR is a stable LTI system or a cell array of stable LTI systems.

If SYSR is a LTI system, then it is in the state-space form

$$\begin{aligned} \lambda x_r(t) &= A_r x_r(t) + B_{ru} u(t) + B_{rd} d(t) + B_{rf} f(t) + B_{rw} w(t), \\ y_r(t) &= C_r x_r(t) + D_{ru} u(t) + D_{rd} d(t) + D_{rf} f(t) + D_{rw} w(t), \end{aligned} \tag{195}$$

where the reference model output $y_r(t)$ is a q-dimensional vector and any of the inputs components $u(t)$, $d(t)$, $f(t)$, or $w(t)$ can be void. For the system SYSR, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively.

If SYSR is a $n_b \times 1$ cell array of LTI systems, then the i-th component proper and stable system SYSR$\{i\}$ is in the state-space form

$$\begin{aligned} \lambda x_r^{(i)}(t) &= A_r^{(i)} x_r^{(i)}(t) + B_{ru}^{(i)} u(t) + B_{rd}^{(i)} d(t) + B_{rf}^{(i)} f(t) + B_{rw}^{(i)} w(t), \\ y_r^{(i)}(t) &= C_r^{(i)} x_r^{(i)}(t) + D_{ru}^{(i)} u(t) + D_{rd}^{(i)} d(t) + D_{rf}^{(i)} f(t) + D_{rw}^{(i)} w(t), \end{aligned} \tag{196}$$

where $y_r^{(i)}(t) \in \mathbb{R}^{q^{(i)}}$ is the i-th reference model output and any of the input components $u(t)$, $d(t)$, $f(t)$ and $w(t)$ can be void. For each system SYSR$\{i\}$, the input groups for $u(t)$, $d(t)$, $f(t)$, and $w(t)$ have the standard names `'controls'`, `'disturbances'`, `'faults'`, and `'noise'`, respectively.

OPTIONS is a MATLAB structure used to specify various synthesis options and has the following fields:

`OPTIONS` **fields**	**Description**
`tol`	relative tolerance for rank computations (Default: internally computed)
`tolmin`	absolute tolerance for observability tests (Default: internally computed)
`reltol`	relative tolerance for the desired accuracy of gamma-iteration (Default: 0.0001)
`smarg`	stability margin for the poles of the updating factor $M(\lambda)$ (if `SYSR` is a LTI system) or of the updating factors $M^{(i)}(\lambda)$ (if `SYSR` is a cell array of LTI systems) (see **Method**) (Default: `-sqrt(eps)` for a continuous-time system `SYSF`; `1-sqrt(eps)` for a discrete-time system `SYSF`).
`sdeg`	prescribed stability degree for the poles of the updating factor $M(\lambda)$ (if `SYSR` is a LTI system) or of the updating factors $M^{(i)}(\lambda)$ (if `SYSR` is a cell array of LTI systems) (see **Method**) (Default: -0.05 for a continuous-time system `SYSF`; 0.95 for a discrete-time system `SYSF`).
`poles`	complex vector containing a complex conjugate set of desired poles (within the stability domain) to be assigned for the updating factor $M(\lambda)$ (if `SYSR` is a LTI system) or for the updating factors $M^{(i)}(\lambda)$ (if `SYSR` is a cell array of LTI systems) (see **Method**) (Default: `[ ]`)
`nullspace`	option to use a specific proper nullspace basis to be employed in the nullspace-based synthesis step `true` – use a minimal proper basis (default); `false` – use a full-order observer based basis (see **Method**). *Note:* This option can only be used if no disturbance inputs are present in (194) and E is invertible.
`simple`	option to employ a simple proper basis for the nullspace-based synthesis `true` – use a simple basis; `false` – use a non-simple basis (default)
`mindeg`	option to compute a minimum degree solution `true` – determine, if possible, a minimum order stable solution `false` – determine a particular stable solution which has possibly non-minimal (default).
`regmin`	regularization option with least order left annihilator selection `true` – perform least order selection (default); `false` – no least order selection to be performed
`tcond`	maximum allowed condition number of the employed non-orthogonal transformations (Default: 10^4).

`normalize`	option for the normalization of the diagonal elements of the updating matrix $M(\lambda)$ (if `SYSR` is a LTI system) or of the updating factors $M^{(i)}(\lambda)$ (if `SYSR` is a cell array of LTI systems) (see **Method**): `'gain'` – scale with the gains of the zero-pole-gain representation (default) `'dcgain'` – scale with the DC-gains `'infnorm'` – scale with the values of infinity-norms
`freq`	complex frequency value to be employed to check frequency response based (admissibility) rank conditions (see **Method**) (Default: `[ ]`, i.e., a randomly generated frequency).
`HDesign`	If `SYSR` contains a LTI filter, then `OPTIONS.HDesign` is a full row rank design matrix H employed for the synthesis of the filter `Q` (see **Method**) (Default: `[ ]`) If `SYSR` contains n_b filters, then separate design matrices can be specified for each filter, as follows: `OPTIONS.HDesign` is an n_b-dimensional cell array, where each `OPTIONS.HDesign`$\{i\}$ is a full row rank design matrix H_i employed for the synthesis of the i-th filter `Q`$\{i\}$ (Default: `[ ]`)
`H2syn`	option to perform a $\mathcal{H}_2$-norm based synthesis `true` – perform a $\mathcal{H}_2$-norm based synthesis; `false` – perform a $\mathcal{H}_\infty$-norm based synthesis (default).

Output data

`Q` is a LTI system or a cell array of LTI systems containing the resulting fault detection filter or the bank of resulting filters, respectively.

If `SYSR` is a LTI system, then `Q` is in a standard state-space representation

$$\begin{aligned} \lambda x_Q(t) &= A_Q x_Q(t) + B_{Q_y} y(t) + B_{Q_u} u(t), \\ r(t) &= C_Q x_Q(t) + D_{Q_y} y(t) + D_{Q_u} u(t), \end{aligned} \tag{197}$$

where the residual signal $r(t)$ is a q-dimensional vector. For the system object `Q`, two input groups `'outputs'` and `'controls'` are defined for $y(t)$ and $u(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r(t)$.

If `SYSR` is a n_b-dimensional cell array of LTI systems, then `Q` is an n_b-dimensional cell array, with `Q`$\{i\}$ containing the resulting i-th filter in a standard state-space representation

$$\begin{aligned} \lambda x_Q^{(i)}(t) &= A_Q^{(i)} x_Q^{(i)}(t) + B_{Q_y}^{(i)} y(t) + B_{Q_u}^{(i)} u(t), \\ r^{(i)}(t) &= C_Q^{(i)} x_Q^{(i)}(t) + D_{Q_y}^{(i)} y(t) + D_{Q_u}^{(i)} u(t), \end{aligned} \tag{198}$$

where the residual signal $r^{(i)}(t)$ is a q_i-dimensional vector. For each system object `Q`$\{i\}$, two input groups `'outputs'` and `'controls'` are defined for $y(t)$ and $u(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r^{(i)}(t)$. `Q`$\{i\}$ is empty if `SYSR`$\{i\}$ is empty.

`R` is a LTI system or a cell array of LTI systems containing the internal form of the resulting fault detection filter or the bank of internal forms of the resulting filters, respectively.

If `SYSR` is a LTI system, then `R` is in a standard state-space representation

$$\begin{aligned} \lambda x_R(t) &= A_R x_R(t) + B_{R_u} u(t) + B_{R_d} d(t) + B_{R_f} f(t) + B_{R_w} w(t), \\ r(t) &= C_R x_R(t) + D_{R_u} u(t) + D_{R_d} d(t) + D_{R_f} f(t) + D_{R_w} w(t) \end{aligned} \tag{199}$$

with the same input groups defined as for `SYSR` and the output group `'residuals'` defined for the residual signal $r(t)$.

If `SYSR` is a n_b-dimensional cell array of LTI systems, then `R` an n_b-dimensional cell array, with `R`$\{i\}$ containing the resulting internal form of the i-th filter in a standard state-space representation

$$\begin{aligned} \lambda x_R^{(i)}(t) &= A_R^{(i)} x_R^{(i)}(t) + B_{R_u}^{(i)} u(t) + B_{R_d}^{(i)} d(t) + B_{R_f}^{(i)} f(t) + B_{R_w}^{(i)} w(t), \\ r^{(i)}(t) &= C_R^{(i)} x_R^{(i)}(t) + D_{R_u}^{(i)} u(t) + D_{R_d}^{(i)} d(t) + D_{R_f}^{(i)} f(t) + D_{R_w}^{(i)} w(t). \end{aligned} \tag{200}$$

The same input groups are defined as for `SYSR`$\{i\}$ and the output group `'residuals'` is defined for the residual signal $r^{(i)}(t)$. `R`$\{i\}$ is empty if `SYSR`$\{i\}$ is empty.

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

`INFO` **fields**	**Description**
`tcond`	the maximum of the condition numbers of the employed non-orthogonal transformation matrices; a warning is issued if `INFO.tcond` $\geq$ `OPTIONS.tcond`.
`degs`	the left Kronecker indices of $G(\lambda) := \left[\begin{smallmatrix} G_u(\lambda) & G_d(\lambda) \\ I & 0 \end{smallmatrix}\right]$ (see **Method**); also, if `OPTIONS.simple` = `true`, the orders of the basis vectors of the employed simple nullspace basis, or if `OPTIONS.simple` = `false`, the degrees of the basis vectors of an equivalent polynomial nullspace basis. `INFO.degs = [ ]` if no strong FDI or fault detection oriented synthesis is performed.
`M`	If `SYSR` contains a LTI filter, then `INFO.M` is the state-space realization of the employed stable, diagonal and invertible updating matrix $M(\lambda)$ (see **Method**). If `SYSR` contains n_b filters, then `INFO.M` is an $n_b \times 1$ cell array, with `INFO.M`$\{i\}$ containing the employed stable, diagonal and invertible updating matrix $M^{(i)}(\lambda)$ (see **Method**).

`freq`	complex frequency value employed to check frequency response based (admissibility) rank conditions
`HDesign`	If `SYSR` contains a LTI filter, then `INFO.HDesign` contains the design matrix H employed for the synthesis of the fault detection filter `Q`. If `SYSR` contains n_b filters, then `INFO.HDesign` is a an n_b-dimensional cell array, with `INFO.HDesign`$\{i\}$ containing the design matrix H_i employed for the synthesis of the filter `Q`$\{i\}$. `INFO.HDesign = [ ]` if no design matrix was explicitly involved in the filter synthesis.
`nonstandard`	If `SYSR` contains a LTI filter, then `INFO.nonstandard` is a logical value, which is set `true`, for a *non-standard problem* or `false`, for a *standard problem* (see **Method**). If `SYSR` contains n_b filters, then `INFO.nonstandard` is a an n_b-dimensional logical vector, with `INFO.nonstandard`(i) set `true`, for a non-standard i-th problem, and `false`, for a standard i-th problem (see **Method**).
`gammaopt0`	If `SYSR` contains a LTI filter, then `INFO.gammaopt0` is the resulting optimal model-matching performance value $\gamma_{opt,0}$ in (207) (see **Method**). If `SYSR` contains n_b filters, then `INFO.gammaopt0` is an n_b-dimensional vector, with `INFO.gammaopt0`(i) containing the optimal performance value $\gamma_{opt,0}^{(i)}$ in (223).
`gammaopt`	If `SYSR` contains a LTI filter, then `INFO.gammaopt` is the resulting optimal model-matching performance value $\gamma_{opt,0}$ in (207), in the *standard case*, and γ_{opt} in (208) in the *non-standard case* (see **Method**). If `SYSR` contains n_b filters, then `INFO.gammaopt` is an n_b-dimensional vector, with `INFO.gammaopt`(i) containing the optimal performance value $\gamma_{opt,0}^{(i)}$ in (223), in the *standard case*, and $\gamma_{opt}^{(i)}$ in (224) in the *non-standard case* (see **Method**).
`gammasub`	If `SYSR` contains a LTI filter, then `INFO.gammasub` is the resulting suboptimal model-matching performance value γ_{sub} in (209) (see **Method**). If `SYSR` contains n_b filters, then `INFO.gammasub` is an n_b-dimensional vector, with `INFO.gammasub`(i) containing the suboptimal performance value $\gamma_{sub}^{(i)}$ in (225) (see **Method**).

Method

Two cases are considered. The first case is when `SYSR` contains a single LTI reference model. The computational approach employed to address this case is discussed in details. The second case is when `SYSR` contains a collection of reference models. The approach used for a single reference model forms the basis to address the second case, therefore we limit our discussion to the mathematical formulation and associated notations.

Case 1: `SYSR` contains a single LTI reference model

The function `ammsyn` implements the **Procedure AMMS** from [16, Sect. 5.7], which relies on the approximate model-matching synthesis methods proposed in [12] (see also [13] for more computational details). This procedure is primarily intended to approximately solve the particular (but more practice relevant) strong fault detection and isolation problem, which is addressed in the standard formulation of the AMMP in Section 2.6.6. However, the function `ammsyn` is also able to address the more general case of AMMP with an arbitrary stable reference model (see Remark 10 in Section 2.6.6). Therefore, in what follows, we consider the solution of the AMMP in this more general problem setting.

Assume that the system `SYSF` has the input-output form

$$\mathbf{y}(\lambda) = G_u(\lambda)\mathbf{u}(\lambda) + G_d(\lambda)\mathbf{d}(\lambda) + G_f(\lambda)\mathbf{f}(\lambda) + G_w(\lambda)\mathbf{w}(\lambda) \tag{201}$$

and the reference model `SYSR` has the input-output form

$$\mathbf{y}_r(\lambda) = M_{ru}(\lambda)\mathbf{u}(\lambda) + M_{rd}(\lambda)\mathbf{d}(\lambda) + M_{rf}(\lambda)\mathbf{f}(\lambda) + M_{rw}(\lambda)\mathbf{w}(\lambda), \tag{202}$$

where the vectors y, u, d, f, w and y_r have dimensions p, m_u, m_d, m_f, m_w and q, respectively. Furthermore, we assume that $M_r(\lambda) := \big[\, M_{ru}(\lambda)\; M_{rd}(\lambda)\; M_{rf}(\lambda)\; M_{rw}(\lambda) \,\big]$ is stable.

The resulting fault detection filter in (197) has the input-output form

$$\mathbf{r}(\lambda) = Q(\lambda)\left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array}\right], \tag{203}$$

where the resulting dimension of the residual vector r is q.

The function `ammsyn` determines $Q(\lambda)$ by solving the approximate model-matching problem

$$Q(\lambda)G_e(\lambda) \approx M(\lambda)M_r(\lambda)\,, \tag{204}$$

where

$$G_e(\lambda) := \left[\begin{array}{c|c|c|c} G_u(\lambda) & G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ I_{m_u} & 0 & 0 & 0 \end{array}\right], \tag{205}$$

$M(\lambda)$ is a stable, diagonal and invertible transfer function matrix chosen such that the resulting approximate solution $Q(\lambda)$ of (204) is stable and proper. The resulting internal form $R(\lambda)$, contained in `R`, is computed as

$$R(\lambda) := \big[\, R_u(\lambda) \,|\, R_d(\lambda) \,|\, R_f(\lambda) \,|\, R_w(\lambda) \,\big] := Q(\lambda)G_e(\lambda). \tag{206}$$

In the *standard case*, $G_e(\lambda)$ has no zeros on the boundary of the stability domain $\partial\mathbb{C}_s$, and the resulting stable filter $Q(\lambda) = Q_{opt}(\lambda)$, is the optimal solution of the $\mathcal{H}_\infty$- or $\mathcal{H}_2$-norm error minimization problem

$$\gamma_{opt,0} := \|Q_{opt}(\lambda)G_e(\lambda) - M_0(\lambda)M_r(\lambda)\|_{\infty/2} = \min, \tag{207}$$

where $M_0(\lambda) = I$ in the case of $\mathcal{H}_\infty$-norm or of a discrete-time system. In the case of $\mathcal{H}_2$-norm and a continuous-time system, $M_0(s)$ is determined as a stable, diagonal, and invertible transfer function matrix, which ensures the existence of a finite $\mathcal{H}_2$-norm.

In the *non-standard case*, $G_e(\lambda)$ has zeros on the boundary of the stability domain $\partial\mathbb{C}_s$, and the optimal solution $Q_{opt}(\lambda)$ of (207) is possibly unstable or improper. In this case, $Q(\lambda)$ is chosen as $Q(\lambda) = M_1(\lambda)Q_{opt}(\lambda)$, with $M_1(\lambda)$ a stable, diagonal, and invertible transfer function matrix determined to ensure the stability of $Q(\lambda)$. In this case, $Q(\lambda)$ can be interpreted as a suboptimal solution of the updated $\mathcal{H}_\infty$- or $\mathcal{H}_2$-norm error minimization problem

$$\gamma_{opt} := \|\widetilde{Q}_{opt}(\lambda)G_e(\lambda) - M_1(\lambda)M_0(\lambda)M_r(\lambda)\|_{\infty/2} = \min, \tag{208}$$

whose optimal solution $\widetilde{Q}_{opt}(\lambda)$ is possibly unstable or improper, but for which $Q(\lambda)$ represents a stable and proper suboptimal solution, with the corresponding suboptimal model-matching error norm

$$\gamma_{sub} := \|Q(\lambda)G_e(\lambda) - M_1(\lambda)M_0(\lambda)M_r(\lambda)\|_{\infty/2}. \tag{209}$$

The value of $\gamma_{opt,0}$ in (207) is returned in `INFO.gammaopt0`, and also in `INFO.gammaopt` and `INFO.gammasub` in the *standard case*. In the *non-standard case*, the value γ_{opt} in (208) is returned in `INFO.gammaopt` and the value of γ_{sub} in (209) is returned in `INFO.gammasub`. The updating matrix $M(\lambda) := M_0(\lambda)$, in the *standard case*, or $M(\lambda) := M_1(\lambda)M_0(\lambda)$, in the *non-standard case*, is returned in `INFO.M`.

Two cases are separately addressed in what follows, depending on the presence or absence of the terms $M_{ru}(\lambda)$, $M_{rd}(\lambda)$ and $M_{rw}(\lambda)$ in the reference model (183).

(a) *The case with* $\big[\, M_{ru}(\lambda)\ M_{rd}(\lambda)\ M_{rw}(\lambda) \,\big] = 0$.

This case corresponds to the standard formulation of the AMMP of Section 2.6.6 and we assume that $M_{rf}(\lambda)$ has all its columns nonzero (to enforce complete fault detectability). If $M_{rf}(\lambda)$ is invertible, we target the solution of a *strong fault detection and isolation* (*strong* FDI) problem. Additionally, we assume that $q \leq p - r_d$, where $r_d := \operatorname{rank} G_d(\lambda)$. The second case, covers the rest of situations and is discussed separately.

To compute the solution $Q(\lambda)$ for the standard formulation of the AMMP, we employ a slight extension of the synthesis method which underlies **Procedure AMMD** in [16]. This procedure essentially determines the filter $Q(\lambda)$ as a stable rational left annihilator of

$$G(\lambda) := \left[\begin{array}{cc} G_u(\lambda) & G_d(\lambda) \\ I_{m_u} & 0 \end{array}\right], \tag{210}$$

such that $Q(\lambda)$ also simultaneously solves the approximate model-matching problem

$$Q(\lambda)\left[\begin{array}{c|c} G_f(\lambda) & G_w(\lambda) \\ 0 & 0 \end{array}\right] \approx M(\lambda)\big[\, M_{rf}(\lambda) \mid 0 \,\big], \tag{211}$$

where $M(\lambda)$ is a stable, diagonal and invertible transfer function matrix chosen such that the resulting solution $Q(\lambda)$ is stable and proper. The resulting internal form $R(\lambda)$, contained in `R`, is computed as

$$R(\lambda) = [\, R_f(\lambda) \mid R_w(\lambda) \,] := Q(\lambda)\left[\begin{array}{c|c} G_f(\lambda) & G_w(\lambda) \\ 0 & 0 \end{array}\right]. \tag{212}$$

If `OPTIONS.H2syn = false`, then a $\mathcal{H}_\infty$-norm based synthesis is performed by determining $Q(\lambda)$ and $M(\lambda)$ such that $\|\mathcal{E}(\lambda)\|_\infty$ is minimized, where

$$\mathcal{E}(\lambda) = [\, R_f(\lambda) \mid R_w(\lambda)\,] - M(\lambda)[\, M_{rf}(\lambda) \mid 0\,]. \tag{213}$$

If `OPTIONS.H2syn = true`, then a $\mathcal{H}_2$-norm based synthesis is performed by determining $Q(\lambda)$ and $M(\lambda)$ such that $\|\mathcal{E}(\lambda)\|_2$ is minimized.

The filter $Q(\lambda)$ is determined in the product form

$$Q(\lambda) = Q_5(\lambda)Q_4(\lambda)Q_3(\lambda)Q_2(\lambda)Q_1(\lambda), \tag{214}$$

where the factors are determined as follows:

(a) $Q_1(\lambda) = N_l(\lambda)$, with $N_l(\lambda)$ a $\big(p - r_d\big) \times (p + m_u)$ proper rational left nullspace basis satisfying $N_l(\lambda)G(\lambda) = 0$ (recall that $r_d := \operatorname{rank} G_d(\lambda)$);

(b) $Q_2(\lambda)$ is a $q \times (p - r_d)$ admissible regularization factor guaranteeing complete fault detectability or strong fault isolability;

(c) $Q_3(\lambda)$ is the inverse or left inverse of a quasi-co-outer factor;

(d) $Q_4(\lambda)$ is the solution of a least distance problem (LDP);

(e) $Q_5(\lambda)$ is a stable invertible factor determined such that $Q(\lambda)$ has a desired dynamics.

The computations of individual factors depend on the user's options and specific choices are discussed in what follows.

Computation of $Q_1(\lambda)$

If `OPTIONS.nullspace = true` or $m_d > 0$ or E is singular, then $N_l(\lambda)$ is determined as a minimal proper nullspace basis. In this case, if `OPTIONS.simple = true`, then $N_l(\lambda)$ is determined as a simple rational basis and the orders of the basis vectors are provided in `INFO.degs`. If `OPTIONS.simple = false`, then $N_l(\lambda)$ is determined as a proper rational basis and `INFO.degs` contains the degrees of the basis vectors of an equivalent polynomial nullspace basis (see [16, Section 9.1.3] for definitions).

If `OPTIONS.nullspace = false`, $m_d = 0$ and E is invertible, then $N_l(\lambda) = [\, I_p \ \ -G_u(\lambda)\,]$ is used, which corresponds to a full-order Luenberger observer.

To check the solvability of the AMMP, the transfer function matrix $\overline{G}_f(\lambda) := Q_1(\lambda)\left[\begin{smallmatrix} G_f(\lambda) \\ 0 \end{smallmatrix}\right]$ is determined and let H be a full row rank design matrix, which is either specified in a nonempty `OPTIONS.HDesign`, or otherwise $H = I_{p-r_d}$ is assumed. The AMMP with complete detectability requirement is solvable provided $H\overline{G}_f(\lambda_s)$ has all its columns nonzero, where λ_s is a suitable frequency value, which can be specified via the `OPTIONS.freq`. To check the solvability of the AMMP with a strong isolability condition, we check that $H\overline{G}_f(\lambda)$ has full column rank m_f. This rank condition is verified by checking the left invertibility condition

$$\operatorname{rank} H\overline{G}_f(\lambda_s) = m_f. \tag{215}$$

In the case when `OPTIONS.freq` is empty, the employed frequency value λ_s is provided in `INFO.freq`.

Computation of $Q_2(\lambda)$

If `OPTIONS.regmin = false`, then $Q_2(\lambda) = H$, where H is a suitable $q\times\big(p-r_d\big)$ full row rank design matrix. H is set as follows. If `OPTIONS.HDesign` is nonempty, then $H =$ `OPTIONS.HDesign`. If `OPTIONS.HDesign` is empty, then the matrix H is chosen such that, either: (1) $H\overline{G}_f(\lambda)$ is invertible, if the solution of a strong FDI problem is targeted; or (2) $H\overline{G}_f(\lambda)$ has all its columns nonzero, if the focus is on guaranteeing complete fault detectability. In both cases, H is built from an admissible set of q rows of the identity matrix I_{p-r_d}.

If `OPTIONS.regmin = true`, then $Q_2(\lambda)$ is a $q\times\big(p-r_d\big)$ transfer function matrix determined in the form

$$Q_2(\lambda) = H + Y_2(\lambda)\,,$$

where $Q_2(\lambda)Q_1(\lambda)$ $\big(= HN_l(\lambda)+Y_2(\lambda)N_l(\lambda)\big)$ and $Y_2(\lambda)$ are the least order solution of a left minimal cover problem [15]. If `OPTIONS.HDesign` is nonempty, then $H =$ `OPTIONS.HDesign`, and if `OPTIONS.HDesign` is empty, then H is chosen as above. This choice ensures that either: (1) $Q_2(\lambda)\overline{G}_f(\lambda)$ is invertible, if a strong FDI problem is solved; or (2) $H\overline{G}_f(\lambda)$ has all its columns nonzero, if the focus is on guaranteeing complete fault detectability.

The structure field `INFO.HDesign` contains the employed value of the design matrix H.

Computation of $Q_3(\lambda)$

Let $\tilde{r}$ be the rank of $[\,\widetilde{G}_f(\lambda)\mid\widetilde{G}_w(\lambda)\,] := Q_2(\lambda)[\,\overline{G}_f(\lambda)\mid\overline{G}_w(\lambda)\,]$. If a strong FDI problem is solved, then $\tilde{r} = m_f = q$. The extended quasi-co-outer–co-inner factorization of $[\,\widetilde{G}_f(\lambda)\;\widetilde{G}_w(\lambda)\,]$ is computed as

$$[\,\widetilde{G}_f(\lambda)\;\widetilde{G}_w(\lambda)\,] = [\,G_o(\lambda)\;0\,]\left[\begin{array}{c} G_{i,1}(\lambda)\\ G_{i,2}(\lambda)\end{array}\right],$$

where $G_o(\lambda)$ is a $q\times\tilde{r}$ full column rank quasi-co-outer factor and $G_i(\lambda) = \left[\begin{smallmatrix} G_{i,1}(\lambda)\\ G_{i,2}(\lambda)\end{smallmatrix}\right]$ is a square inner factor. Note that the potential lack of stability of $G_o(\lambda)$ is not relevant at this stage for the employed solution method. If $\tilde{r} = q$ then, $Q_3(\lambda) = G_o^{-1}(\lambda)$, otherwise $Q_3(\lambda) = G_o^{-L}(\lambda)$, with $G_o^{-L}(\lambda)$ a left inverse of $G_o(\lambda)$, determined such that all its free poles are assigned into the stable domain $\mathbb{C}_s$. It follows, that in the *standard case*, when $G_o(\lambda)$ has no zeros in $\partial\mathbb{C}_s$, $Q_3(\lambda)$ results stable, while in the *non-standard case*, when $G_o(\lambda)$ has zeros in $\partial\mathbb{C}_s$, $Q_3(\lambda)$ results with poles which are either stable or lie in $\partial\mathbb{C}_s$. These latter poles are precisely the unstable zeros of $G_o(\lambda)$ in $\partial\mathbb{C}_s$. The information on the type of the problem to be solved is returned in `INFO.nonstandard`, which is set equal to `false` for a standard problem, and `true` for a non-standard problem.

Computation of $Q_4(\lambda)$

With $\widetilde{F}_1(\lambda) = [\,M_r(\lambda)\;0\,]G^\sim_{i,1}(\lambda)$ and $\widetilde{F}_2(\lambda) = [\,M_r(\lambda)\;0\,]G^\sim_{i,2}(\lambda)$, the $q\times\tilde{r}$ TFM $Q_4(\lambda)$ is determined as the optimal solution of the $\mathcal{H}_{\infty/2}$ least distance problem ($\mathcal{H}_{\infty/2}$-LDP)

$$\gamma_{opt} = \min_{Q_4(\lambda)\in\mathcal{H}_\infty}\left\|\left[\;\widetilde{F}_1(\lambda) - Q_4(\lambda)\;\;\widetilde{F}_2(\lambda)\;\right]\right\|_{\infty/2}. \tag{216}$$

To ensure the existence of the solution of a $\mathcal{H}_2$-LDP in the continuous-time case with $\widetilde{F}_2(s)$ a non-strictly-proper transfer function matrix, a strictly proper and stable updating factor

$\widetilde{M}(s) = \frac{k}{s+k}I$ is used, to form an updated reference model $\widetilde{M}(s)M_r(s)$ and the updated $\widetilde{F}_1(\lambda) = \widetilde{M}(\lambda)[\, M_r(\lambda)\ 0\,]G_{i,1}^{\sim}(\lambda)$ and $\widetilde{F}_2(\lambda) = \widetilde{M}(\lambda[\, M_r(\lambda)\ 0\,]G_{i,2}^{\sim}(\lambda)$ are used. $\widetilde{M}(\lambda) = I$ in all other cases.

The value of γ_{opt} in (216) is returned in `INFO.gammaopt`. For a standard $\mathcal{H}_\infty$-norm based synthesis, γ_{opt} is the optimal approximation error $\|\mathcal{E}(\lambda)\|_\infty$, while for a non-standard problem this value is the optimal $\mathcal{H}_\infty$ least distance which corresponds to an improper or unstable filter. For a standard $\mathcal{H}_2$-norm based synthesis, γ_{opt} is the optimal approximation error $\|\mathcal{E}(\lambda)\|_2$, while for a non-standard problem this value is the optimal $\mathcal{H}_2$ least distance which corresponds to an improper or unstable filter.

Computation of $Q_5(\lambda)$

In the standard case, $Q_5(\lambda) = I$. In the non-standard case, $Q_5(\lambda)$ is a stable, diagonal and invertible transfer function matrix determined such that $Q(\lambda)$ in (214) has a desired dynamics (specified via `OPTIONS.sdeg` and `OPTIONS.poles`). The overall updating factor used in (213) is $M(\lambda) = Q_5(\lambda)\widetilde{M}(\lambda)$ and is provided in `INFO.M`.

(a) *The case with* $\big[\, M_{ru}(\lambda)\ M_{rd}(\lambda)\ M_{rw}(\lambda)\,\big] \neq 0$.

In the second case, we solve the AMMP, as formulated in the more general form (39) in Remark 10. To address the solution of this problem, we formulate a general model-matching problems of the form (204) and solve these problems using general solvers as `glasol`, available in the Descriptor System Tools (DSTOOLS) [4].

The filter $Q(\lambda)$ is determined in the product form

$$Q(\lambda) = Q_3(\lambda)Q_2(\lambda)Q_1(\lambda), \tag{217}$$

where the factors are determined as follows:

(a) $Q_1(\lambda)$ is a proper rational left nullspace basis satisfying $Q_1(\lambda)\widetilde{G}(\lambda) = 0$ for a suitably defined $\widetilde{G}(\lambda)$ (see below);

(b) $Q_2(\lambda)$ is the solution of a reduced or unreduced approximate model-matching problem;

(c) $Q_3(\lambda)$ is a stable invertible factor determined such that $Q(\lambda)$ has a desired dynamics.

The computations of individual factors depend on the user's options and specific choices are discussed in what follows.

Computation of $Q_1(\lambda)$

Three cases are considered.

(i) If both $M_{ru}(\lambda) \neq 0$ and $M_{rd}(\lambda) \neq 0$, then $Q_1(\lambda) = I_{p+m_u}$ and the corresponding reduced system and reference model are $\overline{R}(\lambda) := G_e(\lambda)$ and $\overline{M}_r(\lambda) := M_r(\lambda)$. Note that $\widetilde{G}(\lambda)$ is an $(p + m_u) \times 0$ (empty) matrix.

(*ii*) If $M_{ru}(\lambda) = 0$, but $M_{rd}(\lambda) \neq 0$, then $Q_1(\lambda)$ is a left proper rational nullspace basis of $\widetilde{G}(\lambda) := \left[\begin{smallmatrix} G_u(\lambda) \\ I_{m_u} \end{smallmatrix}\right]$.

If `OPTIONS.nullspace = true` or E is singular, then $Q_1(\lambda)$ is determined as a minimal proper nullspace basis. In this case, if `OPTIONS.simple = true`, then $Q_1(\lambda)$ is determined as a simple rational basis, while if `OPTIONS.simple = false`, then $Q_1(\lambda)$ is determined as a proper rational basis. The corresponding reduced system and reference model are

$$\overline{R}(\lambda) := Q_1(\lambda) \left[\begin{array}{c|c|c} G_d(\lambda) & G_f(\lambda) & G_w(\lambda) \\ 0 & 0 & 0 \end{array}\right], \qquad \overline{M}_r(\lambda) = \left[\, M_{rd}(\lambda) \mid M_{rf}(\lambda) \mid M_{rw}(\lambda) \,\right].$$

If `OPTIONS.nullspace = false` and E is invertible, then $Q_1(\lambda) = [\, I_p \;\; -G_u(\lambda) \,]$ is used, which corresponds to a full-order Luenberger observer. The corresponding reduced system and reference model are

$$\overline{R}(\lambda) := \left[\, G_d(\lambda) \mid G_f(\lambda) \mid G_w(\lambda) \,\right], \qquad \overline{M}_r(\lambda) = \left[\, M_{rd}(\lambda) \mid M_{rf}(\lambda) \mid M_{rw}(\lambda) \,\right].$$

(*iii*) If both $M_{ru}(\lambda) = 0$ and $M_{rd}(\lambda) = 0$, then $Q_1(\lambda)$ is a left proper rational nullspace basis of $\widetilde{G}(\lambda) := \left[\begin{smallmatrix} G_u(\lambda) & G_d(\lambda) \\ I_{m_u} & 0 \end{smallmatrix}\right]$. If the resulting nullspace is empty, then the case (*ii*) can be applied.

If `OPTIONS.nullspace = true` or $m_d > 0$ or E is singular, then $Q_1(\lambda)$ is determined as a minimal proper nullspace basis. In this case, if `OPTIONS.simple = true`, then $Q_1(\lambda)$ is determined as a simple rational basis and if `OPTIONS.simple = false`, then $Q_1(\lambda)$ is determined as a proper rational basis. The corresponding reduced system and reference model are

$$\overline{R}(\lambda) := Q_1(\lambda) \left[\begin{array}{c|c} G_f(\lambda) & G_w(\lambda) \\ 0 & 0 \end{array}\right], \qquad \overline{M}_r(\lambda) = \left[\, M_{rf}(\lambda) \mid M_{rw}(\lambda) \,\right].$$

If `OPTIONS.nullspace = false`, $m_d = 0$ and E is invertible, then $Q_1(\lambda) = [\, I_p \;\; -G_u(\lambda) \,]$ is used, which corresponds to a full-order Luenberger observer. The corresponding reduced system and reference model are

$$\overline{R}(\lambda) := \left[\, G_f(\lambda) \mid G_w(\lambda) \,\right], \qquad \overline{M}_r(\lambda) = \left[\, M_{rf}(\lambda) \mid M_{rw}(\lambda) \,\right].$$

Computation of $Q_2(\lambda)$

$Q_2(\lambda)$ is computed as the optimal solution which minimizes the model-matching error norm such that

$$\| Q_2(\lambda)\overline{R}_1(\lambda) - \overline{M}_r(\lambda) \|_{\infty/2} = \min . \tag{218}$$

The standard-case corresponds to $\overline{R}_1(\lambda)$ without zeros in $\partial\mathbb{C}_s$, while the non-standard case corresponds to $\overline{R}_1(\lambda)$ having zeros in $\partial\mathbb{C}_s$.

Computation of $Q_3(\lambda)$

In the standard case, $Q_3(\lambda) = I$. In the non-standard case, $Q_3(\lambda)$ is a stable, diagonal and invertible transfer function matrix determined such that $Q(\lambda)$ in (217) has a desired dynamics (specified via `OPTIONS.sdeg` and `OPTIONS.poles`). The overall updating factor used in (185) is $M(\lambda) = Q_3(\lambda)$ and is provided in `INFO.M`.

In this case, `INFO.degs = [ ]` and `INFO.HDesign = [ ]` are returned in the `INFO` structure.

Case 2: `SYSR` contains a collection of reference models

If `SYSR` contains n_b reference models with the i-th model having the input-output form

$$\mathbf{y}_r^{(i)}(\lambda) = M_{ru}^{(i)}(\lambda)\mathbf{u}(\lambda) + M_{rd}^{(i)}(\lambda)\mathbf{d}(\lambda) + M_{rf}^{(i)}(\lambda)\mathbf{f}(\lambda) + M_{rw}^{(i)}(\lambda)\mathbf{w}(\lambda), \tag{219}$$

where $y_r^{(i)}$ has dimension $q^{(i)}$, then the resulting `Q` contains n_b filters, with the i-th filter in (198) having the input-output form

$$\mathbf{r}^{(i)}(\lambda) = Q^{(i)}(\lambda)\left[\begin{array}{c}\mathbf{y}(\lambda)\\ \mathbf{u}(\lambda)\end{array}\right], \tag{220}$$

where the resulting dimension of the residual vector $r^{(i)}$ is $q^{(i)}$. $Q^{(i)}(\lambda)$ is determined by solving the general approximate model-matching problem

$$Q^{(i)}(\lambda)G_e(\lambda) \approx M^{(i)}(\lambda)M_r^{(i)}(\lambda)\,, \tag{221}$$

where

$$M_r^{(i)}(\lambda) := \left[\, M_{ru}^{(i)}(\lambda) \mid M_{rd}^{(i)}(\lambda) \mid M_{rf}^{(i)}(\lambda) \mid M_{rw}^{(i)}(\lambda) \,\right],$$

$G_e(\lambda)$ is defined in (205) and $M^{(i)}(\lambda)$ is a stable, diagonal and invertible transfer function matrix chosen such that the resulting approximate solution $Q^{(i)}(\lambda)$ of (221) is stable and proper. The resulting internal form $R^{(i)}(\lambda)$, contained in `R{`i`}`, is computed as

$$R^{(i)}(\lambda) := \left[\, R_u^{(i)}(\lambda) \,|\, R_d^{(i)}(\lambda) \,|\, R_f^{(i)}(\lambda) \,|\, R_w^{(i)}(\lambda) \,\right] := Q^{(i)}(\lambda)G_e(\lambda). \tag{222}$$

In the *standard case*, $G_e(\lambda)$ has no zeros on the boundary of the stability domain $\partial\mathbb{C}_s$, and the resulting stable filter $Q^{(i)}(\lambda) = Q_{opt}^{(i)}(\lambda)$, is the optimal solution of the $\mathcal{H}_\infty$- or $\mathcal{H}_2$-norm error minimization problem

$$\gamma_{opt,0}^{(i)} := \|Q_{opt}^{(i)}(\lambda)G_e(\lambda) - M_0^{(i)}(\lambda)M_r^{(i)}(\lambda)\|_{\infty/2} = \min, \tag{223}$$

where $M_0^{(i)}(\lambda) = I$ in the case of $\mathcal{H}_\infty$-norm or of a discrete-time system. In the case of $\mathcal{H}_2$-norm and a continuous-time system, $M_0^{(i)}(s)$ is determined as a stable, diagonal, and invertible transfer function matrix, which ensures the existence of a finite $\mathcal{H}_2$-norm.

In the *non-standard case*, $G_e(\lambda)$ has zeros on the boundary of the stability domain $\partial\mathbb{C}_s$, and the optimal solution $Q_{opt}^{(i)}(\lambda)$ of (223) is possibly unstable or improper. In this case, $Q^{(i)}(\lambda)$ is chosen as $Q^{(i)}(\lambda) = M_1^{(i)}(\lambda)Q_{opt}^{(i)}(\lambda)$, with $M_1^{(i)}(\lambda)$ a stable, diagonal, and invertible transfer function matrix determined to ensure the stability of $Q^{(i)}(\lambda)$. In this case, $Q^{(i)}(\lambda)$ can be interpreted as a suboptimal solution of the updated $\mathcal{H}_\infty$- or $\mathcal{H}_2$-norm error minimization problem

$$\gamma_{opt}^{(i)} := \|\widetilde{Q}_{opt}^{(i)}(\lambda)G_e(\lambda) - M_1^{(i)}(\lambda)M_0^{(i)}(\lambda)M_r^{(i)}(\lambda)\|_{\infty/2} = \min, \tag{224}$$

whose optimal solution $\widetilde{Q}_{opt}^{(i)}(\lambda)$ is possibly unstable or improper, but for which $Q^{(i)}(\lambda)$ represents a stable and proper suboptimal solution, with the corresponding suboptimal model-matching error norm

$$\gamma_{sub}^{(i)} := \|Q^{(i)}(\lambda)G_e(\lambda) - M_1^{(i)}(\lambda)M_0^{(i)}(\lambda)M_r^{(i)}(\lambda)\|_{\infty/2}. \tag{225}$$

The value of $\gamma_{opt,0}^{(i)}$ in (223) is returned in `INFO.gammaopt0`(i), and also in `INFO.gammaopt`(i) and `INFO.gammasub`(i) in the *standard case*. In the *non-standard case*, the value $\gamma_{opt}^{(i)}$ in (224) is returned in `INFO.gammaopt`(i) and the value of $\gamma_{sub}^{(i)}$ in (225) is returned in `INFO.gammasub`(i). The updating matrix $M^{(i)}(\lambda) := M_0^{(i)}(\lambda)$, in the *standard case*, or $M^{(i)}(\lambda) := M_1^{(i)}(\lambda)M_0^{(i)}(\lambda)$, in the *non-standard case*, is returned in `INFO.M`$\{i\}$.

For the computation of $Q^{(i)}(\lambda)$ and $R^{(i)}(\lambda)$, for $i = 1, \ldots, n_b$, the previously sketched computational approaches are employed for all filters, taking into account the selected synthesis options in the `OPTION` structure. The resulting `INFO.M` and `OPTIONS.HDesign` are $n_b \times 1$ cell arrays, and `INFO.M`$\{i\}$ contains the state-space realization of $M^{(i)}(\lambda)$, while `INFO.HDesign`$\{i\}$ contains the design matrix H_i employed for the synthesis of the i-th filter. `INFO.HDesign = [ ]` if no design matrices have been involved.

Example 14. This is *Example* 5.11 from the book [16], with a continuous-time system with additive faults, having the transfer function matrices

$$G_u(s) = \begin{bmatrix} \dfrac{s+1}{s+2} \\ \dfrac{s+2}{s+3} \end{bmatrix}, \quad G_d(s) = 0, \quad G_f(s) = \begin{bmatrix} \dfrac{s+1}{s+2} & 0 \\ 0 & 1 \end{bmatrix}, \quad G_w(s) = \begin{bmatrix} \dfrac{1}{s+2} \\ 0 \end{bmatrix}.$$

The maximally achievable structure matrix is

$$S_{max} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

and therefore we can target to solve an AMMP with strong fault isolability using the following reference model

$$M_r(s) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

This involves to determine a stable $Q(s)$, and possibly also an updating factor $M(s)$, which fulfill

$$\gamma_{opt} := \left\| Q(s) \begin{bmatrix} G_u(s) & G_d(s) & G_f(s) & G_w(s) \\ I_{m_u} & 0 & 0 & 0 \end{bmatrix} - M(s)[\,0\;0\;M_r(s)\;0\,] \right\|_\infty = \min.$$

A least order stable optimal filter $Q(s)$ has been determined by employing **ammsyn** with $M(s) = I_2$. The resulting optimal performance is $\gamma_{opt} = \frac{\sqrt{2}}{2} = 0.7071$. The resulting $Q(s)$ is

$$Q(s) = \left[\begin{array}{ccc} 0.7072\dfrac{s+2}{s+\sqrt{2}} & 0 & -0.7072\dfrac{s+1}{s+\sqrt{2}} \\ 0 & 0.7072 & -0.7072\dfrac{s+2}{s+3} \end{array} \right]$$

and the resulting $R_f(s)$ and $R_w(s)$ are

$$R_f(s) = \left[\begin{array}{cc} 0.7072\dfrac{s+1}{s+\sqrt{2}} & 0 \\ 0 & 0.7072 \end{array} \right], \quad R_w(s) = \left[\begin{array}{c} 0.7072\dfrac{1}{s+\sqrt{2}} \\ 0 \end{array} \right].$$

The fault-to-noise gaps can be computed using the function **fdif2ngap**, by assuming as structure matrix S_{FDI}, the structure matrix of the reference model $M_r(s)$ (i.e., $S_{FDI} = I_2$). The resulting filter $Q(s)$ can be considered formed by column concatenating two separate filters $Q^{(1)}(s)$ and $Q^{(2)}(s)$, which aims to match the first and second rows of $M_r(s)$, respectively. The resulting fault-to-noise gaps are respectively, $\sqrt{2}$ and ∞, which indicate that the second filter solves, in fact, an EMMP.

The above results have been computed with the following script.

```
% Example - Solution of an AMMP

% define s as an improper transfer function
s = tf('s');
Gu = [(s+1)/(s+2); (s+2)/(s+3)];  % enter Gu(s)
Gf = [(s+1)/(s+2) 0; 0 1];        % enter Gf(s)
Gw = [1/(s+2); 0];                % enter Gw(s)
mu = 1; mf = 2; mw = 1; p = 2; % set dimensions

% build the synthesis model with additive faults
inputs = struct('c',1:mu,'f',mu+(1:mf),'n',mu+mf+(1:mw));
sysf = fdimodset(ss([Gu Gf Gw]),inputs);

% determine the maximally achievable structure matrix
Smax = fdigenspec(sysf,struct('tol',1.e-7))

% choose the targeted reference model
Mr = fdimodset(ss(eye(mf)),struct('faults',1:mf));

% solve the AMMP using AMMSYN
opts_ammsyn = struct('tol',1.e-7,'reltol',5.e-8);
[Q,R,info] = ammsyn(sysf,Mr,opts_ammsyn);
```

```
% display results
minreal(zpk(Q)),  tf(info.M)
Rf = minreal(zpk(R(:,'faults'))), Rw = minreal(zpk(R(:,'noise')))

% optimal and suboptimal performance, and achieved gaps
info
format short e
gap = fdif2ngap(R,[],fditspec(Mr))

% check synthesis performance
gammaopt = fdimmperf(R,Mr)

% check decoupling condition
Ge = [sysf;eye(mu,mu+mf+mw)];
norm_Ru = norm(Q*Ge(:,'controls'),inf)

% check synthesis results
norm_dif = norm(R-Q*Ge(:,{'faults','noise'}),inf)
```

4.9 Functions for the Synthesis of Model Detection Filters

4.9.1 `emdsyn`

Syntax

```
[Q,R,INFO] = emdsyn(SYSM,OPTIONS)
```

Description

`emdsyn` solves the *exact model detection problem* (EMDP) (see Section 3.5.1), for a given stable LTI multiple model `SYSM` containing N models. A bank of N stable and proper residual generation filters $Q^{(i)}(\lambda)$, for $i = 1,\ldots,N$, is determined, in the form (45). For each filter $Q^{(i)}(\lambda)$, its associated internal forms $R^{(i,j)}(\lambda)$, for $j = 1,\ldots,N$, are determined in accordance with (47).

Input data

`SYSM` is a multiple model which contains N stable LTI systems in the state-space form

$$\begin{aligned} E^{(j)}\lambda x^{(j)}(t) &= A^{(j)}x^{(j)}(t) + B_u^{(j)}u^{(j)}(t) + B_d^{(j)}d^{(j)}(t) + B_w^{(j)}w^{(j)}(t)\,, \\ y^{(j)}(t) &= C^{(j)}x^{(j)}(t) + D_u^{(j)}u^{(j)}(t) + D_d^{(j)}d^{(j)}(t) + D_w^{(j)}w^{(j)}(t)\,, \end{aligned} \tag{226}$$

where $x^{(j)}(t) \in \mathbb{R}^{n^{(j)}}$ is the state vector of the j-th system with control input $u^{(j)}(t) \in \mathbb{R}^{m_u}$, disturbance input $d^{(j)}(t) \in \mathbb{R}^{m_d^{(j)}}$ and noise input $w^{(j)}(t) \in \mathbb{R}^{m_w^{(j)}}$, and where any of the inputs components $u^{(j)}(t)$, $d^{(j)}(t)$, or $w^{(j)}(t)$ can be void. The multiple model `SYSM` is either an array of N LTI systems of the form (226), in which case $m_d^{(j)} = m_d$ and $m_w^{(j)} = m_w$ for $j = 1,\ldots,N$, or is a $1\times N$ cell array, with `SYSM`$\{j\}$ containing the j-th component system in the form (226). The input groups for $u^{(j)}(t)$, $d^{(j)}(t)$, and $w^{(j)}(t)$ have the standard names `'controls'`, `'disturbances'`, and `'noise'`, respectively.

`OPTIONS` is a MATLAB structure used to specify various synthesis options and has the following fields:

`OPTIONS` **fields**	**Description**
`tol`	relative tolerance for rank computations (Default: internally computed)
`tolmin`	absolute tolerance for observability tests (Default: internally computed)
`MDTol`	threshold for model detectability checks (Default: 10^{-4})
`MDGainTol`	threshold for strong model detectability checks (Default: 10^{-2})

`rdim`	N-dimensional vector or a scalar; for a vector q, the i-th component q_i specifies the desired number of residual outputs for the i-th filter `Q`$\{i\}$; for a scalar value $\bar{q}$, a vector q with all N components $q_i = \bar{q}$ is assumed. (Default: `[ ]`, in which case: – if `OPTIONS.HDesign`$\{i\}$ is empty, then $q_i = 1$, if `OPTIONS.minimal` = `true`, or $q_i = n_v^{(i)}$, the dimension of the left nullspace which underlies the synthesis of `Q`$\{i\}$, if `OPTIONS.minimal` = `false` (see **Method**); – if `OPTIONS.HDesign`$\{i\}$ is nonempty, then q_i is the row dimension of the design matrix contained in `OPTIONS.HDesign`$\{i\}$.)
`MDFreq`	real vector, which contains the frequency values ω_k, $k = 1, \ldots, n_f$, to be used for strong model detectability checks. For each real frequency ω_k, there corresponds a complex frequency λ_k which is used to evaluate the frequency-response gains. Depending on the system type, $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the common sampling time of the component systems. (Default: `[ ]`)
`emdtest`	option to perform extended model detectability tests using both control and disturbance input channels: `true` – use both control and disturbance input channels; `false` – use only the control channel (default).
`smarg`	prescribed stability margin for the resulting filters `Q`$\{i\}$ (Default: `-sqrt(eps)` for continuous-time component systems; `1-sqrt(eps)` for discrete-time component systems.
`sdeg`	prescribed stability degree for the resulting filters `Q`$\{i\}$ (Default: -0.05 for continuous-time component systems; 0.95 for discrete-time component systems.
`poles`	complex vector containing a complex conjugate set of desired poles (within the stability margin) to be assigned for the resulting filters `Q`$\{i\}$ (Default: `[ ]`)
`nullspace`	option to use a specific type of proper nullspace bases: `true` – use minimal proper bases; `false` – use full-order observer based bases (default) *Note:* This option can only be used if no disturbance inputs are present in (226) and $\forall j$, $E^{(j)}$ is invertible.
`simple`	option to compute simple proper bases: `true` – compute simple bases; the orders of the basis vectors are provided in `INFO.degs`; `false` – no simple basis computed (default)
`minimal`	option to perform least order filter syntheses: `true` – perform least order syntheses (default); `false` – perform full order syntheses.

`tcond`	maximum allowed value for the condition numbers of the employed non-orthogonal transformation matrices (Default: 10^4) (only used if `OPTIONS.simple = true`)
`MDSelect`	integer vector with increasing elements containing the indices of the desired filters to be designed (Default: $[1,\ldots,N]$)
`HDesign`	N-dimensional cell array; `OPTIONS.HDesign`$\{i\}$, if not empty, is a full row rank design matrix employed for the synthesis of the i-th filter. If `OPTIONS.HDesign` is specified as a full row rank design matrix H, then an N-dimensional cell array is assumed with `OPTIONS.HDesign`$\{i\} = H$, for $i = 1,\ldots,N$. (Default: `[ ]`).
`normalize`	option to normalize the the filters `Q{i}` and `R{i,j}` such that the minimum gains of the off-diagonal elements of `R{i,j}` are equal to one; otherwise the standard normalization is performed to ensure equal gains for `R{1,j}` and `R{j,1}` : `true` – perform normalization to unit minimum gains; `false` – perform standard normalization (default).

Output data

`Q` is an $N \times 1$ cell array of filters, where `Q`$\{i\}$ contains the resulting i-th filter in a standard state-space representation

$$\begin{aligned}\lambda x_Q^{(i)}(t) &= A_Q^{(i)} x_Q^{(i)}(t) + B_{Q_y}^{(i)} y(t) + B_{Q_u}^{(i)} u(t),\\ r^{(i)}(t) &= C_Q^{(i)} x_Q^{(i)}(t) + D_{Q_y}^{(i)} y(t) + D_{Q_u}^{(i)} u(t),\end{aligned}$$

where the residual signal $r^{(i)}(t)$ is a q_i-dimensional vector, with q_i specified in `OPTIONS.rdim`. For each system object `Q`$\{i\}$, two input groups `'outputs'` and `'controls'` are defined for $y(t)$ and $u(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r^{(i)}(t)$. `Q`$\{i\}$ is empty for all i which do not belong to the index set specified by `OPTIONS.MDSelect`.

`R` is an $N \times N$ cell array of filters, where the (i,j)-th filter `R`$\{i,j\}$, is the internal form of `Q`$\{i\}$ acting on the j-th model. The resulting `R`$\{i,j\}$ has a standard state-space representation

$$\begin{aligned}\lambda x_R^{(i,j)}(t) &= A_R^{(i,j)} x_R^{(i,j)}(t) + B_{R_u}^{(i,j)} u^{(j)}(t) + B_{R_d}^{(i,j)} d^{(j)}(t) + B_{R_w}^{(i,j)} w^{(j)}(t),\\ r^{(i,j)}(t) &= C_R^{(i,j)} x_R^{(i,j)}(t) + D_{R_u}^{(i,j)} u^{(j)}(t) + D_{R_d}^{(i,j)} d^{(j)}(t) + D_{R_w}^{(i,j)} w^{(j)}(t)\end{aligned}$$

and the input groups `'controls'`, `'disturbances'` and `'noise'` are defined for $u^{(j)}(t)$, $d^{(j)}(t)$, and $w^{(j)}(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r^{(i,j)}(t)$. `R`$\{i,j\}$, $j = 1,\ldots,N$ are empty for all i which do not belong to the index set specified by `OPTIONS.MDSelect`.

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

`INFO` **fields**	**Description**
`tcond`	N-dimensional vector; `INFO.tcond`(i) contains the maximum of the condition numbers of the non-orthogonal transformation matrices used to determine the i-th filter `Q`$\{i\}$; a warning is issued if any `INFO.tcond`(i) $\geq$ `OPTIONS.tcond`.
`degs`	N-dimensional cell array; if `OPTIONS.simple` = `true`, then a nonempty `INFO.degs`$\{i\}$ contains the orders of the basis vectors of the employed simple nullspace basis for the synthesis of the i-th filter component `Q`$\{i\}$; if `OPTIONS.simple` = `false`, then a nonempty `INFO.degs`$\{i\}$ contains the degrees of the basis vectors of an equivalent polynomial nullspace basis; `INFO.degs`$\{i\}$ `= [ ]` for all i which do not belong to the index set specified by `OPTIONS.MDSelect`.
`MDperf`	$N \times N$-dimensional array containing the achieved distance mapping performance, given as the peak gains associated with the internal representations (see **Method**). `INFO.MDperf`(i,j) $= -1$, for $j = 1,\ldots,N$ and for all i which do not belong to the index set specified by `OPTIONS.MDSelect`.
`HDesign`	N-dimensional cell array, where `INFO.HDesign`$\{i\}$ contains the i-th design matrix $H^{(i)}$ employed for the synthesis of the i-th filter (see **Method**)

Method

An extension of the **Procedure EMD** from [16, Sect. 6.2] is implemented, which relies on the nullspace-based synthesis method proposed in [10]. Assume that the j-th model has the input-output form

$$\mathbf{y}^{(j)}(\lambda) = G_u^{(j)}(\lambda)\mathbf{u}^{(j)}(\lambda) + G_d^{(j)}(\lambda)\mathbf{d}^{(j)}(\lambda) + G_w^{(j)}(\lambda)\mathbf{w}^{(j)}(\lambda) \tag{227}$$

and the resulting i-th filter $Q^{(i)}(\lambda)$ has the input-output form

$$\mathbf{r}^{(i)}(\lambda) = Q^{(i)}(\lambda)\left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array}\right] . \tag{228}$$

The synthesis method, which underlies **Procedure EMD**, essentially determines each filter $Q^{(i)}(\lambda)$ as a stable rational left annihilator of

$$G^{(i)}(\lambda) := \left[\begin{array}{cc} G_u^{(i)}(\lambda) & G_d^{(i)}(\lambda) \\ I_{m_u} & 0 \end{array}\right],$$

such that for $i \neq j$ we have $[\, R_u^{(i,j)}(\lambda)\; R_d^{(i,j)}(\lambda)\,] \neq 0$, where

$$R^{(i,j)}(\lambda) := [\, R_u^{(i,j)}(\lambda)\; R_d^{(i,j)}(\lambda)\; R_w^{(i,j)}(\lambda)\,] = Q^{(i)}(\lambda)\left[\begin{array}{ccc} G_u^{(j)}(\lambda) & G_d^{(j)}(\lambda) & G_w^{(j)}(\lambda) \\ I_{m_u} & 0 & 0 \end{array}\right]$$

is the internal form of $Q^{(i)}(\lambda)$ acting on the j-th model.

Each filter $Q^{(i)}(\lambda)$ is determined in the product form

$$Q^{(i)}(\lambda) = Q_3^{(i)}(\lambda)Q_2^{(i)}(\lambda)Q_1^{(i)}(\lambda), \tag{229}$$

where the factors are determined as follows:

(a) $Q_1^{(i)}(\lambda) = N_l^{(i)}(\lambda)$, with $N_l^{(i)}(\lambda)$ a $\big(p - r_d^{(i)}\big) \times (p + m_u)$ proper rational left nullspace basis satisfying $N_l^{(i)}(\lambda)G^{(i)}(\lambda) = 0$, with $r_d^{(i)} := \operatorname{rank} G_d^{(i)}(\lambda)$; ($n_v^{(i)} := p - r_d^{(i)}$ is the number of basis vectors)

(b) $Q_2^{(i)}(\lambda)$ is an admissible factor (i.e., guaranteeing model detectability) to perform least order synthesis;

(c) $Q_3^{(i)}(\lambda)$ is a stable invertible factor determined such that $Q^{(i)}(\lambda)$ has a desired dynamics.

The computations of individual factors depend on the user's options. Specific choices are discussed in what follows.

Computation of $Q_1^{(i)}(\lambda)$

If `OPTIONS.nullspace = true`, then the left nullspace basis $N_l^{(i)}(\lambda)$ is determined as a minimal proper rational basis, or, if `OPTIONS.nullspace = false` (the default option) and $m_d^{(i)} = 0$, then the simple observer based basis $N_l^{(i)}(\lambda) = [\, I \;\; -G_u^{(i)}(\lambda)\,]$ is employed. If $N_l^{(i)}(\lambda)$ is a minimal rational basis and if `OPTIONS.simple = true`, then $N_l^{(i)}(\lambda)$ is determined as a simple rational basis and the orders of the basis vectors are provided in `INFO.degs`. These are also the degrees of the basis vectors of an equivalent polynomial nullspace basis. If `OPTIONS.minimal = false`, a stable basis is determined whose dynamics is specified via `OPTIONS.sdeg` and `OPTIONS.poles`.

Computation of $Q_2^{(i)}(\lambda)$

If `OPTIONS.minimal = false`, then $Q_2^{(i)}(\lambda) = H^{(i)}$, where $H^{(i)}$ is a suitable $q_i \times \big(p - r_d^{(i)}\big)$ full row rank design matrix. $H^{(i)}$ is set as follows. If `OPTIONS.HDesign{`i`}` is nonempty, then $H^{(i)} =$ `OPTIONS.HDesign{`i`}`. If `OPTIONS.HDesign{`i`}` is empty, then $q_i =$ `OPTIONS.rdim` if `OPTIONS.rdim` is nonempty and $q_i = p - r_d^{(i)}$ if `OPTIONS.rdim` is empty, and the matrix $H^{(i)}$ is chosen to build q_i linear combinations of the $p - r_d^{(i)}$ left nullspace basis vectors, such that $H^{(i)}Q_1^{(i)}(\lambda)$ has full row rank. If $q_i = p - r_d^{(i)}$ then the choice $H^{(i)} = I_{p-r_d^{(i)}}$ is used, otherwise $H^{(i)}$ is chosen a randomly generated $q_i \times \big(p - r_d^{(i)}\big)$ real matrix.

If `OPTIONS.minimal = true`, then $Q_2^{(i)}(\lambda)$ is a $q_i \times \big(p - r_d^{(i)}\big)$ transfer function matrix, with q_i chosen as above. $Q_2^{(i)}(\lambda)$ is determined in the form

$$Q_2^{(i)}(\lambda) = H^{(i)} + Y_2^{(i)}(\lambda)\,,$$

such that $Q_2^{(i)}(\lambda)Q_1^{(i)}(\lambda)$ $\big(= H^{(i)}N_l^{(i)}(\lambda) + Y_2^{(i)}(\lambda)N_l^{(i)}(\lambda)\big)$ and $Y_2^{(i)}(\lambda)$ are the least order solution of a left minimal cover problem [15]. If `OPTIONS.HDesign`$\{i\}$ is nonempty, then $H^{(i)} =$ `OPTIONS.HDesign`$\{i\}$, and if `OPTIONS.HDesign`$\{i\}$ is empty, then a suitable randomly generated $H^{(i)}$ is employed (see above).

The structure field `INFO.HDesign`$\{i\}$ contains the employed value of the design matrix $H^{(i)}$.

Computation of $Q_3^{(i)}(\lambda)$

$Q_3^{(i)}(\lambda)$ is a stable invertible transfer function matrix determined such that $Q^{(i)}(\lambda)$ in (229) has a desired dynamics (specified via `OPTIONS.sdeg` and `OPTIONS.poles`).

The resulting $N \times N$ matrix `INFO.MDperf` can be used for the assessment of the achieved distance mapping performance of the resulting model detection filters (see Section 3.6.3 for definitions). If `OPTIONS.MDFreq` is empty, then `INFO.MDperf`$(i,j) = \big\|\big[\, R_u^{(i,j)}(\lambda)\ R_d^{(i,j)}(\lambda)\,\big]\big\|_\infty$ if `OPTIONS.emdtest = true` and `INFO.MDperf`$(i,j) = \big\|R_u^{(i,j)}(\lambda)\big\|_\infty$ if `OPTIONS.emdtest = false`, and, ideally, represents a measure of the distance between the i-th and j-th component systems. If `OPTIONS.MDFreq` contains a set of n_f real frequency values ω_k, $k = 1,\ldots,n_f$ and λ_k, $k = 1,\ldots,n_f$ are the corresponding complex frequencies (see the description of `OPTIONS.MDFreq`), then `INFO.MDperf`$(i,j) = \max_k \big\|\big[\, R_u^{(i,j)}(\lambda_k)\ \ R_d^{(i,j)}(\lambda_k)\,\big]\big\|_\infty$ if `OPTIONS.emdtest = true` and `INFO.MDperf`$(i,j) = \max_k \big\|R_u^{(i,j)}(\lambda_k)\big\|_\infty$ if `OPTIONS.emdtest = false`. In this case, the entry `INFO.MDperf`(i,j) ideally represents a measure of the maximum distance between the frequency responses of the i-th and j-th component systems, evaluated in the selected set of frequency values. If `OPTIONS.normalize = true`, then for each row i, the filters `Q{i}` and `R{i,j}` are scaled such that the least value of `INFO.MDperf`(i,j) for $i \neq j$ is normalized to one. The standard normalization is performed if `OPTIONS.normalize = false`, in which case `INFO.MDperf`$(1,j) =$ `INFO.MDperf`$(j,1)$ for $j > 1$.

Example

Example 15. This is *Example* 6.1 from the book [16], which deals with a continuous-time state-space model, describing, in the fault-free case, the lateral dynamics of an F-16 aircraft with the matrices

$$A^{(1)} = \begin{bmatrix} -0.4492 & 0.046 & 0.0053 & -0.9926 \\ 0 & 0 & 1.0000 & 0.0067 \\ -50.8436 & 0 & -5.2184 & 0.7220 \\ 16.4148 & 0 & 0.0026 & -0.6627 \end{bmatrix}, \quad B_u^{(1)} = \begin{bmatrix} 0.0004 & 0.0011 \\ 0 & 0 \\ -1.4161 & 0.2621 \\ -0.0633 & -0.1205 \end{bmatrix},$$

$$C^{(1)} = I_4, \qquad D_u^{(1)} = 0_{4\times 2}\,.$$

The four state variables are the sideslip angle, roll angle, roll rate and yaw rate, and the two input variables are the aileron deflection and rudder deflection. The model detection problem addresses the synthesis of model detection filters for the detection and identification of loss of efficiency of the two flight actuators, which control the deflections of the aileron and rudder. The individual fault models correspond to different degrees of surface efficiency degradation. A multiple model with $N = 9$ component models is used, which correspond to a two-dimensional parameter grid for N values of the parameter vector $\rho := [\rho_1, \rho_2]^T$. For each component of ρ, we employ the three grid points $\{0, 0.5, 1\}$. The component system matrices in (226) are defined for $i = 1, 2, \ldots, N$ as: $E^{(i)} = I_4$, $A^{(i)} = A^{(1)}$, $C^{(i)} = C^{(1)}$, and $B_u^{(i)} = B_u^{(1)}\Gamma^{(i)}$, where $\Gamma^{(i)} = \text{diag}\left(1 - \rho_1^{(i)}, 1 - \rho_2^{(i)}\right)$ and $\left(\rho_1^{(i)}, \rho_2^{(i)}\right)$ are the values of parameters (ρ_1, ρ_2) on the chosen grid:

ρ_1 :	0	0	0	0.5	0.5	0.5	1	1	1
ρ_2 :	0	0.5	1	0	0.5	1	0	0.5	1

.

For example, $\left(\rho_1^{(1)}, \rho_2^{(1)}\right) = (0, 0)$ corresponds to the fault-free situation, while $\left(\rho_1^{(9)}, \rho_2^{(9)}\right) = (1, 1)$ corresponds to complete failure of both control surfaces. It follows, that the TFM $G_u^{(i)}(s)$ of the i-th system can be expressed as

$$G_u^{(i)}(s) = G_u^{(1)}(s)\Gamma^{(i)}, \tag{230}$$

where

$$G_u^{(1)}(s) = C^{(1)}\left(sI - A^{(1)}\right)^{-1}B_u^{(1)}$$

is the TFM of the fault-free system. Note that $G_u^{(N)}(s) = 0$ describes the case of complete failure.

The distances between the i-th and j-th models can be evaluated—for example, as the $\mathcal{H}_\infty$-norm of $G_u^{(i)}(s) - G_u^{(j)}(s)$, for $i, j = 1, \ldots, N$ and are plotted in Fig. 6.

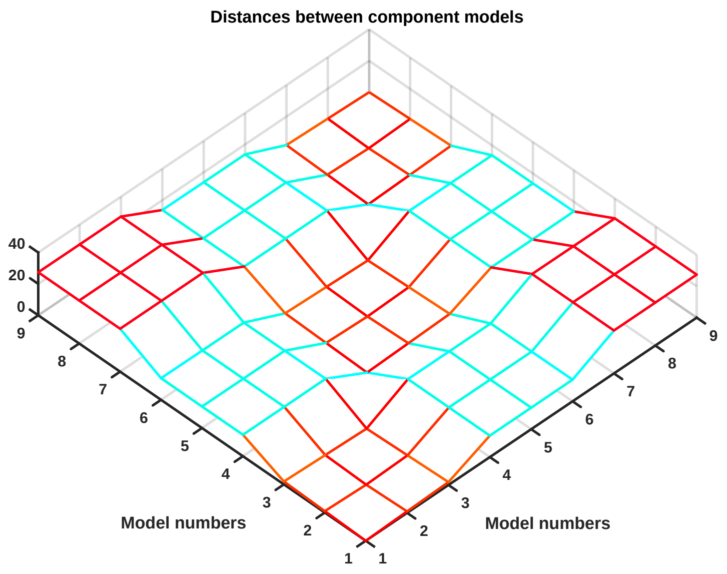

Figure 6: Distances between component models in terms of $\left\|G_u^{(i)}(s) - G_u^{(j)}(s)\right\|_\infty$

We aim to determine N filters $Q^{(i)}(s)$, $i = 1, \ldots, N$, with scalar outputs, having least McMillan degrees and satisfactory dynamics, which fulfill:

– the decoupling conditions: $R_u^{(i,i)}(s) = 0$, $i = 1, \ldots, N$;

– the model detectability condition: $R_u^{(i,j)}(s) \neq 0, \quad \forall j \neq i, \;\; i,j = 1, \ldots, N$.

Additionally, the resulting model detection performance measures $\|R_u^{(i,j)}(s)\|_\infty$ should (ideally) reproduce the shape of distances plotted in Fig. 6.

For the design of scalar filters, we used the same $1 \times p$ design matrix H for the synthesis of all filters, which has been chosen, after some trials with randomly generated values, as

$$H = [\, 0.7645 \;\; 0.8848 \;\; 0.5778 \;\; 0.9026 \,] \, .$$

The filter synthesis, performed by employing `emdsyn`, led to first order stable filters, which, as can be observed in Fig. 7, produces similar shapes of the model detection performance measure as those in Fig. 6.

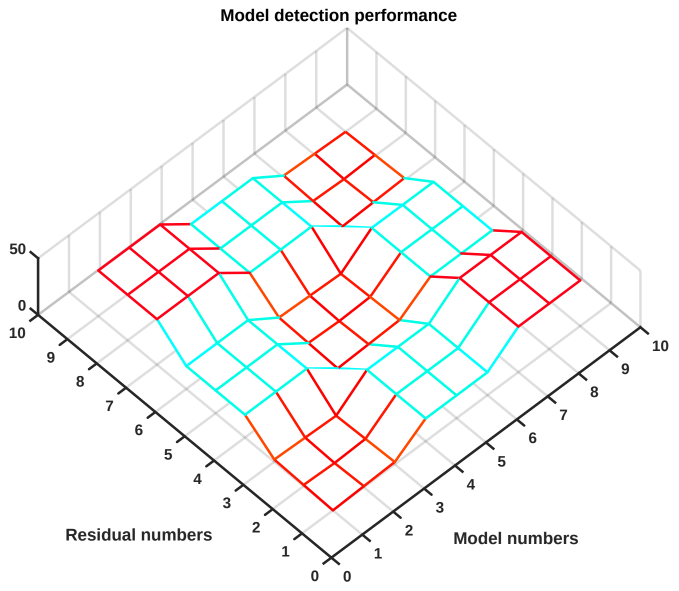

Figure 7: Model detection performance in terms of $\left\|R_u^{(i,j)}(s)\right\|_\infty$

In Fig. 8 the step responses from u_1 (aileron) and u_2 (rudder) are presented for the 9×9 block array, whose entries are the computed TFMs $R^{(i,j)}(s)$. Each column corresponds to a specific model for which the step responses of the N residuals are computed.

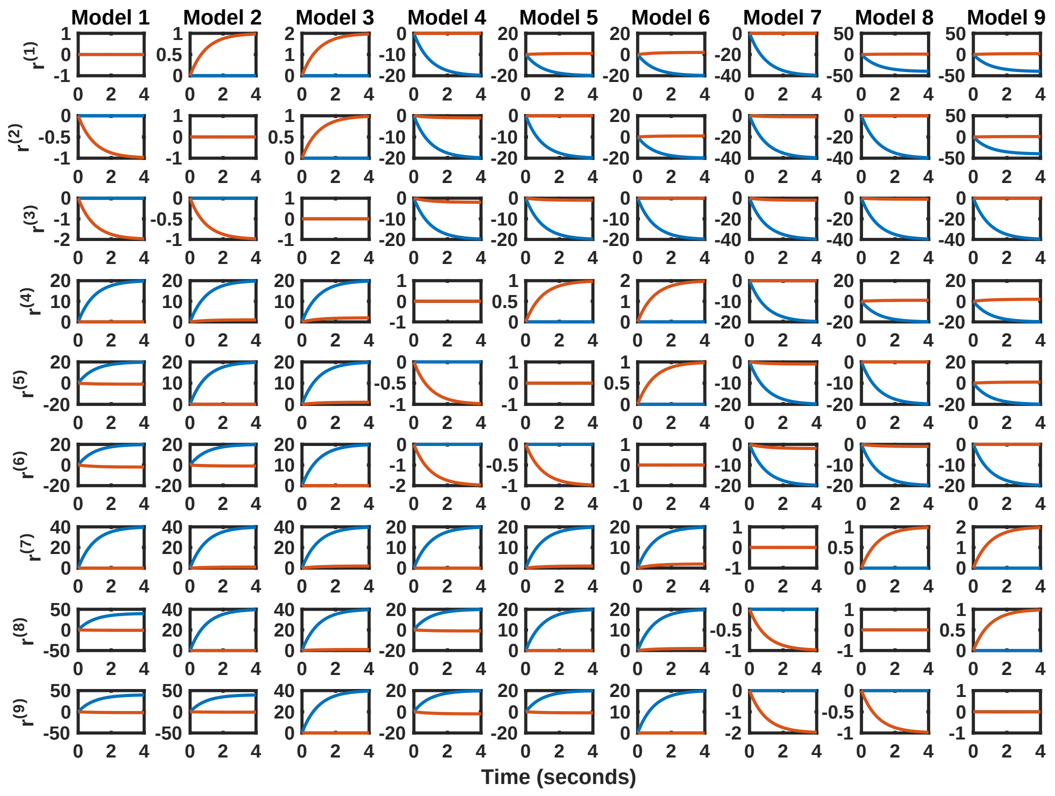

Figure 8: Step responses of $R^{(i,j)}(s)$ from u_1 (blue) and u_2 (red) for least order syntheses.

The following script implements the model building, filter synthesis and analysis steps.

```
% Example - Solution of an exact model detection problem (EMDP)

% Define a lateral aircraft dynamics model (without faults) with
% n  = 4 states
% mu = 2 control inputs
% p  = 4 measurable outputs
A = [-.4492 0.046 .0053 -.9926;
       0    0     1     0.0067;
   -50.8436 0   -5.2184  .722;
    16.4148 0     .0026 -.6627];
Bu = [0.0004 0.0011; 0 0; -1.4161 .2621; -0.0633 -0.1205];
C = eye(4); p = size(C,1); mu = size(Bu,2);
```

```
% define the loss of efficiency (LOE) faults as input scaling gains
% Gamma(i,:) = [ 1-rho1(i) 1-rho2(i) ]
Gamma = 1 - [ 0  0 0 .5 .5 .5 1  1 1;
              0 .5 1  0 .5  1 0 .5 1 ]';
N = size(Gamma,1);  % number of LOE cases

% define a multiple physical fault model Gui = Gu*diag(Gamma(i,:))
sysu = ss(zeros(p,mu,N,1));
for i=1:N
    sysu(:,:,i,1) = ss(A,Bu*diag(Gamma(i,:)),C,0);
end

% setup synthesis model
sysu = mdmodset(sysu,struct('controls',1:mu));

% nu-gap distance plots
nugapdist  = mddist(sysu);
figure, mesh(nugapdist), colormap hsv
title('\nu-gap distances between component models')
ylabel('Model numbers')
xlabel('Model numbers')

% H-infinity norm based distance plots
hinfdist  = mddist(sysu,struct('distance','Inf'));
figure, mesh(hinfdist), colormap hsv
title('H_\infty norm based distances between component models')
ylabel('Model numbers')
xlabel('Model numbers')


% call of EMDSYN with the options for stability degree -1 and pole -1 for
% the filters, tolerance and a design matrix H to form a linear combination
% of the left nullspace basis vectors
H = [ 0.7645   0.8848   0.5778   0.9026 ];
emdsyn_options = struct('sdeg',-1,'poles',-1,'HDesign',H);
[Q,R,info] = emdsyn(sysu,emdsyn_options);

% inspect achieved performance
figure, mesh(info.MDperf), colormap hsv
title('Distance mapping performance')
ylabel('Residual numbers')
xlabel('Model numbers')
```

```
% plot the step responses for the internal filter representations
figure
k1 = 0;
for j = 1:N,
  k1 = k1+1;
  k = k1;
  for i=1:N,
    subplot(N,N,k),
    [r,t] = step(R{j,i},4);
    plot(t,r(:,:,1),t,r(:,:,2)),
    if i == 1, title(['Model ',num2str(j)]), end
    if i == j, ylim([-1 1]), end
    if j == 1, ylabel(['r^(^', num2str(i),'^)'],'FontWeight','bold'), end
    if i == N && j == 5, xlabel('Time (seconds)','FontWeight','bold'), end
    k = k+N;
  end
end
```

4.9.2 `amdsyn`

Syntax

`[Q,R,INFO] = amdsyn(SYSM,OPTIONS)`

Description

`amdsyn` solves the *approximate model detection problem* (AMDP) (see Section 3.5.2), for a given stable LTI multiple model `SYSM` containing N models. A bank of N stable and proper residual generation filters $Q^{(i)}(\lambda)$, for $i = 1,\ldots,N$, is determined, in the form (45). For each filter $Q^{(i)}(\lambda)$, its associated internal forms $R^{(i,j)}(\lambda)$, for $j = 1,\ldots,N$, are determined in accordance with (47).

Input data

`SYSM` is a multiple model which contains N stable LTI systems in the state-space form

$$\begin{aligned} E^{(j)}\lambda x^{(j)}(t) &= A^{(j)}x^{(j)}(t) + B_u^{(j)}u^{(j)}(t) + B_d^{(j)}d^{(j)}(t) + B_w^{(j)}w^{(j)}(t)\,, \\ y^{(j)}(t) &= C^{(j)}x^{(j)}(t) + D_u^{(j)}u^{(j)}(t) + D_d^{(j)}d^{(j)}(t) + D_w^{(j)}w^{(j)}(t)\,, \end{aligned} \tag{231}$$

where $x^{(j)}(t) \in \mathbb{R}^{n^{(j)}}$ is the state vector of the j-th system with control input $u^{(j)}(t) \in \mathbb{R}^{m_u}$, disturbance input $d^{(j)}(t) \in \mathbb{R}^{m_d^{(j)}}$ and noise input $w^{(j)}(t) \in \mathbb{R}^{m_w^{(j)}}$, and where any of the inputs components $u^{(j)}(t)$, $d^{(j)}(t)$, or $w^{(j)}(t)$ can be void. The multiple model `SYSM` is either an array of N LTI systems of the form (231), in which case $m_d^{(j)} = m_d$ and $m_w^{(j)} = m_w$ for $j = 1,\ldots,N$, or is a $1\times N$ cell array, with `SYSM`$\{j\}$ containing the j-th component system in the form (231). The input groups for $u^{(j)}(t)$, $d^{(j)}(t)$, and $w^{(j)}(t)$ have the standard names `'controls'`, `'disturbances'`, and `'noise'`, respectively.

`OPTIONS` is a MATLAB structure used to specify various synthesis options and has the following fields:

`OPTIONS` **fields**	**Description**
`tol`	relative tolerance for rank computations (Default: internally computed)
`tolmin`	absolute tolerance for observability tests (Default: internally computed)
`MDTol`	threshold for model detectability checks (Default: 10^{-4})
`MDGainTol`	threshold for strong model detectability checks (Default: 10^{-2})
`rdim`	N-dimensional vector or a scalar; for a vector q, the i-th component q_i specifies the desired number of residual outputs for the i-th filter `Q`$\{i\}$; for a scalar value $\bar{q}$, a vector q with all N components $q_i = \bar{q}$ is assumed. (Default: `[ ]`, in which case: – if `OPTIONS.HDesign`$\{i\}$ is empty, then $q_i = 1$, if `OPTIONS.minimal` = `true`, or $q_i = n_v^{(i)}$, the dimension of the left nullspace which underlies the synthesis of `Q`$\{i\}$, if `OPTIONS.minimal` = `false` and $r_w^{(i)} = 0$ (see **Method**); $q_i = r_w^{(i)}$, if `OPTIONS.minimal` = `false` and $r_w^{(i)} > 0$ (see **Method**); – if `OPTIONS.HDesign`$\{i\}$ is nonempty, then q_i is the row dimension of the design matrix contained in `OPTIONS.HDesign`$\{i\}$.)
`emdtest`	option to perform extended model detectability tests using both control and disturbance input channels: `true` – use both control and disturbance input channels; `false` – use only the control channel (default).
`smarg`	prescribed stability margin for the resulting filters `Q`$\{i\}$ (Default: `-sqrt(eps)` for continuous-time component systems; `1-sqrt(eps)` for discrete-time component systems.
`sdeg`	prescribed stability degree for the resulting filters `Q`$\{i\}$ (Default: -0.05 for continuous-time component systems; 0.95 for discrete-time component systems.
`poles`	complex vector containing a complex conjugate set of desired poles (within the stability margin) to be assigned for the resulting filters `Q`$\{i\}$ (Default: `[ ]`)
`MDFreq`	real vector, which contains the frequency values ω_k, $k = 1, \ldots, n_f$, to be used for strong model detectability checks. For each real frequency ω_k, there corresponds a complex frequency λ_k which is used to evaluate the frequency-response gains. Depending on the system type, $\lambda_k = \mathrm{i}\omega_k$, in the continuous-time case, and $\lambda_k = \exp(\mathrm{i}\omega_k T)$, in the discrete-time case, where T is the common sampling time of the component systems. (Default: `[ ]`)

`nullspace`	option to use a specific type of proper nullspace bases: `true` – use minimal proper bases; `false` – use full-order observer based bases (default) *Note:* This option can only be used if no disturbance inputs are present in (231) and $\forall j$, $E^{(j)}$ is invertible.
`simple`	option to compute simple proper bases: `true` – compute simple bases; the orders of the basis vectors are provided in `INFO.degs`; `false` – no simple basis computed (default)
`minimal`	option to perform least order filter syntheses: `true` – perform least order syntheses (default); `false` – perform full order syntheses.
`tcond`	maximum allowed value for the condition numbers of the employed non-orthogonal transformation matrices (Default: 10^4) (only used if `OPTIONS.simple = true`)
`MDSelect`	integer vector with increasing elements containing the indices of the desired filters to be designed (Default: $[\,1,\ldots,N\,]$)
`HDesign`	N-dimensional cell array; `OPTIONS.HDesign`$\{i\}$, if not empty, is a full row rank design matrix employed for the synthesis of the i-th filter. If `OPTIONS.HDesign` is specified as a full row rank design matrix H, then an N-dimensional cell array is assumed with `OPTIONS.HDesign`$\{i\} = H$, for $i = 1,\ldots,N$. (Default: `[ ]`).
`epsreg`	regularization parameter (Default: 0.1)
`sdegzer`	prescribed stability degree for zeros shifting (Default: -0.05 for a continuous-time system `SYSF`; 0.95 for a discrete-time system `SYSF`).
`nonstd`	option to handle nonstandard optimization problems (see **Method**): 1 – use the quasi-co-outer–co-inner factorization (default); 2 – use the modified co-outer–co-inner factorization with the regularization parameter `OPTIONS.epsreg`; 3 – use the Wiener-Hopf type co-outer–co-inner factorization. 4 – use the Wiener-Hopf type co-outer-co-inner factorization with zero shifting of the non-minimum phase factor using the stabilization parameter `OPTIONS.sdegzer` 5 – use the Wiener-Hopf type co-outer-co-inner factorization with the regularization of the non-minimum phase factor using the regularization parameter `OPTIONS.epsreg`

Output data

`Q` is an $N \times 1$ cell array of filters, where `Q`$\{i\}$ contains the resulting i-th filter in a standard state-space representation

$$\begin{aligned}\lambda x_Q^{(i)}(t) &= A_Q^{(i)} x_Q^{(i)}(t) + B_{Q_y}^{(i)} y(t) + B_{Q_u}^{(i)} u(t),\\ r^{(i)}(t) &= C_Q^{(i)} x_Q^{(i)}(t) + D_{Q_y}^{(i)} y(t) + D_{Q_u}^{(i)} u(t),\end{aligned}$$

where the residual signal $r^{(i)}(t)$ is a q_i-dimensional vector, with q_i specified in `OPTIONS.rdim`. For each system object `Q`$\{i\}$, two input groups `'outputs'` and `'controls'` are defined for $y(t)$ and $u(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r^{(i)}(t)$. `Q`$\{i\}$ is empty for all i which do not belong to the index set specified by `OPTIONS.MDSelect`.

`R` is an $N \times N$ cell array of filters, where the (i,j)-th filter `R`$\{i,j\}$, is the internal form of `Q`$\{i\}$ acting on the j-th model. The resulting `R`$\{i,j\}$ has a standard state-space representation

$$\begin{aligned}\lambda x_R^{(i,j)}(t) &= A_R^{(i,j)} x_R^{(i,j)}(t) + B_{R_u}^{(i,j)} u^{(j)}(t) + B_{R_d}^{(i,j)} d^{(j)}(t) + B_{R_w}^{(i,j)} w^{(j)}(t),\\ r^{(i,j)}(t) &= C_R^{(i,j)} x_R^{(i,j)}(t) + D_{R_u}^{(i,j)} u^{(j)}(t) + D_{R_d}^{(i,j)} d^{(j)}(t) + D_{R_w}^{(i,j)} w^{(j)}(t)\end{aligned}$$

and the input groups `'controls'`, `'disturbances'` and `'noise'` are defined for $u^{(j)}(t)$, $d^{(j)}(t)$, and $w^{(j)}(t)$, respectively, and the output group `'residuals'` is defined for the residual signal $r^{(i,j)}(t)$. `R`$\{i,j\}$, $j = 1,\ldots,N$ are empty for all i which do not belong to the index set specified by `OPTIONS.MDSelect`.

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

`INFO` **fields**	**Description**
`tcond`	N-dimensional vector; `INFO.tcond`(i) contains the maximum of the condition numbers of the non-orthogonal transformation matrices used to determine the i-th filter `Q`$\{i\}$; a warning is issued if any `INFO.tcond`(i) $\geq$ `OPTIONS.tcond`.
`degs`	N-dimensional cell array; if `OPTIONS.simple` = `true`, then a nonempty `INFO.degs`$\{i\}$ contains the orders of the basis vectors of the employed simple nullspace basis for the synthesis of the i-th filter component `Q`$\{i\}$; if `OPTIONS.simple` = `false`, then a nonempty `INFO.degs`$\{i\}$ contains the degrees of the basis vectors of an equivalent polynomial nullspace basis; `INFO.degs`$\{i\}$ `= [ ]` for all i which do not belong to the index set specified by `OPTIONS.MDSelect`.
`MDperf`	$N \times N$-dimensional array containing the achieved model detection performance measure, given as the gains associated with the internal representations (see **Method**). `INFO.MDperf`$(i,j) = -1$, for $j = 1,\ldots,N$ and for all i which do not belong to the index set specified by `OPTIONS.MDSelect`.

`HDesign`	N-dimensional cell array, where `INFO.HDesign`$\{i\}$ contains the i-th design matrix $H^{(i)}$ employed for the synthesis of the i-th filter (see **Method**)
`MDgap`	N-dimensional vector, which contains the achieved noise gaps. `INFO.MDgap`(i) contains the i-th gap η_i achieved by the i-th filter (see **Method**).

Method

An extension of the **Procedure AMD** from [16, Sect. 6.3] is implemented, which relies on the nullspace-based synthesis method proposed in [10]. Assume that the j-th model has the input-output form

$$\mathbf{y}^{(j)}(\lambda) = G_u^{(j)}(\lambda)\mathbf{u}^{(j)}(\lambda) + G_d^{(j)}(\lambda)\mathbf{d}^{(j)}(\lambda) + G_w^{(j)}(\lambda)\mathbf{w}^{(j)}(\lambda) \tag{232}$$

and the resulting i-th filter $Q^{(i)}(\lambda)$ has the input-output form

$$\mathbf{r}^{(i)}(\lambda) = Q^{(i)}(\lambda)\left[\begin{array}{c} \mathbf{y}(\lambda) \\ \mathbf{u}(\lambda) \end{array}\right]. \tag{233}$$

The synthesis method, which underlies **Procedure AMD**, essentially determines each filter $Q^{(i)}(\lambda)$ as a stable rational left annihilator of

$$G^{(i)}(\lambda) := \left[\begin{array}{cc} G_u^{(i)}(\lambda) & G_d^{(i)}(\lambda) \\ I_{m_u} & 0 \end{array}\right],$$

such that for $i \neq j$ we have $[\, R_u^{(i,j)}(\lambda)\; R_d^{(i,j)}(\lambda)\,] \neq 0$, where

$$R^{(i,j)}(\lambda) := [\, R_u^{(i,j)}(\lambda)\; R_d^{(i,j)}(\lambda)\; R_w^{(i,j)}(\lambda)\,] = Q^{(i)}(\lambda)\left[\begin{array}{ccc} G_u^{(j)}(\lambda) & G_d^{(j)}(\lambda) & G_w^{(j)}(\lambda) \\ I_{m_u} & 0 & 0 \end{array}\right]$$

is the internal form of $Q^{(i)}(\lambda)$ with respect to the j-th model. Additionally, the gap η_i achieved by the i-th filter is maximized.

The resulting $N \times N$ matrix `INFO.MDperf` can be used for the assessment of the achieved distance mapping performance of the resulting model detection filters (see Section 3.6.3 for definitions). If `OPTIONS.MDFreq` is empty, then `INFO.MDperf`$(i,j) = \big\|[\, R_u^{(i,j)}(\lambda)\; R_d^{(i,j)}(\lambda)\,]\big\|_\infty$ if `OPTIONS.emdtest = true` and `INFO.MDperf`$(i,j) = \big\|R_u^{(i,j)}(\lambda)\big\|_\infty$ if `OPTIONS.emdtest = false`, and, ideally, represents a measure of the distance between the i-th and j-th component systems. If `OPTIONS.MDFreq` contains a set of n_f real frequency values ω_k, $k = 1,\ldots,n_f$ and λ_k, $k = 1,\ldots,n_f$ are the corresponding complex frequencies (see the description of `OPTIONS.MDFreq`), then `INFO.MDperf`$(i,j) = \max_k \big\|[\, R_u^{(i,j)}(\lambda_k)\;\; R_d^{(i,j)}(\lambda_k)\,]\big\|_\infty$ if `OPTIONS.emdtest = true` and `INFO.MDperf`$(i,j) = \max_k \big\|R_u^{(i,j)}(\lambda_k)\big\|_\infty$ if `OPTIONS.emdtest = false`. In this case, the entry `INFO.MDperf`(i,j) ideally represents a measure of the maximum distance between the frequency responses of the i-th and j-th component systems, evaluated in the selected set of frequency values. If `OPTIONS.normalize = true`, then for each row i, the filters `Q{i}` and `R{i,j}` are scaled such that the least value of `INFO.MDperf`(i,j) for $i \neq j$ is normalized to one. The standard normalization is performed if `OPTIONS.normalize = false`, in which case `INFO.MDperf`$(1,j) =$ `INFO.MDperf`$(j,1)$ for $j > 1$.

The N-dimensional vector `INFO.MDgap`, contains the resulting noise gaps η_i, for $i = 1, \ldots, N$ (see Section 3.6.5 for definitions). If `OPTIONS.MDFreq` is empty, then η_i is evaluated as

$$\eta_i := \min_{j \neq i} \big\| \big[\, R_u^{(i,j)}(\lambda)\; R_d^{(i,j)}(\lambda) \,\big] \big\|_\infty \big/ \big\| R_w^{(i,i)}(\lambda) \big\|_\infty, \tag{234}$$

if `OPTIONS.emdtest = true` and

$$\eta_i := \min_{j \neq i} \big\| R_u^{(i,j)}(\lambda) \big\|_\infty \big/ \big\| R_w^{(i,i)}(\lambda) \big\|_\infty, \tag{235}$$

if `OPTIONS.emdtest = false`. If `OPTIONS.MDFreq` contains a set of n_f real frequency values ω_k, $k = 1, \ldots, n_f$ and λ_k, $k = 1, \ldots, n_f$ are the corresponding complex frequencies (see the description of `OPTIONS.MDFreq`), then

$$\eta_i := \min_{j \neq i} \max_k \big\| \big[\, R_u^{(i,j)}(\lambda_k)\; R_d^{(i,j)}(\lambda_k) \,\big] \big\|_2 \big/ \big\| R_w^{(i,i)}(\lambda) \big\|_\infty, \tag{236}$$

if `OPTIONS.emdtest = true` and

$$\eta_i := \min_{j \neq i} \max_k \big\| R_u^{(i,j)}(\lambda_k) \big\|_2 \big/ \big\| R_w^{(i,i)}(\lambda) \big\|_\infty, \tag{237}$$

if `OPTIONS.emdtest = false`.

Each filter $Q^{(i)}(\lambda)$ is determined in the product form

$$Q^{(i)}(\lambda) = Q_4^{(i)}(\lambda) Q_3^{(i)}(\lambda) Q_2^{(i)}(\lambda) Q_1^{(i)}(\lambda), \tag{238}$$

where the factors are determined as follows:

(a) $Q_1^{(i)}(\lambda) = N_l^{(i)}(\lambda)$, with $N_l^{(i)}(\lambda)$ a $\big(p - r_d^{(i)}\big) \times (p + m_u)$ proper rational left nullspace basis satisfying $N_l^{(i)}(\lambda) G^{(i)}(\lambda) = 0$, with $r_d^{(i)} := \operatorname{rank} G_d^{(i)}(\lambda)$; ($n_v^{(i)} := p - r_d^{(i)}$ is the number of basis vectors)

(b) $Q_2^{(i)}(\lambda)$ is an admissible regularization factor guaranteeing model detectability;

(c) $Q_3^{(i)}(\lambda)$ represents an optimal choice to maximize the i-th gap η_i in (234) – (237);

(d) $Q_4^{(i)}(\lambda)$ is a stable invertible factor determined such that $Q^{(i)}(\lambda)$ has a desired dynamics.

The computations of individual factors depend on the user's options and the optimization problem features. Specific choices are discussed in what follows.

Computation of $Q_1^{(i)}(\lambda)$

If `OPTIONS.nullspace = true`, then the left nullspace basis $N_l^{(i)}(\lambda)$ is determined as a stable minimal proper rational basis, or, if `OPTIONS.nullspace = false` (the default option) and $m_d^{(i)} = 0$, then the simple observer based basis $N_l^{(i)}(\lambda) = [\, I \;\; -G_u^{(i)}(\lambda)\,]$ is employed. If $N_l^{(i)}(\lambda)$ is a minimal rational basis and if `OPTIONS.simple = true`, then $N_l^{(i)}(\lambda)$ is determined as a simple rational basis and the orders of the basis vectors are provided in `INFO.degs`. These are also the degrees of the basis vectors of an equivalent polynomial nullspace basis. The dynamics of $Q_1^{(i)}(\lambda)$ is specified via `OPTIONS.sdeg` and `OPTIONS.poles`.

Computation of $Q_2^{(i)}(\lambda)$

Let $r_w^{(i)}$ be the rank of $\overline{G}_w^{(i,i)}(\lambda) := Q_1^{(i)}(\lambda)\left[\begin{smallmatrix} G_w^{(i)}(\lambda) \\ 0 \end{smallmatrix}\right]$ and let q_i be the desired number of residual outputs for the i-th filter. If `OPTIONS.rdim` is nonempty, then $q_i =$ `OPTIONS.rdim`$\{i\}$, while if `OPTIONS.rdim` is empty a default value of q_i is defined depending on the setting of `OPTIONS.HDesign`$\{i\}$ and `OPTIONS.minimal`. If `OPTIONS.HDesign`$\{i\}$ is nonempty, then q_i is the row dimension of the design matrix contained in `OPTIONS.HDesign`$\{i\}$. If `OPTIONS.HDesign`$\{i\}$ is empty, then $q_i = 1$ if `OPTIONS.minimal = true`. If `OPTIONS.minimal = false` and $r_w^{(i)} = 0$, then $q_i = p - r_d^{(i)}$, while if $r_w^{(i)} > 0$, then $q_i = r_w^{(i)}$.

If `OPTIONS.minimal = false`, then $Q_2^{(i)}(\lambda) = M^{(i)}(\lambda)H^{(i)}$, where $H^{(i)}$ is a suitable $q_i \times \big(p - r_d^{(i)}\big)$ full row rank design matrix and $M^{(i)}(\lambda)$ is a stable invertible transfer function matrix determined such that $Q_2^{(i)}(\lambda)Q_1^{(i)}(\lambda)$ has a desired dynamics (specified via `OPTIONS.sdeg` and `OPTIONS.poles`). $H^{(i)}$ is set as follows. If `OPTIONS.HDesign`$\{i\}$ is nonempty, then $H^{(i)} =$ `OPTIONS.HDesign`$\{i\}$. If `OPTIONS.HDesign`$\{i\}$ is empty, the matrix $H^{(i)}$ is chosen to build q_i linear combinations of the $p - r_d^{(i)}$ left nullspace basis vectors, such that $H^{(i)}Q_1^{(i)}(\lambda)$ has full row rank. If $q_i = p - r_d^{(i)}$ then the choice $H^{(i)} = I_{p-r_d^{(i)}}$ is used, otherwise $H^{(i)}$ is chosen a randomly generated $q_i \times \big(p - r_d^{(i)}\big)$ real matrix.

If `OPTIONS.minimal = true`, then $Q_2^{(i)}(\lambda)$ is a $q_i \times \big(p - r_d^{(i)}\big)$ transfer function matrix, with q_i chosen as above. $Q_2^{(i)}(\lambda)$ is determined in the form

$$Q_2^{(i)}(\lambda) = M^{(i)}(\lambda)\widetilde{Q}_2^{(i)}(\lambda)\,,$$

where $\widetilde{Q}_2^{(i)}(\lambda) := H^{(i)} + Y_2^{(i)}(\lambda)$, $\widetilde{Q}_2^{(i)}(\lambda)Q_1^{(i)}(\lambda)$ $\big(= H^{(i)}N_l^{(i)}(\lambda) + Y_2^{(i)}(\lambda)N_l^{(i)}(\lambda)\big)$ and $Y_2^{(i)}(\lambda)$ are the least order solution of a left minimal cover problem [15], and $M^{(i)}(\lambda)$, a stable invertible transfer function matrix determined such that $M^{(i)}(\lambda)\widetilde{Q}_2^{(i)}(\lambda)Q_1^{(i)}(\lambda)$ has a desired dynamics. If `OPTIONS.HDesign`$\{i\}$ is nonempty, then $H^{(i)} =$ `OPTIONS.HDesign`$\{i\}$, and if `OPTIONS.HDesign`$\{i\}$ is empty, then a suitable randomly generated $H^{(i)}$ is employed (see above).

Note: The stabilization with $M^{(i)}(\lambda)$ is only performed if $r_w^{(i)} = 0$ (i.e., to also cover the case of an exact synthesis).

The structure field `INFO.HDesign`$\{i\}$ contains the employed value of the design matrix $H^{(i)}$.

Computation of $Q_3^{(i)}(\lambda)$

Let define $\widetilde{G}_w^{(i,i)}(\lambda) := Q_2^{(i)}(\lambda)\overline{G}_w^{(i,i)}(\lambda)$ and let $\tilde{r}_w^{(i)} = \operatorname{rank} \widetilde{G}_w^{(i,i)}(\lambda)$, which satisfies $\tilde{r}_w^{(i)} \leq q_i$. If $\tilde{r}_w^{(i)} = 0$ then $Q_3^{(i)}(\lambda) = I$. If $\tilde{r}_w^{(i)} > 0$, then the quasi-co-outer–co-inner factorization of $\widetilde{G}_w^{(i,i)}(\lambda)$ is computed as

$$\widetilde{G}_w^{(i,i)}(\lambda) = R_{wo}^{(i)}(\lambda)R_{wi}^{(i)}(\lambda),$$

where $R_{wo}^{(i)}(\lambda)$ is a (full column rank) quasi-co-outer factor and $R_{wi}^{(i)}(\lambda)$ is a (full row rank) co-inner factor. In the *standard case* $R_{wo}^{(i)}(\lambda)$ is outer (i.e., has no zeros on the boundary of the stability domain $\partial\mathbb{C}_s$) and thus, there exists a stable left inverse $\big(R_{wo}^{(i)}(\lambda)\big)^{-L}$ such that $\big(R_{wo}^{(i)}(\lambda)\big)^{-L}R_{wo}^{(i)}(\lambda) = I$. In this case, we choose $Q_3^{(i)}(\lambda) = \big(R_{wo}^{(i)}(\lambda)\big)^{-L}$. This is an optimal choice which ensures that the maximal gap η_i is achieved by the i-th filter (see below). If $\tilde{r}_w = q_i$ (the usual case), then $Q_3^{(i)}(\lambda)$ is simply $Q_3^{(i)}(\lambda) = \big(R_{wo}^{(i)}(\lambda)\big)^{-1}$.

In the *non-standard case* $R_{wo}^{(i)}(\lambda)$ is only quasi-outer and thus, has zeros on the boundary of the stability domain $\partial\mathbb{C}_s$. Depending on the selected option to handle nonstandard optimization problems `OPTIONS.nonstd`, several choices are possible for $Q_3^{(i)}(\lambda)$ in this case:

- If `OPTIONS.nonstd` $= 1$, then $Q_3^{(i)}(\lambda) = \big(R_{wo}^{(i)}(\lambda)\big)^{-L}$ is used, where all spurious poles of the left inverse are assigned to values specified via `OPTIONS.sdeg` and `OPTIONS.poles`.

- If `OPTIONS.nonstd` $= 2$, then a modified co-outer–co-inner factorization of $[\, R_{wo}^{(i)}(\lambda)\ \epsilon I\,]$ is computed, whose co-outer factor $R_{wo,\epsilon}^{(i)}(\lambda)$ satisfies

  $$R_{wo,\epsilon}^{(i)}(\lambda)\big(R_{wo,\epsilon}^{(i)}(\lambda)\big)^{\sim} = \epsilon^2 I + R_{wo}^{(i)}(\lambda)\big(R_{wo}^{(i)}(\lambda)\big)^{\sim}.$$

  Then $Q_3^{(i)}(\lambda) = \big(R_{wo,\epsilon}^{(i)}(\lambda)\big)^{-L}$ is used. The value of the regularization parameter ϵ is specified via `OPTIONS.epsreg`.

- If `OPTIONS.nonstd` $= 3$, then a Wiener-Hopf type co-outer–co-inner factorization is computed in the form

  $$\widetilde{G}_w^{(i,i)}(\lambda) = R_{wo}^{(i)}(\lambda)R_{wb}^{(i)}(\lambda)R_{wi}^{(i)}(\lambda), \qquad (239)$$

  where $R_{wo}^{(i)}(\lambda)$ is co-outer, $R_{wi}^{(i)}(\lambda)$ is co-inner, and $R_{wb}^{(i)}(\lambda)$ is a square stable factor whose zeros are precisely the zeros of $\widetilde{G}_w^{(i,i)}(\lambda)$ in $\partial\mathbb{C}_s$. $Q_3^{(i)}(\lambda)$ is determined as before $Q_3^{(i)}(\lambda) = \big(R_{wo}^{(i)}(\lambda)\big)^{-L}$.

- If `OPTIONS.nonstd` $= 4$, then the Wiener-Hopf type co-outer–co-inner factorization (239) is computed and $Q_3^{(i)}(\lambda)$ is determined as $Q_3^{(i)}(\lambda) = \big(R_{wb}^{(i)}(\tilde{\lambda})\big)^{-1}\big(R_{wo}^{(i)}(\lambda)\big)^{-1}$, where $\tilde{\lambda}$ is a small perturbation of λ to move all zeros of $R_{wb}^{(i)}(\lambda)$ into the stable domain. In the continuous-time case $\tilde{s} = \frac{s-\beta_z}{1-\beta_z s}$, while in the discrete-time case $\tilde{z} = z/\beta_z$, where the zero shifting parameter β_z is the prescribed stability degree for the zeros specified in `OPTIONS.sdegzer`. For the evaluation of $R_{wb}^{(i)}(\tilde{\lambda})$, a suitable bilinear transformation is performed.

- If `OPTIONS.nonstd` $= 5$, then the Wiener-Hopf type co-outer–co-inner factorization (239) is computed and $Q_3^{(i)}(\lambda)$ is determined as $Q_3^{(i)}(\lambda) = \big(R_{wb,\epsilon}^{(i)}(\lambda)\big)^{-1}\big(R_{wo}^{(i)}(\lambda)\big)^{-1}$, where $R_{wb,\epsilon}^{(i)}(\lambda)$ is the co-outer factor of the co-outer–co-inner factorization of $\big[\, R_{wb}^{(i)}(\lambda)\ \epsilon I \,\big]$ and satisfies

$$R_{wb,\epsilon}^{(i)}(\lambda)\big(R_{wb,\epsilon}^{(i)}(\lambda)\big)^{\sim} = \epsilon^2 I + R_{wb}^{(i)}(\lambda)\big(R_{wb}^{(i)}(\lambda)\big)^{\sim}.$$

  The value of the regularization parameter ϵ is specified via `OPTIONS.epsreg`.

A typical feature of the non-standard case is that, with the exception of using the option `OPTIONS.nonstd` $= 3$, all other choices of `OPTIONS.nonstd` lead to a poor dynamical performance of the resulting filter, albeit arbitrary large noise gaps η_i can be occasionally achieved.

Computation of $Q_4^{(i)}(\lambda)$

In the standard case, $Q_4^{(i)}(\lambda) = I$. In the non-standard case, $Q_4^{(i)}(\lambda)$ is a stable invertible transfer function matrix determined such that $Q^{(i)}(\lambda)$ in (238) has a desired dynamics (specified via `OPTIONS.sdeg` and `OPTIONS.poles`).

Example

Example 16. This is Example 6.2 from the book [16], which deals with a continuous-time state-space model, describing, in the fault-free case, the lateral dynamics of an F-16 aircraft with the matrices

$$A^{(1)} = \begin{bmatrix} -0.4492 & 0.046 & 0.0053 & -0.9926 \\ 0 & 0 & 1.0000 & 0.0067 \\ -50.8436 & 0 & -5.2184 & 0.7220 \\ 16.4148 & 0 & 0.0026 & -0.6627 \end{bmatrix}, \quad B_u^{(1)} = \begin{bmatrix} 0.0004 & 0.0011 \\ 0 & 0 \\ -1.4161 & 0.2621 \\ -0.0633 & -0.1205 \end{bmatrix}, \quad B_w^{(1)} = [\, I_4\ 0_{4\times 2} \,]\,,$$

$$C^{(1)} = \begin{bmatrix} 57.2958 & 0 & 0 & 0 \\ 0 & 57.2958 & 0 & 0 \end{bmatrix}, \quad D_u^{(1)} = 0_{2\times 2}, \quad D_w^{(1)} = [\, 0_{2\times 4}\ \ I_2 \,]\,.$$

The four state variables are the sideslip angle, roll angle, roll rate and yaw rate, and the two input variables are the aileron deflection and rudder deflection. The two measured outputs are the sideslip angle and roll angle, and, additionally input noise and output noise are included in the model. The component system matrices in (231) are defined for $i = 1, 2, \ldots, N$ as: $E^{(i)} = I_4$, $A^{(i)} = A^{(1)}$, $C^{(i)} = C^{(1)}$, $B_w^{(i)} = B_w^{(1)}$, $D_w^{(i)} = D_w^{(1)}$, and $B_u^{(i)} = B_u^{(1)}\Gamma^{(i)}$, where $\Gamma^{(i)} = \mathrm{diag}\big(1-\rho_1^{(i)}, 1-\rho_2^{(i)}\big)$ and $\big(\rho_1^{(i)}, \rho_2^{(i)}\big)$ are the values of parameters (ρ_1, ρ_2) on the chosen grid points:

ρ_1 :	0	0	0	0.5	0.5	0.5	1	1	1
ρ_2 :	0	0.5	1	0	0.5	1	0	0.5	1

.

The TFMs $G_u^{(i)}(s)$ and $G_w^{(i)}(s)$ of the i-th system can be expressed as

$$G_u^{(i)}(s) = G_u^{(1)}(s)\Gamma^{(i)}, \quad G_w^{(i)}(s) = G_w^{(1)}(s)\,, \tag{240}$$

where

$$G_u^{(1)}(s) = C^{(1)}\big(sI - A^{(1)}\big)^{-1}B_u^{(1)}, \quad G_w^{(1)}(s) = C^{(1)}\big(sI - A^{(1)}\big)^{-1}B_w^{(1)} + D_w^{(1)}\,.$$

The individual fault models correspond to different degrees of surface efficiency degradation. The values $\big(\rho_1^{(1)}, \rho_2^{(1)}\big) = (0,0)$ correspond to the fault-free situation, while $G_u^{(N)}(s) = 0$ describes the case of complete failure.

The approximate model detection problem addresses the synthesis of model detection filters for the detection and identification of loss of efficiency of the two flight actuators, which control the deflections of the aileron and rudder, in the presence of noise inputs. For the design of the model detection system, we aim to determine the N filters $Q^{(i)}(s)$, $i = 1, \ldots, N$ having satisfactory dynamic responses and exhibiting the maximally achievable gaps.

In Fig. 9 the time responses of the residual evaluation signals $\theta_i(t)$ are presented, where $\theta_i(t)$ are computed using a Narendra-type evaluation filter [3] with input $\|r^{(i)}(t)\|_2^2$ and parameters $\alpha = 0.9$, $\beta = 0.1$, $\gamma = 10$ (see also Remark 3.13 in [16]). The control inputs have been chosen as follows: $u_1(t)$ is a step of amplitude 0.3 added to a square wave of period 2π, and $u_2(t)$ is a step of amplitude 1.5 added to a sinus function of unity amplitude and period π. The noise inputs are zero mean white noise of amplitude 0.01 for the input noise and 0.03 for the measurement noise. Each column corresponds to a specific model for which the time responses of the N residual evaluation signals are computed. The achieved typical structure matrix for model detection (with zeros down the diagonal) can easily be read out from this signal based assessment, even in presence of noise.

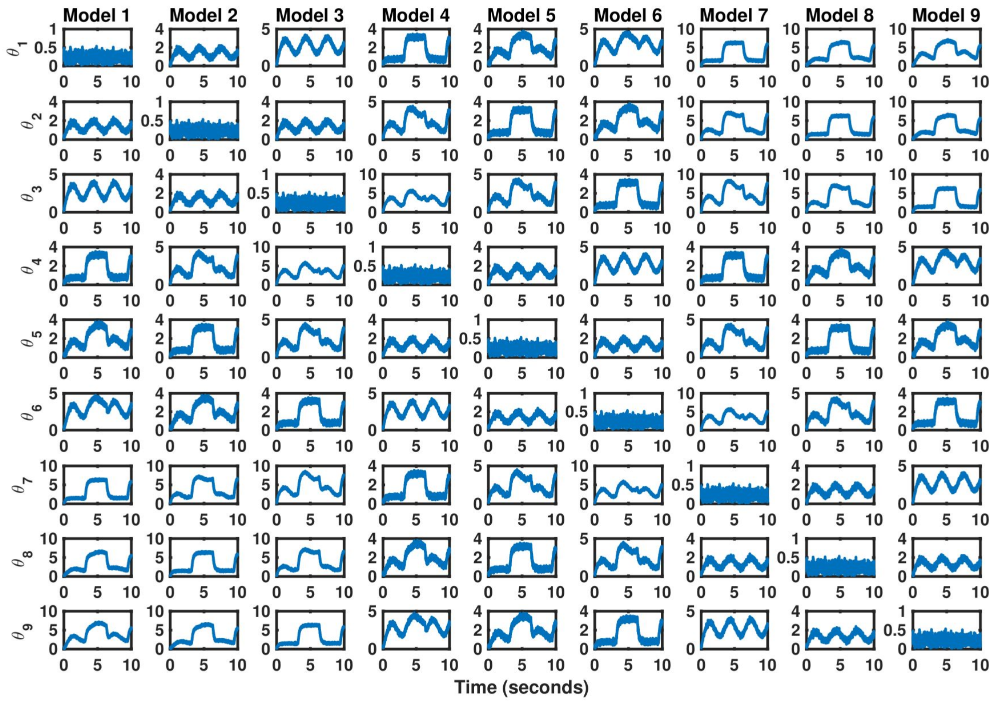

Figure 9: Time responses of evaluation signals for optimal syntheses

The following script implements the model building, synthesis and analysis steps.

```
% Example - Solution of an approximate model detection problem (AMDP)

% Define a lateral aircraft dynamics model (without faults) with
% n  = 4 states
% mu = 2 control inputs
% p  = 2 measurable outputs
% mw = 6 noise inputs

A = [-.4492 .046  .0053 -.9926;
       0    0     1      .0067;
   -50.8436 0   -5.2184  .722;
    16.4148 0     .0026 -.6627];
Bu = [.0004 .0011; 0 0; -1.4161 .2621; -.0633 -.1205];
[n,mu] = size(Bu); p = 2; mw = n+p; m = mu+mw;
Bw = eye(n,mw);
C = 180/pi*eye(p,n);  Du = zeros(p,mu); Dw = [zeros(p,n) eye(p)];

% define the loss of efficiency (LOE) faults as input scaling gains
% Gamma(i,:) = [ 1-rho1(i) 1-rho2(i) ]
Gamma = 1 - [ 0  0 0 .5 .5 .5 1  1 1;
              0 .5 1  0 .5  1 0 .5 1 ]';
N = size(Gamma,1);  % number of LOE cases

% define a multiple physical fault model with Gui = Gu*diag(Gamma(i,:)), Gwi = Gw
sysuw = ss(zeros(p,mu+mw,N,1));
for j = 1:N
    sysuw(:,:,j,1) = ss(A,[Bu*diag(Gamma(j,:)) Bw],C,[Du Dw]);
end

% setup synthesis model
sysuw = mdmodset(sysuw,struct('controls',1:mu,'noise',mu+(1:mw)));

% use nonminimal design with  AMDSYN with stability degree -1 and poles  at -1
opt_amdsyn = struct('sdeg',-1,'poles',-1,'minimal',false);
[Q,R,info] = amdsyn(sysuw,opt_amdsyn);
info.MDperf, info.MDgap

% inspect achieved performance
figure, mesh(info.MDperf), colormap hsv
title('Model detection performance')
ylabel('Residual numbers')
xlabel('Model numbers')
```

```
% generate input signals for Ex. 6.2
d = diag([ 1 1 0.01 0.01 0.01 0.01 0.03 0.03]);
t = (0:0.01:10)';  ns = length(t);
usin = gensig('sin',pi,t(end),0.01)+1.5;
usquare = gensig('square',pi*2,t(end),0.01)+0.3;
u = [ usquare usin (rand(ns,mw)-0.5)]*d;

% plot the step responses for the internal filter representations
figure
k1 = 0; alpha = 0.9; beta = 0.1; gamma = 10;
for j = 1:N,
  k1 = k1+1; k = k1;
  for i=1:N,
    subplot(N,N,k),
    [r,t] = lsim(R{j,i},u,t);
    % use a Narendra filter with (alpha,beta,gamma) = (0.9,0.1,10)
    theta = alpha*sqrt(r(:,1).^2+r(:,2).^2)+...
        beta*sqrt(lsim(tf(1,[1 gamma]),r(:,1).^2+r(:,2).^2,t));
    plot(t,theta),
    if i == 1, title(['Model ',num2str(j)]), end
    if i == j, ylim([0 1]), end
    if j == 1, ylabel(['\theta_', num2str(i)],'FontWeight','bold'), end
    if i == N && j == 5, xlabel('Time (seconds)','FontWeight','bold'), end
    k = k+N;
  end
end
```

References

[1] K. Glover and A. Varga. On solving non-standard $\mathcal{H}_-/\mathcal{H}_{2/\infty}$ fault detection problems. In *Proceedings of the IEEE Conference on Decision and Control, Orlando, FL, USA*, 2011, pages 891–896.

[2] MathWorks. *Control System Toolbox (R2015b), User's Guide.* The MathWorks Inc., Natick, MA, 2015.

[3] K. S. Narendra and J. Balakrishnan. Adaptive control using multiple models. *IEEE Trans. Automat. Control*, 42:171–187, 1997.

[4] A. Varga. DSTOOLS – The Descriptor System Tools for MATLAB. `https://sites.google.com/site/andreasvargacontact/home/software/dstools`.

[5] A. Varga. On computing least order fault detectors using rational nullspace bases. In *Proceedings of the IFAC Symposium SAFEPROCESS, Washington D.C., USA*, 2003.

[6] A. Varga. New computational approach for the design of fault detection and isolation filters. In *Advances in Automatic Control*, M. Voicu, editor, volume 754 of *The Kluwer International Series in Engineering and Computer Science*, Kluwer Academic Publishers, Dordrecht, 2004, pages 367–381.

[7] A. Varga. On designing least order residual generators for fault detection and isolation. In *Proceedings of the 16th International Conference on Control Systems and Computer Science, Bucharest, Romania*, 2007, pages 323–330.

[8] A. Varga. On computing nullspace bases – a fault detection perspective. In *Proceedings of the IFAC World Congress, Seoul, Korea*, 2008, pages 6295–6300.

[9] A. Varga. General computational approach for optimal fault detection. In *Proceedings of the IFAC Symposium SAFEPROCESS, Barcelona, Spain*, 2009, pages 107–112.

[10] A. Varga. Least order fault and model detection using multi-models. In *Proceedings of the Conference on Decision and Control, Shanghai, China*, 2009, pages 1014–1019.

[11] A. Varga. On computing achievable fault signatures. In *Proceedings of the IFAC Symposium SAFEPROCESS, Barcelona, Spain*, 2009, pages 935–940.

[12] A. Varga. Integrated computational algorithm for solving $\mathcal{H}_\infty$-optimal FDI problems. In *Proceedings of the IFAC World Congress, Milano, Italy*, 2011, pages 10187–10192.

[13] A. Varga. Descriptor system techniques in solving $\mathcal{H}_{2/\infty}$-optimal fault detection and isolation problems. In *Control and Optimization with Differential-Algebraic Constraints*, L. T. Biegler, S. L. Campbell, and V. Mehrmann, editors, volume 23 of *Advances in Design and Control*, SIAM, 2012, pages 105–125.

[14] A. Varga. New computational paradigms in solving fault detection and isolation problems. *Annu. Rev. Control*, 37:25–42, 2013.

[15] A. Varga. Descriptor System Tools (DSTOOLS) User's Guide. 2017. `https://arxiv.org/abs/1707.07140`.

[16] A. Varga. *Solving Fault Diagnosis Problems – Linear Synthesis Techniques*, volume 84 of *Studies in Systems, Decision and Control*. Springer International Publishing, 2017.

[17] M. Vidyasagar. *Control System Synthesis: A Factorization Approach.* "Morgan & Claypool", 2011.

[18] G. Vinnicombe. Frequency domain uncertainty and the graph topology. *IEEE Trans. Automat. Control*, 38:1371–1383, 1993.

[19] G. Vinnicombe. *Uncertainty and Feedback: $\mathcal{H}_\infty$ Loop-shaping and the ν-gap Metric.* Imperial College Press, London, 2001.

[20] Z. Yuan, G. C. Vansteenkiste, and C. Y. Wen. Improving the observer-based FDI design for efficient fault isolation. *Int. J. Control*, 68(1):197–218, 1997.

A Installing FDITOOLS

FDITOOLS runs with MATLAB R2015b (or later versions) under 64-bit Windows 7 (or later). Additionally, the *Control System Toolbox* (Version 9.10 or later) and the *Descriptor Systems Tools* (**DSTOOLS**) collection (Version 0.64 or later) are necessary to be installed. To install **FDITOOLS**, perform the following steps:

- download **FDITOOLS** and **DSTOOLS** as zip files from Bitbucket[3]
- create on your computer the directories `fditools` and `dstools`
- extract, using any unzip utility, the functions of the **FDITOOLS** and **DSTOOLS** collections in the corresponding directories `fditools` and `dstools`, respectively
- start MATLAB and put the directories `fditools` and `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 `FDIToolsdemo.m`

Note: The software accompanying the book [16] can be also downloaded as a zip file,[4] which also includes **FDITOOLS** V0.2 and **DSTOOLS** V0.5. To install and execute the example and case-study scripts listed in the book, follow the steps indicated in the web page. An updated collection of MATLAB scripts with examples is also available from Bitbucket.[5]

[3]Download **FDITOOLS** from `https://bitbucket.org/DSVarga/fditools`, and **DSTOOLS** from `https://bitbucket.org/DSVarga/dstools`

[4]`https://sites.google.com/site/andreasvargacontact/home/book/matlab`

[5]`https://bitbucket.org/DSVarga/fdibook_examples`

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:

```
% FDITOOLS - Fault detection and isolation filter synthesis tools.
% Version 1.0           30-November-2018
% Copyright 2016-2018 A. Varga
%
% Demonstration.
%   FDIToolsdemo - Demonstration of FDITOOLS.
%
% Setup of synthesis models.
%   fdimodset  - Setup of models for solving FDI synthesis problems.
%   mdmodset   - Setup of models for solving model detection synthesis problems.
%
% Analysis.
%   fdigenspec - Generation of achievable FDI specifications.
%   fdichkspec - Checking the feasability of a set of specifications.
%   mddist     - Computation of distances between component models.
%   mddist2c   - Computation of distances to a set of component models.
%
% Performance evaluation of FDI filters
%   fditspec   - Computation of the weak or strong structure matrix.
%   fdisspec   - Computation of the strong structure matrix.
%   fdifscond  - Fault sensitivity condition of FDI filters.
%   fdif2ngap  - Fault-to-noise gap of FDI filters.
%   fdimmperf  - Model-matching performance of FDI filters.
%
% Performance evaluation of model detection filters
%   mdperf     - Distance mapping performance of model detection filters.
%   mdmatch    - Distance matching performance of model detection filters.
%   mdgap      - Noise gaps of model detection filters.
%
% Synthesis of FDI filters.
%   efdsyn     - Exact synthesis of fault detection filters.
%   afdsyn     - Approximate synthesis of fault detection filters.
%   efdisyn    - Exact synthesis of fault detection and isolation filters.
%   afdisyn    - Approximate synthesis of fault detection and isolation filters.
%   emmsyn     - Exact model matching based synthesis of FDI filters.
%   ammsyn     - Approximate model matching based synthesis of FDI filters.
%
% Synthesis of model detection filters.
%   emdsyn     - Exact synthesis of model detection filters.
%   amdsyn     - Approximate synthesis of model detection filters.
%
% Miscellaneous.
%   hinfminus  - H-(infinity-) index of a stable transfer function matrix.
%   hinfmax    - Maximum of H-inf norms of columns of a transfer function matrix.
%   efdbasesel - Selection of admissible basis vectors to solve the EFDP.
%   afdbasesel - Selection of admissible basis vectors to solve the AFDP.
%   emmbasesel - Selection of admissible basis vectors to solve the strong EFDIP.
%   ammbasesel - Selection of admissible basis vectors to solve the strong AFDIP.
%   emdbasesel - Selection of admissible basis vectors to solve the EMDP.
```

C FDITOOLS Release Notes

The **FDITOOLS** Release Notes describe the changes introduced in the successive versions of the **FDITOOLS** collection, as new features, enhancements to functions, or major bug fixes. The following versions of **FDITOOLS** have been released:

Version	Release date	Comments
V0.2	December 31, 2016	Initial version accompanying the book [16].
V0.21	February 15, 2017	Enhanced user interface of function `genspec`.
V0.3	April 7, 2017	New function for the exact model-matching based synthesis.
V0.4	August 31, 2017	New function for the exact synthesis of model detection filters and substantial enhancements of most functions. The option to use "design matrices" has been added to all synthesis functions.
V0.8	February 28, 2018	Complete set of functions for the approximate synthesis of fault detection and model detection filters implemented.
V0.85	July 7, 2018	New functions for performance evaluation of fault detection filters have been implemented together with notable enhancements of existing analysis functions. New synthesis options have been implemented in almost all synthesis functions (e.g., using an observer-based nullspace basis, option to perform exact synthesis for approximate synthesis functions, etc.)
V0.87	August 15, 2018	Enhancements of the approximate model-matching synthesis function performed and a new function implemented for model-matching performance evaluation.
V1.0	November 30, 2018	Many enhancements of the approximate synthesis functions performed, by using the version V0.71 of **DSTOOLS**. A set of new functions have been implemented for the analysis of multiple models and performance evaluation of model detection filters.
V1.0.5	July 7, 2019	Enhancements of functions `emmsyn` and `ammsyn` to handle a bank of reference models.

C.1 Release Notes V0.2

This is the initial version of the **FDITOOLS** collection of M-functions, which accompanies the book [16]. All numerical results presented in this book have been obtained using this version of **FDITOOLS**.

C.1.1 New Features

The M-functions available in the Version 0.2 of **FDITOOLS** are listed below:

```
% FDITools - Fault detection and isolation filter synthesis tools.
% Version 0.2            31-Dec-2016
% Copyright 2017 A. Varga
%
% Demonstration.
%   FDIToolsdemo - Demonstration of FDITools.
%
% Analysis functions.
%   fditspec   - Computation of the structure matrix of a system.
%   fdisspec   - Computation of the strong structure matrix of a system.
%   genspec    - Generation of achievable fault detection specifications.
%
% Synthesis functions.
%   efdsyn     - Exact synthesis of fault detection filters.
%   efdisyn    - Exact synthesis of fault detection and isolation filters.
%
% Miscellaneous.
%   efdbasesel - Selection of admissible basis vectors to solve the EFDP.
%
```

C.2 Release Notes V0.21

Version 0.21, dated February 15, 2017, is a minor improvement over version 0.2 of **FDITOOLS**.

C.2.1 New Features

A new version of the function `genspec` is provided, with an enhanced user interface. The new calling syntax, which also covers the previously used calling syntax, is similar to that used by the synthesis functions and allows the direct handling of systems having control, disturbance and additive fault inputs.

C.3 Release Notes V0.3

Version 0.3, dated April 7, 2017, provides a complete set of functions implementing the exact synthesis approaches of fault detection and isolation filters.

C.3.1 New Features

Two new functions have been implemented:

- The function `emmsyn` is provided for the exact synthesis of fault detection and isolation filters, by using an exact model-matching approach.

- The function `emmbasesel` is provided for the selection of admissible left nullspace basis vectors to solve the strong exact fault detection and isolation problem, using an exact model-matching approach.

C.4 Release Notes V0.4

Version 0.4, dated August 31, 2017, is a major new release, including substantial revisions of most functions, by adding exhaustive input parameter checks, new user options and several enhancements and simplifications of the implemented codes. Besides these modifications, two new functions have been implemented for the exact synthesis of model detection filters.

C.4.1 New Features

Two new functions have been implemented:

- The function `emdsyn` is provided for the exact synthesis of model detection filters, by using a nullspace-based synthesis approach.
- The function `emdbasesel` is provided for the selection of admissible left nullspace basis vectors for solving the exact model detection problem.

Several new features have been implemented:

- The functionality of `fditspec` has been enhanced, by generating a three-dimensional structure matrix in the case of several frequency values specified at input. In this case, the pages (along the third dimension) of this array contain the strong structure matrices at different frequency values.
- The functionality of `fdisspec` has been enhanced, by generating a three-dimensional structure matrix in the case of several frequency values, whose pages (along the third dimension) contain the strong structure matrices at different frequency values. An error message is issued if any of the specified frequencies is a system pole.
- The functionality of `genspec` has been restricted, by only allowing a single frequency value to generate strong specifications.
- In the function `efdsyn` a new option to specify a design parameter has been implemented and the internally employed/generated value of this parameter is returned in the `INFO` structure. This allows, among others, the reproducibility of the computed results. This feature also can serve for optimization purposes (e.g., minimizing the sensitivity conditions; see [16, Remark 5.6]).
- In the function `efdisyn` a new option has been implemented to specify design parameters for the synthesis of individual filters. The internally employed/generated values of these design parameters are returned in the `INFO` structure. This allows, among others, the reproducibility of the computed results. This feature also can serve for optimization purposes (e.g., minimizing the sensitivity conditions of the individual filters; see [16, Remark 5.6]).

- In the function `efdisyn` a new option has been implemented to specify a subset of indices of the filters to be designed. This allows, for example, to design separately individual filters of the overall bank of filters.
- In the function `emmsyn` several enhancements of the algorithm implementation have been performed, new options have been implemented for the normalization of the diagonal elements of the updating factor, for the specification of a regularization option, for the specification of a design parameter and for the specification of a complex frequency value to be used for solvability checks. The actually employed design matrix and frequency are returned in two fields of the `INFO` structure.
- A new function `emdsyn` is provided for the exact synthesis of model detection filters, by using a nullspace-based synthesis approach.
- The implementation of `efdbasesel` has been simplified and its functionality has been enhanced, by allowing as input, a three-dimensional structure matrix as computed by `fdisspec`. Also, the selection of admissible vectors is performed, regardless the degree information for the basis vectors is provided or not.
- The implementation of `emmbasesel` has been simplified and the selection of admissible vectors is performed, regardless the degree information for the basis vectors is provided or not.

C.4.2 Bug Fixes

The function `efdsyn` has been updated by fixing a bug in the strong fault detectability test, in the case of more than one frequency values, or in the case when one of the specified frequency values coincides with a system pole.

C.4.3 Compatibility Issues

The function `emmsyn` has been updated to comply with the version V0.6 of **DSTOOLS**.

C.5 Release Notes V0.8

Version 0.8, dated February 28, 2018, is a major new release, including four new functions for the approximate synthesis of fault and model detection filters. Besides this, a few modifications have been performed by adding new user options and fixing some bugs.

C.5.1 New Features

Several new functions have been implemented:

- The function `afdsyn` is provided for the approximate synthesis of fault detection filters, by using a nullspace-based synthesis approach.
- The function `afdisyn` is provided for the approximate synthesis of fault detection and isolation filters, by using a nullspace-based synthesis approach.

- The function `ammsyn` is provided for the approximate synthesis of fault detection and isolation filters, by using an approximate model-matching approach.
- The function `amdsyn` is provided for the approximate synthesis of model detection filters, by using a nullspace-based synthesis approach.
- The function `ammbasesel` is provided for the selection of admissible left nullspace basis vectors to solve the strong fault detection and isolation problem, using an approximate model-matching approach.

Several new features have been implemented:

- In the function `emdsyn`, the following modifications have been performed in specifying user options: (1) the option for a design matrix has been enhanced, by allowing to specify, besides a cell array of matrices, also a unique design matrix; (2) a new option allows to choose either an observer-based nullspace or a minimal rational nullspace; (3) the option specifying the maximum number of residual outputs has been enhanced.

C.5.2 Bug Fixes

Several bug fixes has been performed:

- The function `efdsyn` has been updated to correctly handle the case of no fault inputs and to fix several bugs related to handling the optional design matrix.
- The function `emmsyn` has been updated by fixing several bugs related to handling the optional design matrix.
- The function `emdsyn` has been updated by fixing several bugs (e.g., in selecting a preliminary design, in handling the optional design matrix, fixing names of variables).

C.6 Release Notes V0.85

Version 0.85, dated July 7, 2018, provides notable enhancements of several functions and includes a number of new functions as well.

C.6.1 New Features

The following new functions have been implemented:

- The function `fdimodset` is provided for an easy setup of synthesis models with additive faults for solving FDI synthesis problems.
- The function `mdmodset` is provided for an easy setup of multiple synthesis models for solving model detection synthesis problems.
- The function `fdifscond` is provided to compute the fault sensitivity condition of a stable system or of a collection of systems with additive faults.

- The function **fdif2ngap** is provided to compute the fault-to-noise gap of a stable system or of a collection of systems.
- The function **hinfminus** is provided for the computation of the $\mathcal{H}_{\infty-}$-index of the transfer function matrix of a stable LTI system.
- The function **hinfmax** is provided for the computation of the maximum of the $\mathcal{H}_\infty$-norm of the columns of the transfer function matrix of a stable LTI system.
- The function **afdbasesel** is provided for the selection of admissible left nullspace basis vectors to solve the approximate fault detection problem.

Several new features have been implemented:

- In the function **fditspec**, the following enhancements have been implemented: (1) an option has been added to perform a block-structure based evaluation of the structure matrix (see (14) in Section 2.5); (2) the applicability has been extended to cell arrays of systems sharing the same inputs or the same input group **'faults'**, to allow a block-structure based evaluation of the joint structure matrix (see (14) in Section 2.5). This extension permits the evaluation of the joint structure matrix of the resulting batch of internal filter representations, as computed by the functions **efdisyn** and **afdisyn**.
- In the function **fdisspec**, the following enhancements have been implemented: (1) an option has been added to perform a block-structure based evaluation of the structure matrix (see (14) in Section 2.5); (2) the applicability has been extended to cell arrays of systems sharing the same inputs or the same input group **'faults'**, to allow a block-structure based evaluation of the joint structure matrix (see (14) in Section 2.5). This extension permits the evaluation of the joint structure matrix of the resulting batch of internal filter representations, as computed by the functions **efdisyn** and **afdisyn**.
- In the function **efdsyn**, a new option allows to choose in the initial synthesis step either an observer-based (possibly non-minimal) nullspace basis (only in the case of lack of disturbance inputs) or a minimal rational nullspace basis (the default option).
- In the function **afdsyn**, the following enhancements have been performed: (1) the resulting fault detection filter has, in general, a two-block structure (see [16, Remark 5.10]); (2) accordingly, the option to specify the number of residual outputs has been enhanced; (3) a new user option allows to specify separate design matrices for the two components of the filter; (4) a new option allows to choose in the initial synthesis step either an observer-based (possibly non-minimal) nullspace basis or a minimal rational nullspace; (5) a new option to perform *exact* synthesis is provided (this functionality equivalent to that of the function **efdsyn**); (6) a new option is available to specify a test frequency value to be employed to check the rank-based admissibility condition; (7) an extended set of additional information is returned to the user.
- In the function **efdisyn**, the following enhancements have been performed: (1) a new option allows to choose in the initial synthesis step either an observer-based (possibly non-minimal) nullspace basis (only in the case of lack of disturbance inputs) or a minimal

rational nullspace basis (the default option); (2) the option to specify the numbers of residual outputs of each filter has been implemented.

- In the function `afdisyn`, the following enhancements have been performed: (1) each component of the resulting bank of fault detection filters has, in general, a two-block structure (see [16, Remark 5.10]); (2) accordingly, the option to individually specify the numbers of residual outputs of each filter has been implemented; (3) a new user option allows to specify separate design matrices for the two components of the filters; (4) a new option allows to choose in the initial synthesis step either an observer-based (possibly non-minimal) nullspace basis (only in the case of lack of disturbance inputs) or a minimal rational nullspace basis; (5) a new option to perform *exact* synthesis is provided (this functionality equivalent to that of the function `efdisyn`); (6) a new option is available to specify a test frequency value to be employed to check the rank-based admissibility conditions; (7) the additional information returned to the user has been updated.
- In the function `emdsyn`, the option to individually specify the numbers of residual outputs of each filter has been implemented.
- In the function `amdsyn`, the option to individually specify the numbers of residual outputs of each filter has been implemented.
- The demonstration script `FDIToolsdemo` has been updated to use calls to the newly implemented confort functions to setup models and evaluate suitable performance measures.

The checking of solvability conditions related to the existence of left nullspace bases has been enhanced for all synthesis functions, by issuing appropriate error messages in the case of empty nullspace bases.

C.6.2 Bug Fixes

Several minor bug fixes have been performed in the functions `efdsyn` and `afdsyn`.

C.7 Release Notes V0.87

Version 0.87, dated August 15, 2018, provides enhancements of the approximate model-matching synthesis function `ammsyn` and includes the new function `fdimmperf` for model-matching performance evaluation.

C.7.1 New Features

The following new function has been implemented:

- The function `fdimmperf` is provided to compute the model-matching performance of a stable system or of a collection of stable systems.

Several new features have been implemented:

- In the function `ammsyn`, the following enhancements have been performed: (1) extension to handle arbitrary reference models with control, disturbance, fault and noise inputs; (2) enhancing the nullspace-based synthesis, by explicitly considering strong FDI problems and fault detection problems; (3) a new option allows to choose in the initial synthesis step either an observer-based (possibly non-minimal) nullspace basis (only in the case of lack of disturbance inputs) or a minimal rational nullspace basis (the default option); (4) support for void fault inputs provided; (5) providing both optimal and suboptimal performance values in the `INFO` structure.
- The function `ammbasesel` has been extended to handle both strong FDI related selection as well as fault detection oriented selection of basis vectors.

C.7.2 Bug Fixes

Several minor bug fixes have been performed in the functions `fdifscond` and `fdif2ngap`.

C.8 Release Notes V1.0

Version 1.0, dated November 30, 2018, concludes the development of a fairly complete collection of tools to address the main computational aspects of the synthesis of fault detection and model detection filters using linear synthesis techniques. This version provides several new functions for the analysis of model detection problems and the evaluation of the performance of model detection filters. Besides this, several enhancements of the approximate synthesis functions have been performed related to the handling of non-standard cases, as well as of the synthesis functions of model detection filters. This version of **FDTOOLS** relies on the version V0.71 of the Descriptor Systems Tools **DSTOOLS**.

C.8.1 New Features

The following new function has been implemented:

- The function `fdichkspec` is provided to check the feasibility of a set of FDI specifications.
- The function `mdperf` is provided to compute the distance mapping performance of model detection filters.
- The function `mdmatch` is provided to compute the distance matching performance of model detection filters.
- The function `mdgap` is provided to compute the noise gap performance of model detection filters.

The following new features have been added:

- The function `fdigenspec` replaces `genspec` and can now handle multiple frequency values to determine strong FDI specifications. For compatibility purposes, the obsolete function `genspec` can be still used, but it will be removed in a future version of **FDITOOLS**.

- In the functions `afdsyn` and `afdisyn`, new options have been added to the `OPTIONS` structure, to handle non-standard problems. Also, the detection of non-standard problems has been simplified, being done without additional computations, using the new information provided by `goifac` in the version V0.7 (and later) of **DSTOOLS**.
- In the function `ammsyn`, the optimal performance value for the original problem is provided in the `INFO` structure, jointly with the optimal and suboptimal values of the updated problem. Also, the detection of non-standard problems has been simplified, being done without additional computations, using the new information provided by `goifac` in the version V0.7 (and later) of **DSTOOLS**.
- In the function `emdsyn`, a new option has been added to the `OPTIONS` structure, to handle the extended model detectability. Also, an new option has been added to perform specific normalizations of the resulting model detection filters.
- In the function `amdsyn`, a new option has been added to the `OPTIONS` structure, to handle the extended model detectability. Also, new options have been added, to handle non-standard problems. In this context, the detection of non-standard problems has been simplified, being done without additional computations, using the new information provided by `goifac` in the version V0.7 (and later) of **DSTOOLS**.

C.8.2 Bug Fixes and Minor Updates

- The functions `fditspec`, `fdisspec`, `fdifscond`, `fdif2ngap` and `fdimmperf` and their documentations have been updated by using a new notation for the underlying internal forms of the FDI filters.
- Several minor bug fixes have been performed in the function `efdisyn`.
- The functions `emdsyn` and `amdsyn` have been updated to comply with the new definitions of model detectability and extended model detectability introduced in Section 3.4.

C.9 Release Notes V1.0.5

C.9.1 New Features

Several new features have been implemented:

- The functions `emmsyn` and `ammsyn` have been extended to handle a bank of reference models, as, for example, the resulting internal forms from the synthesis functions `efdisyn` and `afdisyn`.

C.9.2 Bug Fixes and Minor Updates

- The functions `efdsyn` and `afdsyn` have been updated to improve the automatic (random) selection of design matrices.
- Several minor bug fixes have been performed in the functions `fditspec` and `fdisspec`.

Index