DLR-IB-FT-BS-2026-9

FitlabGui - A MATLAB® Tool for Flight Data Analysis and Parameter Estimation - Version 2.8.0

Interner Bericht

Susanne Seher-Weiß

DLR German Aerospace Center
Institute of Flight Systems
Rotorcraft
Braunschweig

DLR Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Institute Report
DLR-IB-FT-BS-2026-9

FitlabGui
A MATLAB® Tool for Flight Data Analysis and Parameter Estimation
Version 2.8.0

S. Seher-Weiß

Institute of Flight Systems
Braunschweig

German Aerospace Center (DLR)
Institute of Flight Systems
Department Rotorcraft

Braunschweig, 30 January 2026

Signatures

Director of Institute:	Prof. Dr. Ch. Keßler
Head of Department:	M. Höfinger
Author:	S. Seher-Weiß

DLR German Aerospace Center
Institute of Flight Systems
Rotorcraft
Dipl.-Ing. Marc Höfinger and Dr. Klausdieter Pahlke
Lilienthalplatz 7
38108 Braunschweig
Germany
Tel: +49 531 295-2691
Fax: +49 531 295-2641
Web: https://www.dlr.de/ft/en/hub

Susanne Seher-Weiß
Tel: +49 531 295-3080/-2616
Mail: susanne.seher-weiss@dlr.de

Document Identification:

Report number	DLR-IB-FT-BS-2026-9
Title	FitlabGui - A MATLAB® Tool for Flight Data Analysis and Parameter Estimation - Version 2.8.0
Author(s)	Susanne Seher-Weiß
Filename	FitlabGui.tex
Last saved on	30th January 2026

Contents

Nomenclature

Symbol	Description
A	stability matrix
B	control matrix
b_x, b_y	bias parameters of the state resp. observation equations
C	observation matrix
F	transfer function
f	frequency
G_{xx}, G_{yy}	input and output autospectra
G_{xy}	cross spectrum
H	frequency response
I	identity matrix
J	information matrix
j	imaginary unit $j = \sqrt{-1}$
K_u	control amplification parameter
k	discrete time/frequency index, short notation for t_k or f_k
L	cost function
m	number of observation variables
N	number of data points
N_ω	number of frequency points
N_{TF}	number of transfer functions to be approximated together
NZ	number of time intervals
P	estimation error covariance matrix
R	covariance matrix of measurement noise
S	spectral density matrix of measurement noise
s	Laplace variable
T	length of time interval
t	time
u	control input vector
V	Fourier transform of noise
v	measurement noise vector
W	weighting function/matrix
w_{ap}	weighting ratio of amplitude vs. phase error
w_γ	coherence weighting
X	Fourier transform of x
x	state vector
Y	Fourier transform of y
y	observation vector
z	measurement vector

Symbol	Description
$\Delta u, \Delta z$	measurement error (offset) in the inputs and outputs
Δt	sampling interval
$\Delta\theta$	parameter increment
δ	parameter variation
δ_{ij}	Kronecker delta ($= 1$ if $i = j, = 0$ otherwise)
ϵ_r	random error
ϵ	vector of magnitude and phase errors
ζ	damping ratio
γ^2_{xy}	coherence between x and y
θ	complete parameter vector
φ	vector of unknown system parameters
σ_i	standard deviation of θ_i
$\sigma(\ldots)$	standard deviation
$\sigma^2(\ldots)$	variance
$\rho(\theta_i, \theta_j)$	correlation between θ_i and θ_j
τ	time delay
ω	frequency
ω_n	natural frequency
$\angle$	phase angle, deg
$\lvert\ldots\rvert_{dB}$	log-amplitude, dB
$\Re(\ldots)$	real part
$\Im(\ldots)$	imaginary part

Subscripts

0	initial value
$free$	free parameter (not reaching bound)
in	input
m	measured value
min	minimal value
out	output

Superscripts

T	transpose of real matrix
$*$	conjugate transpose of complex matrix
-1	inverse of matrix
$\hat{}$	optimal value

Abbreviations

FR	frequency response
FRD	frequency response data
GUI	graphical user interface
LTI	linear time-invariant
ML	Maximum Likelihood
SS	state space
SysID	system identification
TF	transfer function
ZPK	zero-pole-gain

The nomenclature of the examples in chapter 10 is as per conventions of LN 9300/ISO 1151.

1. Background and Introduction

System identification has long been a focal point of research efforts at the DLR Institute of Flight Systems [1, 2]. Consequently several parameter estimation software packages were developed over the years, that could handle linear systems in the time and frequency domain as well as nonlinear systems and implemented output error, filter error and extended Kalman filter methods [3–8]. All of these software packages were written in FORTRAN.

When more and more research at the Institute was performed using MATLAB and Simulink [9, 10], it was desired to perform system identification directly in the MATLAB environment. That let to the development of the parameter estimation package PENSUM [11], later renamed into FITLAB, that implemented Maximum Likelihood (ML) parameter estimation of general nonlinear systems written in MATLAB / Simulink.

Soon, FITLAB was equipped with a graphical user interface (GUI) to provide easy access to the standard tasks of reading the measured data, specifying the model and parameters, running the identification and looking at plots of the results. The software package was thus called FitlabGui and over time it was enhanced to allow for the identification of different model types, namely

- nonlinear models (ML method)
- linear models in the time domain (ML method)
- linear models in the frequency domain (ML method)
- transfer function models (frequency response method)
- linear models in the frequency domain (frequency response method)

For parameter optimization, FitlabGui provides a standard Gauss-Newton algorithm and in addition a subplex based method. Upper and lower bounds for the unknown parameters can be specified for both methods.

This report covers the standard FitlabGui tool. The optional add-on package HQ-Tools (handling qualities analysis tools) is described in a separate report [12].

After the description of the installation, all menus of FitlabGui are explained in detail. Additional chapters cover the demonstration examples and the command line version of the underlying system identification tool FITLAB. Basic information about frequency

response generation, system identification theory and the implemented algorithms is provided in the appendices.

2. Installation & Call

2.1. Installation

FitlabGui is delivered as a zip-archive. Most routines are contained as MATLAB p-code. Only those routines which the user might want to modify or use as a template for own routines are delivered as m-files.

For a normal installation, the user has to choose a directory, where he wants the zip-archive to be extracted. Subdirectories named `fitlab_pcode`, `fitlabGui_pcode`, and `fitlabDemo` are then created and filled. All of these directories have to be included in the MATLAB search path.

For MATLAB version 2012b and newer, FitlabGui can also be installed as an app. To do so,

- select the tab **APPS**,
- press the icon **Install App**, and
- select the file **FitlabGui.mlappinstall** from the zip-archive.

A FitlabGui icon is then created in the APPS tab.

2.2. Calling Procedure

If FitlabGui has been installed as an app, it is started through the FitlabGui icon. Otherwise, FitlabGui is invoked by typing `fitlabGui` on the command line. The main FitlabGui panel is depicted in figure 2.1.

The main menus are **Project**, **Data**, **Model**, **Execution**, **Plotting**, and **Help**. All of these menus are described in the following chapters. If the optional Heli-HQ toolbox is installed, an additional menu **Heli-HQ** appears between **Plotting** and **Help**. The menus **Model** and **Execution** pertain to parameter estimation with FITLAB and the

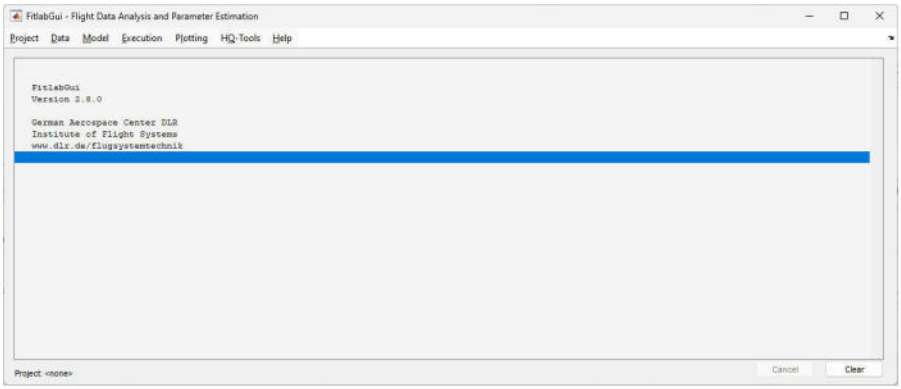


Figure 2.1.: Main FitlabGui panel

corresponding chapters in this report can thus be skipped by users that do not want to perform parameter estimation.

The scrollable window of the main FitlabGui panel displays some status information as well as all output of the called routines for frequency response generation, system identification, or handling qualities analysis. The **Clear** button on the main panel allows to clear this window if the displayed information is no more needed (doing so makes the subsequent calculations faster).

The GUI and the underlying routines for frequency response generation and system identification use a global variable named `FITLAB` for communication. This global variable must not be altered or cleared by the user.

FITLAB requires MATLAB 2012 (8.0), Simulink 8.0 and Control System Toolbox 9.4. It has been tested with newer versions up to MATLAB 2024b. For reading flight test data in the Institute's standard format, CDF version 3.5 [13] is used.

3. Project

All current settings in FitlabGui can be saved and retrieved by so-called project files. The project files are mat-files that contain a structure that stores all settings for all FitlabGui menus. The name of the current project file (if any) is displayed on the bottom line of the main panel (see figure 2.1).

The menu items in the **Project** menu are:

New Start a new project from scratch.

Open Open an existing project.
The corresponding mat-file has to be selected.

Save Save all current settings to the currently opened project file.

Save as Save all current settings to another project file.
Only the name of the logfile for the system identification (see figure 6.1) is changed to the new project name with the file extension log.
Similarly, the logfile for Heli-HQ is changed to `<new_project_name>_hat.log` if one had been defined.

Info Create or display the information about the current project.
This information can be used as a subtitle in the plots (see section 7).

Exit FitlabGui
Exit the program FitlabGui.
If changes have been made compared to the settings of the current project, the user is asked whether he wants to save the changes before exiting the program.

4. Data

FitlabGui allows to evaluate and analyze both time history and frequency response data. The **Data** menu of FitlabGui has the following items:

- **Load Time Sections from File**
- **Select Time Sections from Maneuver Database**
- **Maneuver Database Definition**
- **Unit Conversion**
- **Channel Arithmetic**
- **Export**
- **Load Frequency Responses from File**
- **Frequency Response Generation**
- **HOS Definition**

The first six items pertain to time history data whereas the other items handle frequency response data. The supported file formats for both data types as well as all data menu items are explained in the next sections.

4.1. Supported Formats for Time History Data

The FitlabGui supports the following file formats for time history data:

CDF CDF datasets are based on the Common Data Format [13] developed at the National Space Science Data Center (NSSDC) of NASA. At the Institute, two variants of CDF datasets are currently used, namely IS3 and R-CDF.
IS3 is the older of the two formats. IS3 datasets contain all data channels already interpolated to a common sampling rate and all data are real numbers. A short description of the IS3 standard can be found in [14].
R-CDF (Raw Common Data Format) is the data format used for all current flight test programs at the Institute. In R-CDF datasets, different data channels can be of different type and have different sampling rates. A description of the R-CDF data format is found in chapter 10 of [15]. Upon reading R-CDF data into FitlabGui, all data channels are interpolated to the desired sampling rate and

converted to double. Sample-and-hold is used for integer data whereas linear interpolation is used for real data.

MATLAB
MATLAB mat-files to be read by the FitlabGui have to contain each measured data channel as a separate vector variable. An optional structure `channel` with character fields `channel.names` and `channel.units` can be present to define the units for each data channel. If a `channel` structure is present, the character strings in `channel.names` have to correspond to the names of the vector variables with data. Other non-numeric variables and scalar variables contained in the mat-file are ignored. Examples for data mat-files with units are the data files for the X-31 flight path reconstruction example from section 10.

ASCII, Excel
ASCII files and Excel data files have to contain the data including the variable names in tabular form with the different channels in separate columns or rows. Additional header lines can be present. Data units are not supported for ASCII or Excel files.

CDF files are read directly using the corresponding routines to extract the header information and the data itself. For the other file types, a **Data Import** menu is opened that displays all signals available in the data file. One variable must then be selected as the time signal.

Data that are not in one of the formats supported by FitlabGui can be imported with a user written data interface routine. This function is executed if it has been specified in the top part of the **Load Time Sections from File** panel (see figure 4.1).

The data interface routine has to take `filename`, the name of the data file to be read, as an input argument. The routine must return a structure (e.g. `A`) with the fields

`A.colheaders`	data headers if the data is in columns (cell array of strings)
`A.rowheaders`	data headers if the data is in rows (cell array of strings) (Only one of `A.colheaders` or `A.rowheaders` may be specified.)
`A.units`	units of the data channels (cell array of strings, optional) (The units in `A.units` must correspond to the names in `A.colheaders` resp. `A.rowheaders`.)
`A.data`	matrix with the data (in column or row orientation)

4.2. Load Time Sections from File

Time history data is imported via the **Load Time Sections from File** item from the **Data** menu of the GUI. Figure 4.1 shows the corresponding panel. On the very top, a checkbox allows to select a **Customer Data Interface** (see section section 4.1), for which the corresponding m-file then has to be specified. Next, the data file for each time section has to be chosen. The **Open File** button allows to select one or more data files.

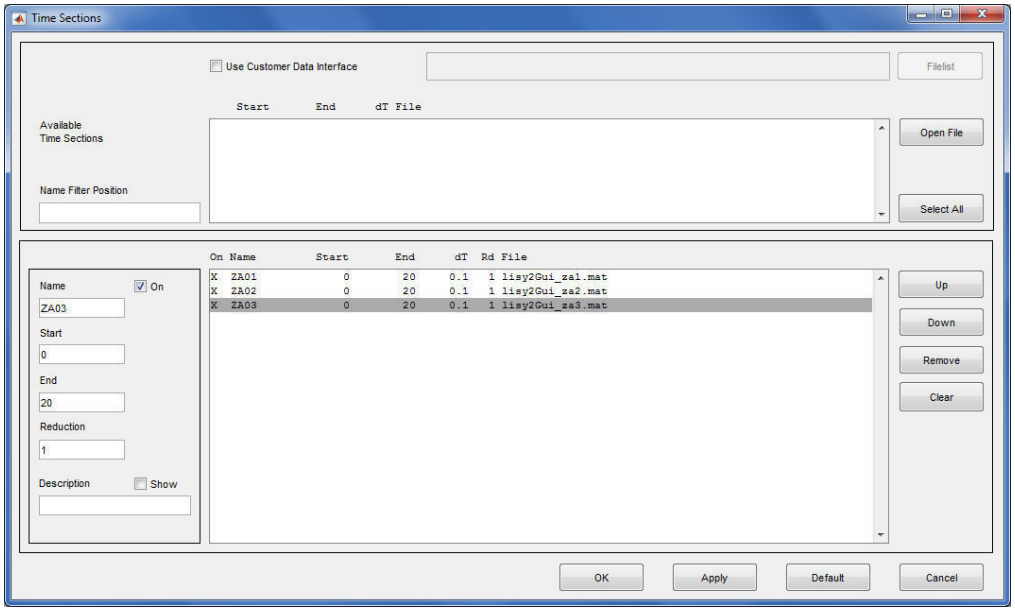


Figure 4.1.: Selection of data files and time intervals

Once one or more data files have been loaded, all available time sections from all opened files are listed. (Several time sections in one file are only possible for CDF files of type IS3.) Clicking on one of the time sections adds this section to the selection list in the bottom part of the panel. Alternatively, all available time sections can be selected at once by pressing the **Select All** button.

If the name of the data files contains information that could be used to name the corresponding time section, the **Name Filter Position** allows to specify the range of characters within the filename that shall be used to name the time section. In the example in figure 4.1 the characters 10 through 12 have been used to label the time sections.

For each time section that is to be evaluated, the bottom part of the **Time Sections** panel allows to

- switch the time section on or off
  (Only time sections that are on can be used for plotting, system identification or handling qualities analysis.)
- give a descriptive name to the time section
  (Default names are ZA01, ZA02, ... These names are used for annotation in the different plots.)
- specify the start and end times
  (The interval must be available in one of the time sections of the data file.)
- specify if the data is to be reduced by reading only every i-th data point
- edit the description of the time section
- show or hide the time section descriptions

The data files that are evaluated together do neither need to have the same number of channels nor do the channels have to be in identical order. All channels that are needed for each chosen evaluation have of course to be available in all data files. Channels are identified by name and the measured channels of the first data file are used for creating the channel lists in all panels.

4.3. Select Time Sections from Maneuver Database

Flight test campaigns often consist of many maneuvers. If a list of these maneuvers has been stored in a database of an appropriate format (see section 4.4) the maneuvers to be evaluated can be selected with the **Select Time Sections from Maneuver Database** option. The corresponding panel is shown in figure 4.2.

The **List of Maneuvers** shows all maneuvers available in the database and the corresponding field values. There are three lines for each maneuver: 1. `Description`, 2. Available filter parameters, 3. `Name`, `TS`, `TE`, `FileName`.

`Description`, `Name`, `TS`, `TE` and `FileName` are required for the database. Any additional fields yield the criteria that are available for filtering the data. These fields can be of data type numerical or character. They are automatically available in the **Filter Criteria** pulldown menus on the right of the panel. Each filter criterion can be compared to a value with the '>', '<', '=', and '~=' (not equal) condition. Several criteria are combined with 'and'. The filtered maneuvers can optionally be sorted by a criterion. Therefore, the example in figure 4.2 looks for all maneuvers at Mach numbers greater than 0.7, a CG location of less than 23% and with gear up and the results are sorted by increasing altitude.

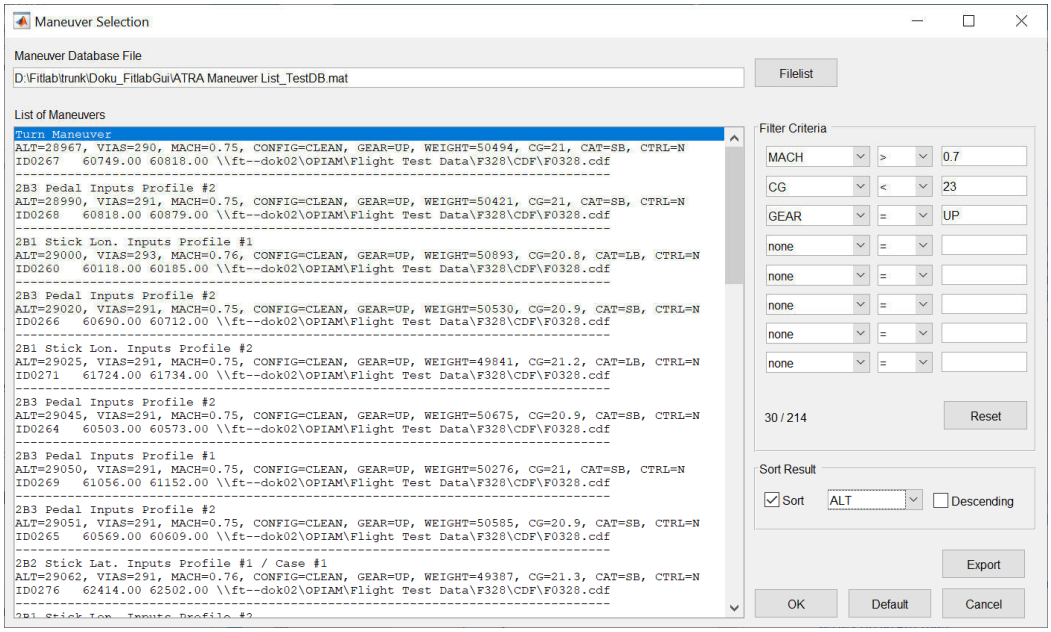


Figure 4.2.: Selection from maneuver database

Upon pressing the **OK**-button, a popup menu appears where the user has to specify whether the filtered maneuvers shall replace the maneuvers currently contained in the **Load Time Sections from File** panel or whether they shall be appended. Using the **Append** option allows to implement 'or'-combinations of different filter criteria. The `Name` for each maneuver is automatically used as name in the **Load Time Sections from File** panel, see figure 4.1.

The **Export**-button allows to export the currently filtered maneuvers to an Excel-file.

4.4. Maneuver Database Definition

The **Maneuver Database File** must be a mat-file containing a vector of structures `maneuverDB` with the following required fields:

`Name`	maneuver name
`TS`	start time of the maneuver
`TE`	end time of the maneuver
`FileName`	name of the datafile that contains the maneuver
`Description`	description of the maneuver

Additional fields in the maneuver database allow to filter the maneuvers as shown in the previous section.

The **Maneuver Database Definition** option allows to create or modify such a maneuver database. The corresponding panel is shown in figure 4.3.

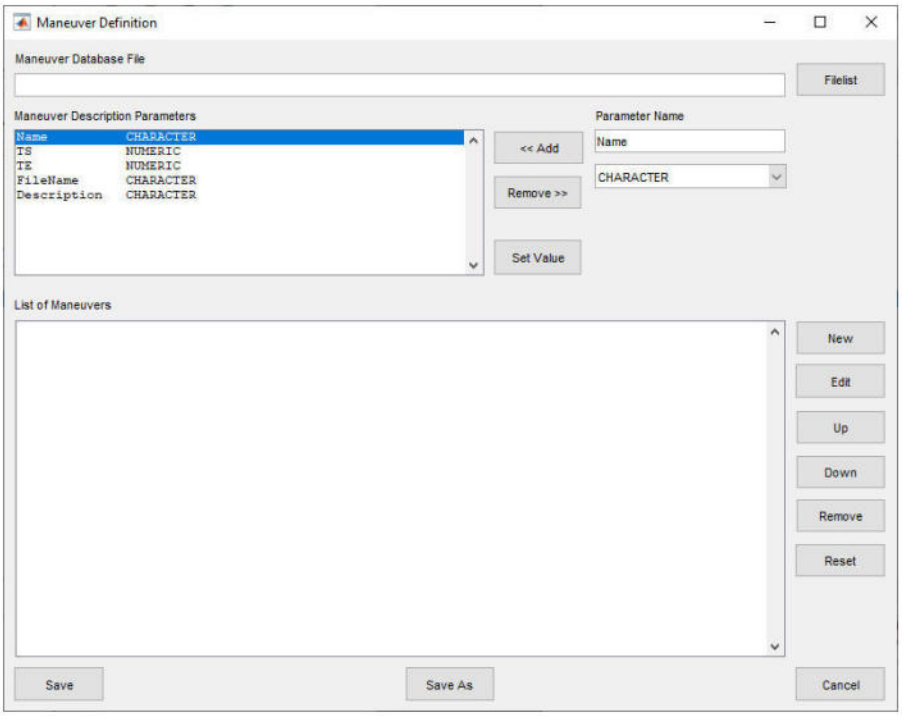


Figure 4.3.: Maneuver definition

To generate a completely new database, the name of the database file has to be specified in the **Maneuver Database File** field. The same field is used to select an already existing database file, that is to be modified. In the latter case all maneuver description parameters and all maneuvers available in that database are then listed, whereas in case of a new database only the required description parameters are listed. Additional maneuver description parameters can be inserted by specifying the corresponding name and data type (character or numeric) and then pressing the **«Add** button. Similarly, the **Remove»** button allows to remove parameters from the database.

New maneuvers are added to the database via the **New** button next to the list of maneuvers. A panel similar to the quicklook plot (see section 7.1) opens where the flight test file for the maneuver has to be specified. All data channels available in this file are then listed and up to 12 channels can be selected for plotting. Pressing the **Plot** button creates a quicklook plot with an additional **Maneuver Cut** option

in the menu bar. Selecting this option allows to specify the start and end times (TS, TE) of a maneuver with the mouse pointer. Once these times have been selected, a pop-up menu appears where at least all required description parameters have to be given for the just specified maneuver. Values for additional parameters can also be specified in this pop-up menu, but for cases where the value of an additional parameter is to be derived from the measured data, the options of the **Set Value** button (see below) should be used after all maneuvers have been defined. When the **OK** button is pressed, the maneuver is marked in the plot and the values are added to the database.

Two options are available for setting or changing values in the database. To set values for several maneuvers at once, select the parameter to be set in the list of description parameters and the corresponding maneuvers in the maneuver list. Pressing the **Set Value** button next to the list of description parameters opens a panel that allows to set the value of the currently selected maneuver description parameter either for all or only for the currently selected maneuvers. It is possible to set a fixed value, to use either the first value or the mean over the first 100 ms of a specified data channel.

To change values for only one specific maneuver, select the maneuver in the list and press the **Edit** button. A popup window appears where all of the maneuver description parameters for this maneuver can be edited.

The remaining buttons beside the list of maneuvers allow to change the maneuver order.

4.5. Unit Conversion

The **Unit Conversion** item of the **Data** menu allows to rescale data that has been selected via the **Load Time Sections from File** panel, for example to remove sensor offsets or change data units. All channels available in the dataset(s) are shown and for each channel x_{orig} a conversion of the form

$$x_{new} = F * x_{orig} + O$$

with factor F and offset O can be specified, i.e. the signal x_{orig} is replaced by the new, converted signal x_{new}. For the most commonly used conversions between different data units (e.g. angles from radians to degrees or vice versa), over 25 conversions are predefined and can be selected from a list.

4.6. Channel Arithmetic

More elaborate calculations than simple unit conversions can be performed on the data using the **Channel Arithmetic** item of the **Data** menu which invokes the panel shown in figure 4.4. These calculations are performed after any unit conversions so that the converted channels are available.

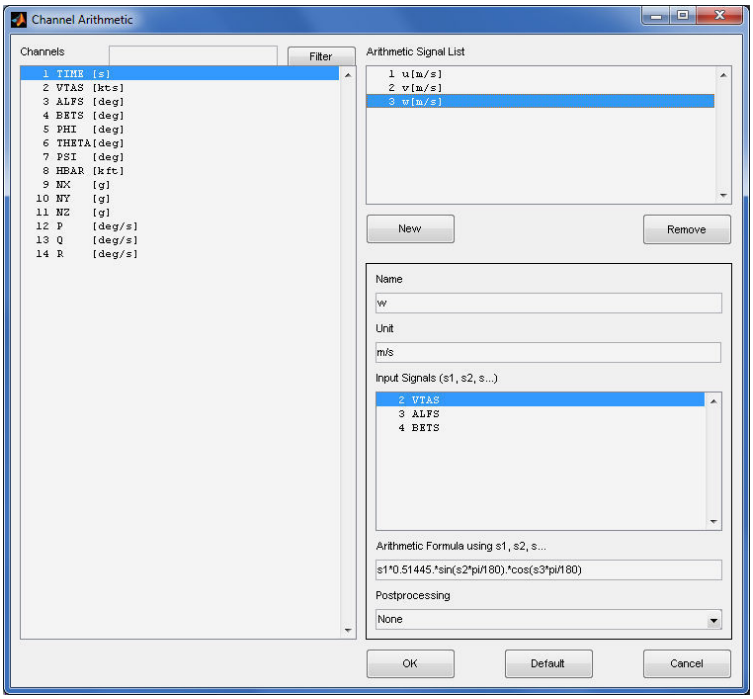


Figure 4.4.: Channel arithmetic

A new arithmetic signal is created by pressing the **New** button. Each arithmetic signal has to be given a name and optionally a unit. The name for the calculated signal must not contain any operators. As many signals as needed can be selected as input signals from the list of all available measured signals and are accessed as `s1`, `s2`, `s3`, ... in the arithmetic formula.

The **Arithmetic Formula** may contain any string that can be evaluated using the MATLAB `eval` command. Thus, any function that is on the MATLAB searchpath can be called in the formula. FitlabGui provides several utility functions that can be used in the channel arithmetic. They are listed and described in chapter 9.

Integration and differentiation are possible **Postprocessing** options. Integration is done using the trapezoidal rule

$$y_1 = 0, \qquad y_i = y_{i-1} + \Delta t(y_{i-1} + y_i)/2$$

where y is the integral of x. The five point method for the first derivative

$$\dot{x}_i = (x_{i-2} - 8x_{i-1} + 8x_{i+1} - x_{i+2})/(12\Delta t)$$

is used for differentiation.

The channels that are created by the channel arithmetic are appended to the end of the channel list. Upon saving the channel arithmetic definition with the **OK** button, each formula is first tested with 10 artificial data points to make sure that the return value is of correct dimension. If the user wants to save the channels that were calculated with the channel arithmetic, he can do so with the **Export** item from the **Data** menu that is described in the next section.

4.7. Export

The **Export** item of the **Data** menu allows to export data from some or all data channels to a file. This option can for example be used when the original CDF files are very large and only a subset of the original data is needed for the evaluation. Possible file formats for the export are the MATLAB mat-format readable by FitlabGui, CDF format and ASCII format. Blanks within channel names are replaced by underscores for the export.

The data channels to be exported are selected from the list of all available channels. The settings from the menu items **Unit Conversion** and **Channel Arithmetic** are accounted for, so that converted and/or newly calculated channels can be exported. Once a simulation or an identification run has been performed, the model outputs as well as the output errors can be exported as well.

Alternatively, only the channel list without any data can be written into an ASCII file by selecting **Pure Channel Names without Data** as the output format.

4.8. Supported Formats for Frequency Response Data

Frequency responses that can be used in FitlabGui must be scalar LTI-objects (linear time invariant systems, see [16]) that are saved in mat-files. They can either be

analytically defined or measured.

Analytically defined LTIs can be of type TF (transfer function) or ZPK (zero-pole-gain). This type of LTIs is easily defined in FitlabGui via the **HOS Definition** item (see section 4.11) of the **Data** menu. Furthermore, LTIs of type SS (state-space system) or genSS (generalized SS) can be used, if they have only one input and one output variable.

LTIs that contain measured frequency responses must be of type FRD (frequency response data) or genFRD (generalized FRD) and the coherence can optionally be saved in the 'Userdata' of the FRD-object. Frequency responses of the FRD type can easily be created from time domain data via the **Frequency Response Generation** item (see section 4.10) of the **Data** menu.

4.9. Load Frequency Responses from File

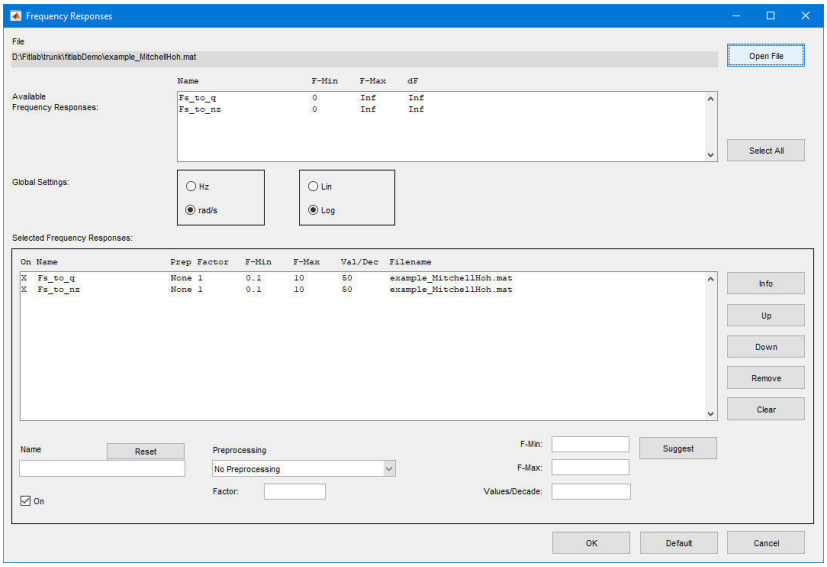


Figure 4.5.: Selection of frequency responses from file

Figure 4.5 shows the panel for loading frequency responses from a file. First, the **Open File** button allows to select a data file. Once a file has been selected, the frequency responses available in this file are listed. Clicking on one of the frequency responses adds this response to the list of selected frequency responses in the bottom half of the panel. Pressing the **Select All** button adds all frequency responses from the currently selected file.

Global settings such as the unit of the frequency axis and the scaling (linear or logarithmic) can also be chosen in the top half of the panel. When logarithmic scaling is selected and the original data is linearly spaced, a moving average algorithm is used to reduce the data.

For each selected frequency response, the bottom part of the **Frequency Responses** panel allows to

- switch the frequency response on or off
  (Only frequency responses that are on can be used for system identification or handling qualities analysis.)
- give a new name to the frequency response
  (Frequency responses that are to be used for system identification with the polynomial or frequency response method have to have distinct names. The **Reset** button reverts the name to the original name from the data file.)
- apply preprocessing to the frequency response
  (Possible preprocessing options are differentiation and integration, default is no preprocessing.)
- scale the frequency response by a factor
  (Caution: Only the frequency response is scaled, not the corresponding spectra!)
- specify the minimum and maximum frequency and the frequency spacing
  For (gen)FRD objects the **Suggest** button shows the frequency range available. For TF, ZPK, and (gen)SS objects, a default frequency range of 0.01 to 100 rad/s is suggested with either a linear spacing of 0.1 rad/s or a logarithmic spacing of 1000 values per decade.

The **Info** button on the right side displays information about the currently selected frequency response. For FRD objects, the frequency range and spacing of the data is displayed, for TF and ZPK objects the coefficients respectively poles and zeros are shown.

4.10. Frequency Response Generation

The **Frequency Response Generation** item of the **Data** menu of FitlabGui invokes the panel shown in figure 4.6. It allows to generate frequency responses from the time domain data selected via the **Time Sections** panel (see figure 4.1). If several time sections are currently selected, the data from all sections will be detrended and concatenated before using it to determine the frequency responses.

On the left side of the panel, all channels available in the time domain data are listed. On the top of the right side, radio buttons allow to switch between the **Primary Input**,

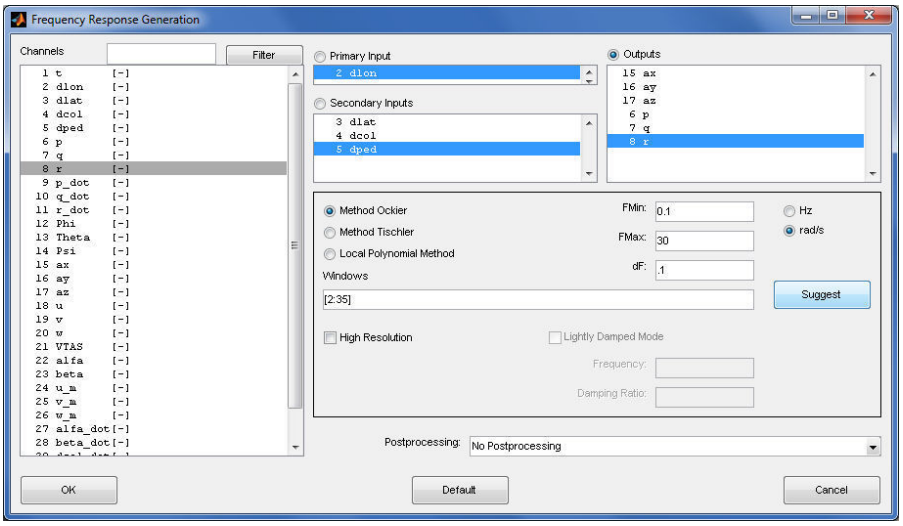


Figure 4.6.: Frequency response generation from time domain data

Secondary Inputs (optional) and the **Outputs**. Clicking on any channel in the list on the left moves the corresponding channel to the currently active section on the right.

Once the inputs and outputs have been selected, the method for generating the frequency responses as well as the frequency range and spacing have to be specified. FitlabGui offers three methods for generating frequency responses, namely the methods of **Ockier**, **Tischler**, and the **LPM** (local polynomial method). All three methods are described in detail in appendix A.

The first two methods use segmenting and windowing and differ mainly in the determination of the composite frequency response from the results for different segments. The method of Ockier determines the frequency response and the corresponding coherence whereas the method of Tischler also yields the input and output autospectra and the cross spectrum.

- For the method developed by **Ockier**, a range of window numbers has to be specified. The weighting function for computing the composite frequency response is based on whether **High Resolution** was selected.
- For the alternative method developed by **Tischler**, a range of window lengths has to be specified. The suggested method for choosing these window lengths is given in [17] and requires accounting for the **Lightly Damped Mode** with the highest frequency if any lightly damped modes are present.

In contrast to the windowing methods, the **LPM** does not eliminate the leakage term through the application of windows, but it considers the leakage as an unknown

function that has to be determined together with the desired frequency response. The LPM method requires the user to specify a vector of neighboring frequencies and yields no coherence information.

For all three methods, the **Suggest** button adjusts the frequency range to the values that are available based on the record length and the sampling rate of the time domain data. The button also suggests values for the number of windows resp. window lengths resp. neighboring frequencies based on the frequency range and the options regarding high resolution or lightly damped modes. Integration and differentiation can be selected as **Postprocessing** options for all methods.

Pressing the **OK** button starts the generation of frequency responses from the selected input to the selected outputs. Multi input /single output (MISO) conditioning is only performed if secondary inputs were selected.

Once the frequency responses have been generated, they can be inspected via Bode plots (similar to figure 7.10) and saved as FRD objects in mat-files so they can later be used for modeling. Upon saving the frequency responses, the user is asked whether the generated frequency responses shall be automatically loaded into FitlabGui.

The generated FRD objects have the following components:

`Frequency`	frequency vector
`ResponseData`	complex frequency response
`Units`	'rad/s' or 'Hz'
`InputName`	name of the (primary) input channel (as in the dataset)
`OutputName`	name of the output channel (as in the dataset)
`Notes`	string that lists the method that was used for FR generation
`UserData.coh`	coherence (not for LPM)
`UserData.Gxx`	(conditioned) input autospectrum (Tischler method only)
`UserData.Gyy`	(conditioned) output autospectrum (Tischler method only)
`UserData.Gxy`	(conditioned) cross spectrum (Tischler method only)
`UserData.Err`	random error (Tischler method only)

Should the user wish to cancel a running frequency response generation, he can do so by pressing the **Cancel** button on the main FitlabGui panel (see figure 2.1).

4.11. High Order System Definition

The **HOS Definition** item of the **Data** menu allows the generation of high order systems of either TF or ZPK type that can be saved and then used for low order equivalent system modeling. The corresponding panel is shown in figure 4.7.

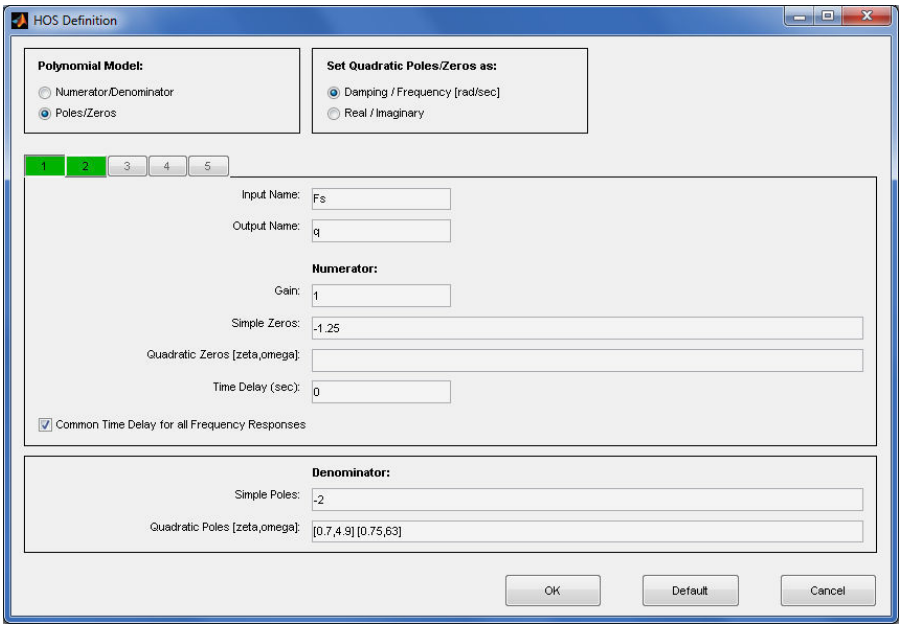


Figure 4.7.: High Order System definition

The top part of the panel allows to switch between systems of the numerator/denominator (TF) or the pole/zero (ZPK) type. For systems of the ZPK type, the quadratic poles and zeros can either be specified as real and imaginary part or through damping and natural frequency of the corresponding pole or zero.

The numerators are defined in the center part of the panel where tabs are used to switch between the different numerators and the denominator is defined in the bottom part. Up to 5 systems with a common denominator can be defined. The names of the input and the outputs are used to create default names for the generated TF or ZPK objects.

For TF systems, the coefficients of the numerator(s) and the denominator have to be given, whereas for ZPK systems, the gain of the numerator(s) as well as the simple and quadratic poles and zeros have to be specified. The time delay can be specified separately for each numerator or a common time delay can be used.

Pressing the **OK** button allows inspecting the generated transfer functions via Bode plots and saving them to a mat-file.

5. Model

FitlabGui allows for parameter estimation of dynamic systems of the following types:

Table 5.1.: **Possible Model Types**

Model Type	System	Method / Cost Function
nonlinear	arbitrary nonlinear system	ML time domain
linear time domain	linear SS system	ML time domain
linear frequency domain	linear SS system	ML frequency domain
polynomial	transfer function	frequency response
linear frequency response	linear SS system	frequency response

The first three model types use a Maximum Likelihood cost function (see appendix B.1 whereas the other two types use the frequency response cost function (see appendix B.2).

The identification of nonlinear and linear systems requires the user to provide a model file whereas the definition for polynomial models is done directly on the corresponding panel (see section 5.5). The next section gives the specifications for the user provided model files. The remaining sections of this chapter describe all items of the **Model** menu of FitlabGui.

5.1. User Model Files

5.1.1. Nonlinear Model

For nonlinear models, the user must provide the calculated output as a function of the values of the parameters and bias parameters, the time vector and the control inputs of the current time section.

Thus the function definition of the model m-file must be:

```
[y] = model(par_val,bias_val,t,u)
```

with

`par_val`	vector with the values of all parameters (as defined in the parameter panel from figure 5.3)
`bias_val`	vector with the values of all bias parameters for the current time section
`t`	time vector for the current time section
`u`	measured control input variables for the current time section
`y`	resulting calculated output variables

Examples for nonlinear model files can be found in the demonstration examples (see chapter 10).

To avoid wasting long computations, FitlabGui first tests each nonlinear model by running it for 5 data points to check correct dimensions of the returned results. Thus, care must be taken that the nonlinear model runs also with only 5 data points.

If the user model contains a Simulink model, the execution of the Simulink model can be sped up significantly, if the Simulink Coder is available. The necessary steps are:

- Open and initialize the model.
- Select all elements (Ctrl-a) on the top level.
- Right click on any element and choose "Create Subsystem" (or Ctrl-g).
- Right click on the created subsystem and choose "Real Time Workshop - Generate s-function". A window will pop up containing a list of the model parameters.
- Mark parameters to be identified as tuneable parameters in this list.
- Click on the build button and save the (renamed) model.

Note that parameters within structures are not shown as model parameters and therefore cannot be identified when using the coder.

Another option to speed up the identification of nonlinear models is the use of code written in c-language and parallelization. Within the Gauss-Newton optimization method (see appendix B.3.1), the calculation of the parameter improvement vector requires the determination of the sensitivity matrix. To determine this matrix, simulations with variations in each of the model parameters that are to be estimated have to be performed. These simulations often account for the largest fraction of the overall computational effort and can be run in parallel as they are independent.

Therefore, parallel versions `residuals4c` and `sensitivity4c` of the corresponding routines, written as c-code, have been developed. These parallel versions are automatically used if the nonlinear model is provided as c-code. For a computer with four kernels, the computation time reduces by about a factor of three compared to the non-parallel version.

A template file `user_model.cpp` for a nonlinear user model written in c-code is provided in the `fitlabDemo` directory. The function header of the model c-file must be identical to that of the example file, the remainder has to be provided by the user. The function name must be `model_simulation`, but the name of the compiled library can be chosen arbitrarily to distinguish between different models. Depending on the MATLAB version used, the user has to compile a 64-bit or 32-bit version of the dynamic linked library. This library is then used in place of a model m-file in the model specification panel from section 5.2.

5.1.2. Linear Model

Linear models both in time and in frequency domain are handled by FITLAB through the use of state space linear time-invariant systems (SS-LTI) as defined in the Control System Toolbox [16]. The user model m-file has to provide the system matrices as a function of the parameter and bias parameter values. Unlike for the nonlinear models, the simulation, i.e. the determination of the calculated response to the measured inputs, is done by FITLAB using routines from the Control System Toolbox.

Thus the m-file for linear models must have the following interface:

```
[A,B,C,D,tau_in,tau_out,bx,by] = model(par_val,bias_val)
```

where

`par_val`	values of all parameters
`bias_val`	values of all bias parameters for the current time section
`A,B,C,D`	system matrices
`tau_in,tau_out`	time delay vectors of the input resp. output variables
`bx,by`	bias vectors of the state resp. observation equations

The use of SS-LTIs allows the specification of different time delays for all input and output variables. If time delays are specified, the length of the time delay vectors `tau_in` and `tau_out` must be equal to the number of control inputs and calculated outputs respectively. Similarly, if bias vectors `bx` and `by` are specified, their length must be equal to the number of equations in the corresponding state and observation equations. Normally, not all elements of `tau_in`, `tau_out` and `bx`, `by` will be estimated at the same time.

Examples for linear models with and without time delays and bias vectors are found in the demonstration examples from section 10.

5.2. Type

The **Model** menu of FitlabGui allows to specify all information pertaining to the identification model. The first menu item **Type** (see figure 5.1 for the corresponding panel) is used to define the model type (see table 5).

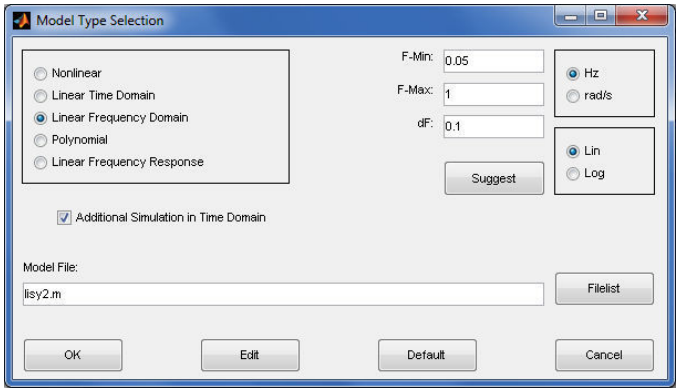


Figure 5.1.: Model specification

For output error frequency domain identification of a linear system, the frequency range and spacing (linear or logarithmic) that will be used for transformation of the data into the frequency domain has to be specified. The **Suggest** button gives the minimum and maximum frequencies possible (based on the sampling rate of the data and the length of the shortest time interval). Transformation of the time domain data into the frequency domain is performed using the routine `fitlab_td2fd` (see section 11.4).

For linear models in the frequency domain as well as linear models identified with the frequency response method, an optional simulation in the time domain can be selected.

For all but polynomial models the model m-file has to be specified. The panel has a **Filelist** button to search for a model-file and an **Edit** button to open the file in the editor window.

For nonlinear models, the model can also be specified as compiled C-code (see section 5.1.1). In this case, the dll-File (on Windows systems) or the shared library (on Linux systems) has to be specified as the model file.

Depending on the chosen model type, only those menu items of the **Model** menu that apply to this model type are active: **Channels** and **Parameters** for all models except polynomial ones, **Polynomial Models** only for transfer function models and

Frequency Response Allocation only for linear models to be identified with the frequency response cost function.

5.3. Channels

Depending on the model type and other options, the output and input variables and maybe also the state variables of the model have to be defined and linked to measured data channels. This is done with the **Channels** menu item that invokes the panel shown in figure 5.2. On the left side of the panel, all channels available in the dataset are shown. On top of the center, three tabs allow to switch between the output, input, and state signals. Depending on the setting of the **Add**/**Replace** radio button, clicking on a signal in the list on the left adds or replaces this signal in the current list (output, input or state) in the center.

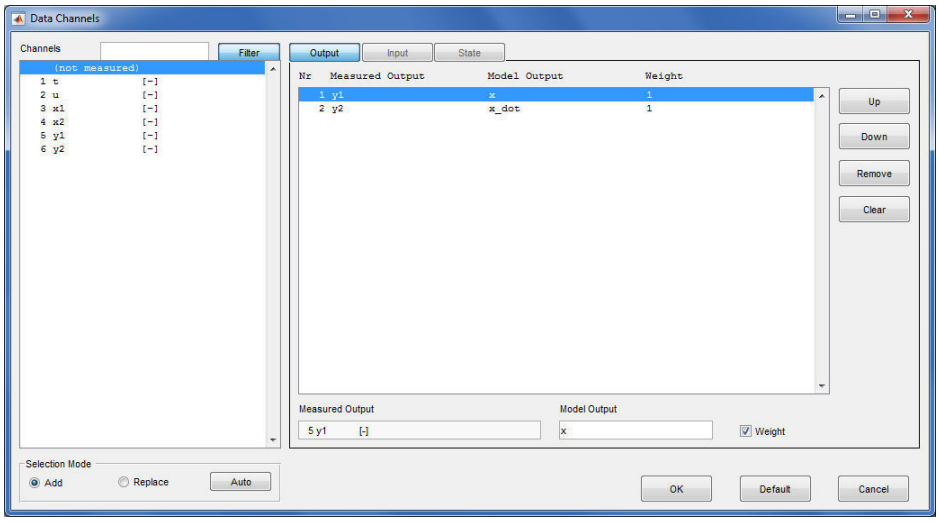


Figure 5.2.: Selection of data channels

If a state-space LTI model has been loaded via the **Load SS-LTI Model** option from section 5.7, the **Auto** button allows to automatically match data channels with model channels if the names are identical.

For all signals, a model name can optionally be specified (default is the signal name) and for output channels, the weighting ('on'/'off', defaults to 'on', see section 11.3) has to be specified. For output signals that are not weighted, the corresponding channel can be empty (entry `(not measured)` at the top of the channel list on the left).

For nonlinear models, the output and input channels must be defined, all input channels must be measured and the states are ignored. For linear models that are identified with the ML method in the time or frequency domain, the output and input channels must be given and all inputs must be measured. Data channels for the states are only necessary if a startup calculation is to be performed.

For linear models that are identified with the frequency response method, only the model names of the input and output signals need to be given. Model names for the states are optional and data channels for the inputs and outputs are only necessary if an additional simulation in the time domain is to be performed.

For all model types that use linear models, the defined output, input, and state model names are used to label the outputs, inputs, and states in the SS-LTI that is returned as a result of the identification (output argument `lti` of FITLAB, see section 11).

5.4. Parameters/Biases

The model parameters are divided into parameters and bias parameters. Parameters are valid for all time intervals, that are to be evaluated, whereas bias parameters have the same function but different numerical values for each time interval. Examples for bias parameters are initial conditions or trim settings.

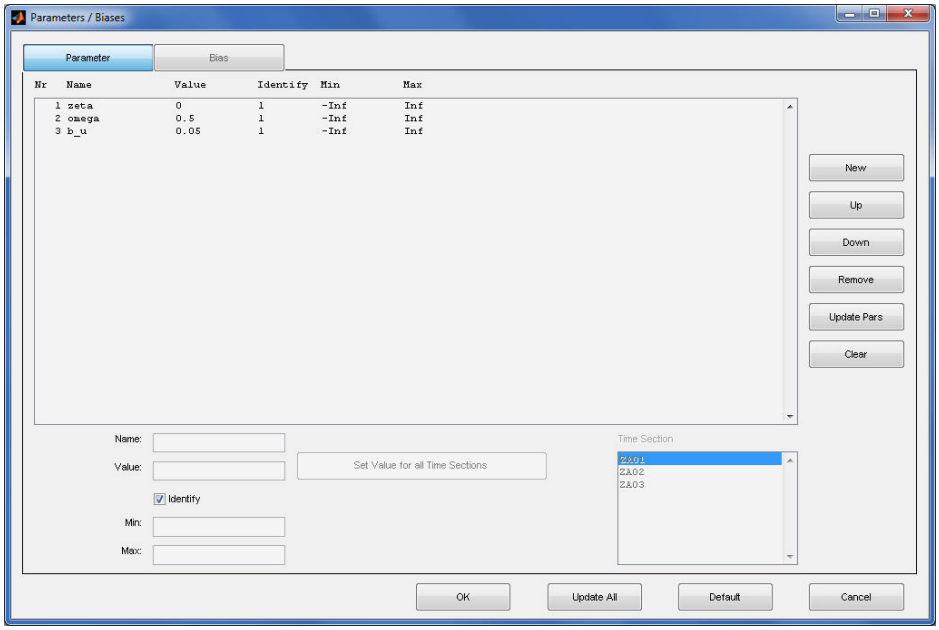


Figure 5.3.: Specification of parameters and bias parameters

Parameters and bias parameters are defined through the panel shown in figure 5.3, that is invoked through the **Parameters/Biases** item of the **Model** menu. For each parameter, a name, the (starting) value, whether the parameter is to be identified, and the minimum and maximum value allowed have to be specified. The **New** button on the right side inserts a new parameter after the currently selected one. Defaults are a starting value of 1, identify on, and limits of -Inf and Inf. For moving parameters or changing settings, several parameters can be selected together.

The definition of the bias parameters is very similar and invoked by selecting the **Bias** tab on the panel in figure 5.3. For bias parameters, values have to be specified for all time sections. This can be done by either selecting the corresponding time sections from the list on the bottom right or by using the **Set Value for all Time Sections** button.

Once an identification run has been performed, the **Update Pars** or **Update Bias** buttons on the right side of the panel allow to update the values for all parameters or biases (depending on the tab currently chosen) with the identified values. The **Update All** button on the bottom of the panel updates all parameters and all biases.

5.5. Polynomial Models

Polynomial models are defined via the **Polynomial Model** item of the **Model** menu which is only active once a polynomial model has been selected in the **Type** panel. The corresponding panel is shown in figure 5.4. The top part of the panel allows to switch between numerator/denominator and pole/zero models.

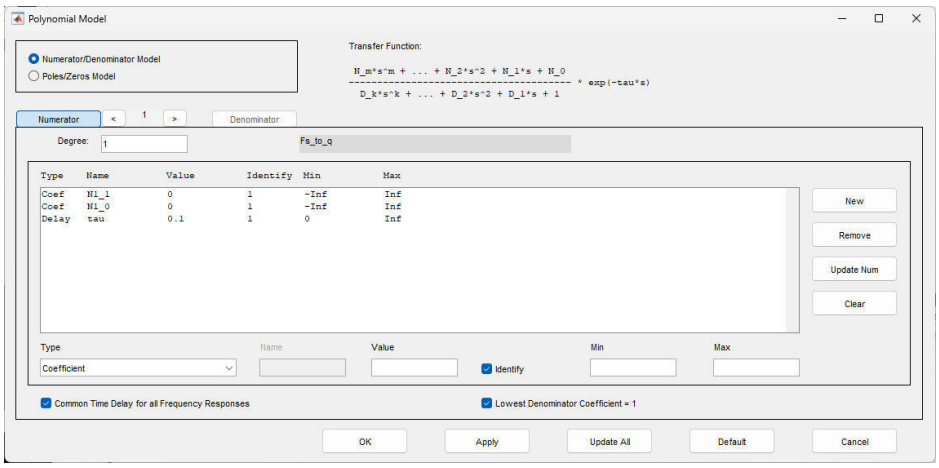


Figure 5.4.: Specification of polynomial models

Below, tabs allow to switch between the different numerators and the denominator. For the numerators, the name of the corresponding frequency response as selected from the **Load Frequency Responses** panel (see section 4.9) is shown. Frequency responses that are currently switched off are marked as inactive. Once an identification run is started, only the parameters corresponding to those frequency responses with their weighting set to on are estimated.

For numerator/denominator models, only the degrees of the numerator(s) and the denominator have to be given and the parameter list below is automatically filled with the corresponding parameters. If one or more time delays are needed, they can be added via the **New** button and then selecting **Time Delay** from the pull-down list. Starting values for the coefficients will normally not be available and if none are given, FITLAB will automatically use a startup calculation (see appendix B.1.4). This startup algorithm, however, requires that all frequency responses have identical frequency axes and that the time delays are either all fixed or all free to be identified.

When modeling with the pole/zero type is selected, all parameters have to be defined via the **New** button. The corresponding pull-down list allows to select the parameter types **Gain**, **Simple Root**, **Quadratic Root**, and **Time Delay**. Selecting the type **Quadratic Root** automatically adds two parameters, namely the damping and frequency of the quadratic root. Parameter names as well as minima and maxima can also be specified here. Pressing the **Apply** button sorts the parameters in the order necessary for polynomial models (see section 11.1) and checks consistency.

When the model type is switched from numerator/denominator to pole/zero or vice versa, model type conversion is performed. However, when switching back and forth, the order of the poles and zeros and the names and limits are not retained but set to default values.

Once an identification run has been performed, the **Update Num** or **Update Den** buttons on the right side of the panel allow to update the values for the corresponding numerator or denominator parameters (depending on the tab currently chosen) with the identified values. The **Update All** button on the bottom of the panel updates the parameters of all numerators as well as the denominator.

5.6. Frequency Response Allocation

For linear models that are identified with the frequency response method, the frequency responses that are to be matched have to be connected with the input/output combinations of the model. This is done with the **Frequency Response Allocation** item from the **Model** menu that invokes the panel shown in figure 5.5.

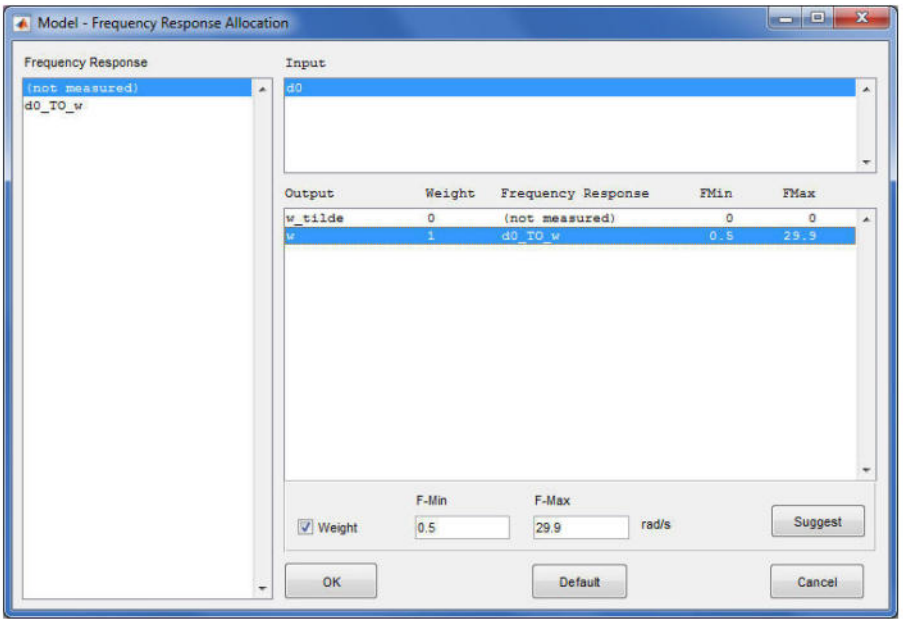


Figure 5.5.: Frequency response allocation panel

On the left side of the panel, all active frequency responses are listed. On the right side, the top window allows to switch between the different model inputs. The bottom window allows to specify the measured frequency responses and the frequency range of fit for all model outputs. An output must first be marked on the right side and the corresponding frequency response then be selected from the list on the left. For frequency responses that are not available, the entry `(not measured)` can be used.

By default, the frequency range of the selected frequency response corresponds to the range specified in the **Load Frequency Responses from File** panel (see section 4.9), but a subinterval of this frequency range can be chosen at the bottom of the panel. For each frequency response, the weighting can be switched 'on' or 'off' with the checkbox at the bottom. Only frequency responses that are 'on' are weighted in the cost function. For frequency responses that are not measured, the weighting is automatically set to 'off'. A frequency range can still be specified and is used in the generation of the corresponding model output.

5.7. Load SS-LTI Model

The menu item **Load SS-LTI Model** allows to read a MATLAB SS-LTI (state-space linear time-invariant) object from a mat-file for simulation within FitlabGui and handling

qualities analysis (if Heli-HQ is installed). The user has to select the file and can then choose one of the SS-LTIs in the file.

Upon pressing the **OK** button, a linear model is automatically built from the SS-LTI. The default model type (see section 5.2) is a 'Linear Time Domain' model with the model file `general_SS_model.m` which is provided with FitlabGui. The model input, output, and state channel names (see section 5.3) are set to those of the LTI model. If no names are found, default names are used. The number of state, input, and output variables as well as all system matrix elements and time delays are converted into model parameters (see section 5.4). Bias parameters for all state and observation equations are created with starting values of zero. The **identify** checkbox for all parameters and bias parameters is set to 'off'.

For a simulation with the model in the time domain, time domain data has to be read via the **Load Time Sections from File** panel (see section 4.2) and the model channels have to be matched with the measured channels via the **Channels** panel (see section 5.3).

To create frequency responses from the model, the model type has to be changed to 'Linear Frequency Response'. Frequency ranges have to be specified in the **Frequency Response Allocation** panel (see section 5.6) and corresponding measured frequency responses can be specified.

6. Execution

Once the measured data and all necessary items from the **Model** menu have been specified, the FITLAB parameter estimation program can be started from the **Execution** menu.

6.1. Run Simulation

The menu item **Run Simulation** performs a simulation. As no parameters are varied, no parameter values are shown and only the resulting cost function is displayed.

6.2. Run Estimation

Run Estimation starts an estimation run. Depending on the **Options** settings (see section 6.4), the amount of output after each iteration and at the end of the identification run varies.

Should the user wish to cancel an executing identification run, either because he made a mistake or because the identification does not progress in the right direction, he can do so by pressing the **Cancel** button on the main FitlabGui panel (see figure 2.1). This button becomes enabled while the FITLAB program is running and pressing this button halts execution of FITLAB almost immediately. If an estimation was canceled, the user is asked whether the identification results from the last completed iteration shall be saved or discarded.

6.3. Restart

The menu item **Restart** allows to continue an estimation run that has ended because the maximum number of iterations was reached. The estimation is restarted with the parameter values from the last run and continued with the settings currently specified in **Options**. If startup iterations had been specified for the first run, the number of startup iterations should usually be set to zero before a restart.

6.4. Options

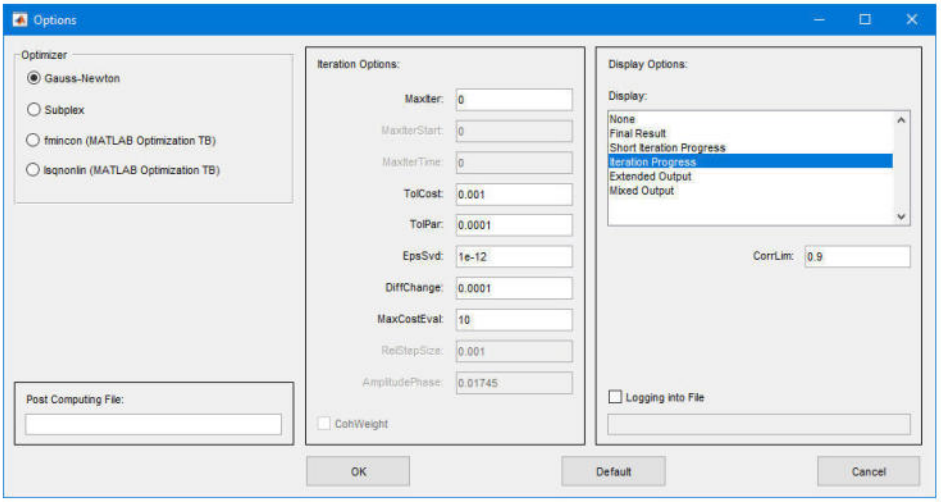


Figure 6.1.: Specification of FITLAB options

If any options have to be altered from the default values (see table 11.1), this can easily be done via the menu item **Options** that invokes the panel shown in figure 6.1. The optimization method is chosen on the left of the panel. In the center, the iteration options are specified. Depending on the type of system to be identified and on the choice for the optimization method, only the options pertaining to this selection can be altered.

The display options are chosen on the right side of the panel. The different options are

None	no output
Final Result	display only the final results
Short Iteration Progress	display cost function and parameter values for each iteration, display bias parameter values only for the final results
Iteration Progress	display cost function as well as parameter and bias parameter values for each iteration
Extended Output	same as **Iteration Progress** plus display of the cost for each time interval and the (root) mean square error for each output variable
Mixed Output	same as **Short Iteration Progress** during iteration and same as **Extended Output** for final display

If a logfile is newly selected, its name defaults to the name of the current project with the file extension log. The **Default** button sets all options to the defaults from table 11.1.

On the lower left of the panel, a **Post Computing** routine can be specified that is automatically executed once an identification run has ended. For this routine and for other postprocessing, the identification results can be accessed via the variables `par_id`, `bias_id`, `data_id`, `fdata_id`, `cost`, and `lti` (see chapter 11), that are written to the MATLAB workspace.

7. Plotting

The **Plotting** menu provides the following plots to visualize the measured data and to illustrate the identification results:

- **Quick Plot Time Domain**
- **Report Time Domain**
- **Cross Plot**
- **Azimuth Plot**
- **Density Plot**
- **Geographic Plot**
- **Hardread Plot**
- **Quick Bode Plot**
- **Report Bode Plot**
- **Spectral Plot**
- **Mismatch Envelope Plot**
- **Quick Plot Frequency Domain**
- **Report Frequency Domain**

Depending on whether time domain data and/or frequency responses have been defined and depending on the current choice of the model type (see section 5.2), only a subset of the plotting routines is available. The first seven plots work on time domain data, whereas the next four plots work on frequency response data. The last two plots use data that has been transformed from the time domain into the frequency domain by the routine `fitlab_td2fd` (see section 11.4).

All plot windows produced by FitlabGui are normal MATLAB figures. Therefore, if the user wants to alter the appearance of any of the plots (e.g. to change the scaling or the color of individual lines), he can do so by using the options of the figure menus or by issuing the corresponding commands from the MATLAB command window. In all FitlabGui plots that have a common x-axis (either time or frequency axis), the diagrams are linked such that zooming within one diagram automatically adjusts the scaling of the x-axis in the other diagrams. Furthermore, for the frequency domain plots, the line properties (color, line style, etc.) in corresponding diagrams are linked so that e.g. changing the line color of one frequency response in the amplitude diagram automatically applies the same changes in the phase and coherence diagrams.

7.1. Quick Plot Time Domain

For the **Quick Plot Time Domain**, individual signals can be selected from the list of all available channels. To make handling of files with many data channels easier, a **Filter** button allows to filter the channel names by giving one or more strings (separated by blanks) that have to be part of the signal name. The use of '?' and '*' as wildcards is possible. Once all channels have been selected, the **Plot** button initiates a strip-chart plot of the concatenated data from all time sections that can be used for quick inspection of the test data.

If only one time section is plotted, the time axis can be chosen as either the relative time (starting at zero) or the originally recorded daytime values (in seconds or in hh:mm:ss format). For several concatenated time sections, an artificial relative time axis is always used.

7.2. Report Time Domain

More elaborated plots, that allow up to four channels per diagram and several plot pages with up to nine diagrams per page, are created with the **Report Time Domain** menu item. Once a simulation or an identification has been performed, this plot type also allows to create plots that compare the model outputs with the corresponding measured signals. Furthermore, the errors between the model output and the measured output variables are available as additional signals and thus can be plotted also.

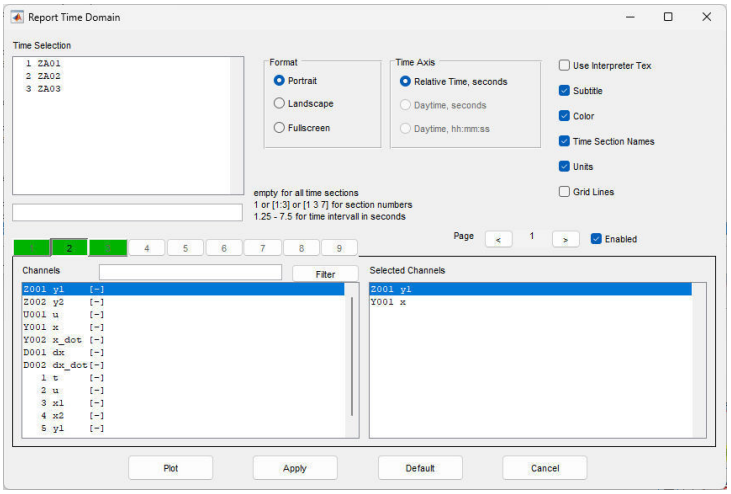


Figure 7.1.: Panel for the time domain report plot

Figure 7.1 shows the panel invoked by **Report Time Domain**. The time sections available are listed on the top left of the panel. Below this list, the time interval(s) to be plotted can be selected as a subset of all time intervals. Alternatively, a time interval to be plotted can be specified. On the top right, some general settings can be chosen that are valid for all plot pages. The options for the time axis are the same as for the **Quick Plot**.

In the bottom half of the panel, the arrows on the right allow to switch between plot pages. Each plot page can be enabled or disabled individually (only the enabled pages are plotted, but the settings for all pages are saved in the project file). For each plot page up to 9 diagrams can be chosen by the tabs on the left and up to 4 channels can be selected for each diagram.

The list of all available data channels can be filtered in the same way as for the quickplot (see section 7.1). Once a simulation or an identification has been performed that generates time domain data, the measured outputs (`Zxxx`), measured inputs (`Uxxx`), model outputs (`Yxxx`), and model errors (`Dxxx`) are found at the top of the channel list and are thus readily available for plotting.

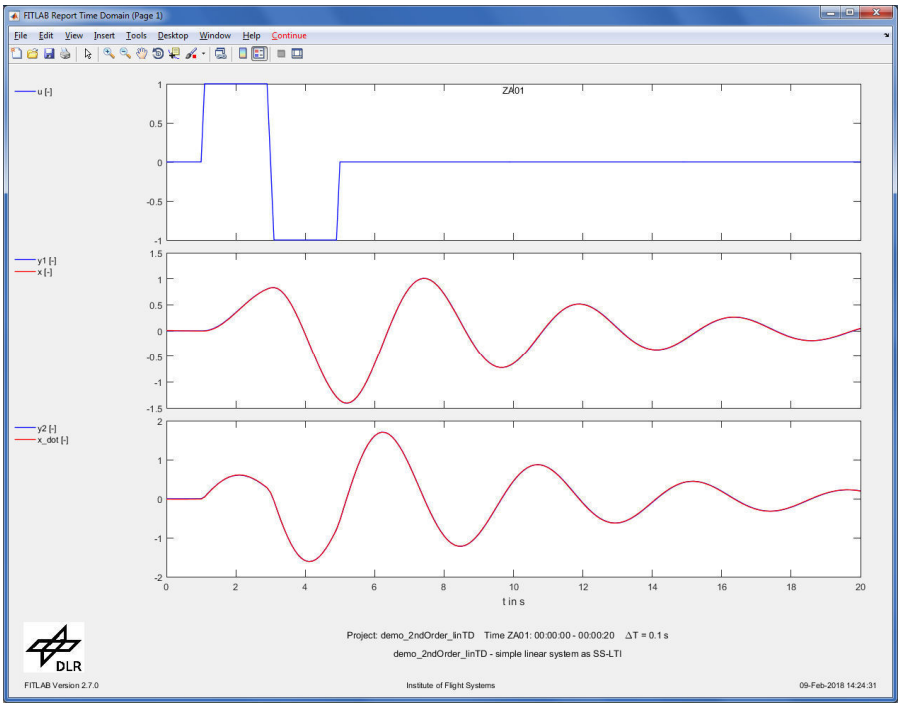


Figure 7.2.: Example for time domain report plot

Figure 7.2 shows an example of a time domain report plot. If only a part of the overall time axis available was selected for the plot (either by specifying a time section or a time interval), a **Continue** item appears in the menu bar on top of the plot window.

Clicking on this item allows to step through the data either time section by time section or in time intervals like the first one specified.

7.3. Cross Plot

The menu item **Cross Plot** allows to create cross plots from pairs of data channels. The channels to be displayed on the x- and y-axis are selected in the same way as for the other time domain plots. Up to 10 pairs of channels can be plotted and each curve can be approximated by a regression of up to 3rd order.

The selection of the time interval(s) for which the data is to be plotted is the same as for the **Report Time Domain** plot and the same **Continue** option exists.

Once a simulation or identification resulting in time domain data has been performed, all measured and calculated model variables as well as the errors are available for plotting.

7.4. Azimuth Plot

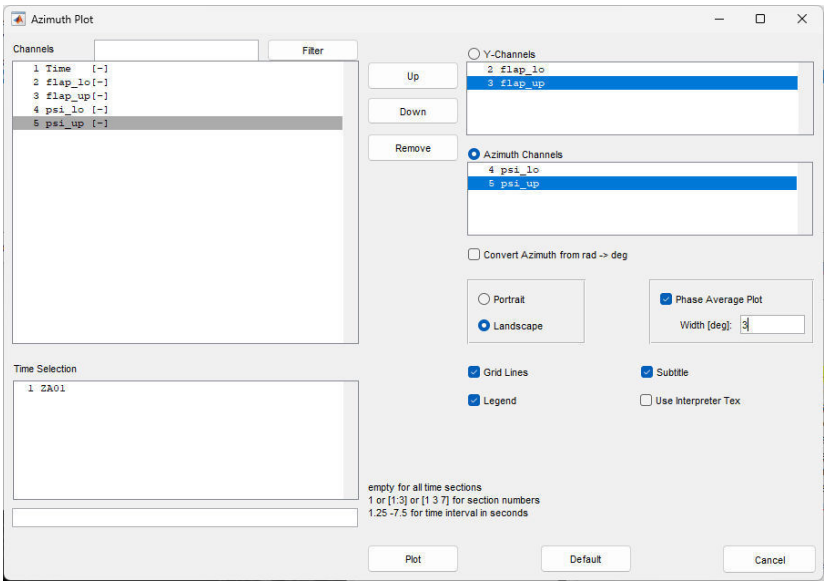


Figure 7.3.: Panel for the azimuth plot

The **Azimuth Plot** is a special version of the cross plot, where data is plotted versus an azimuth angle and the wrapping at 360 deg is accounted for. The corresponding

panel is shown in figure 7.3. On the upper right, the signals to be displayed on the y-axis and the corresponding azimuth signals have to be selected from the list of all available signals on the left.

The selection of the time interval(s) for which the data is to be plotted is the same as for the **Cross Plot**.

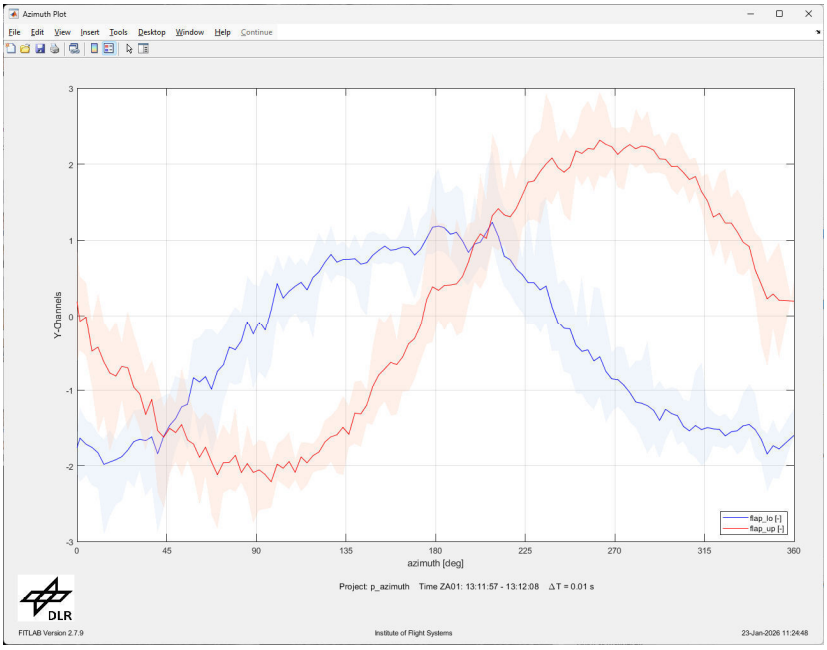


Figure 7.4.: Example for phase average plot

Instead of plotting the measured data directly, checking the **Phase Average Plot** option averages the data over several revolutions and displays the mean value as a solid line surrounded by a shaded area marking the minimum and maximum values. A width value has to be specified and the azimuth range of 0-360 deg is then subdivided into bins of the width and the data averaged within each bin. Figure 7.4 shows an example for a phase average plot with a width of 3 deg.

7.5. Density Plot

The **Density Plot** allows to analyze the distribution of two signals versus each other. The corresponding panel is shown in figure 7.5. The channels to be displayed on the x- and y-axis are selected in the same way as for the other time domain plots. One or more time sections can be selected from the list of active time sections which is displayed on the right. Between 10 and 1000 bins can be selected for partitioning the

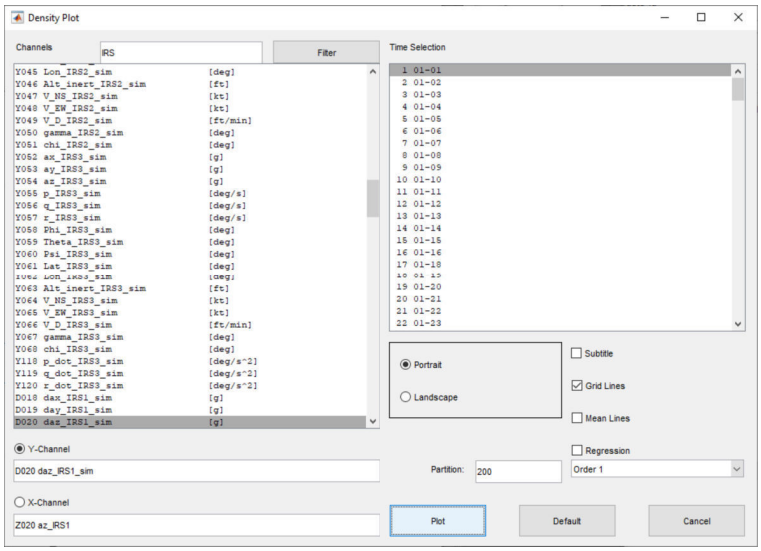


Figure 7.5.: Panel for the density plot

data. Optionally the mean values of the two variables and a regression line can be displayed.

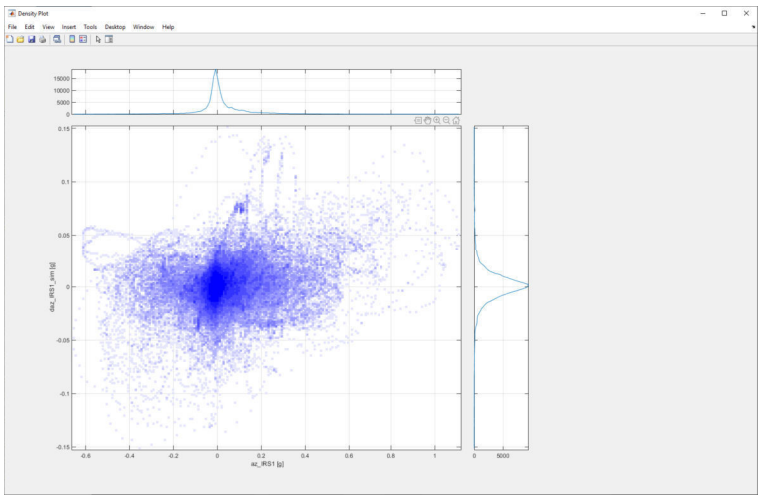


Figure 7.6.: Example for a density plot

Figure 7.6 shows an example for a density plot. The center diagram displays the binned data where the color intensity corresponds to the amount of data in the corresponding bin. In the top and right diagrams, the data distribution of the x- and y-channels is displayed.

7.6. Geographic Plot

For Matlab version 2018b and newer, the **Geographic Plot** allows to plot a flight path over a geographic map if latitude and longitude are available as measured signals. Latitude and longitude have to be given in degrees or converted to degrees using the **Unit Conversion** option in the **Data** panel (see section 4.5).

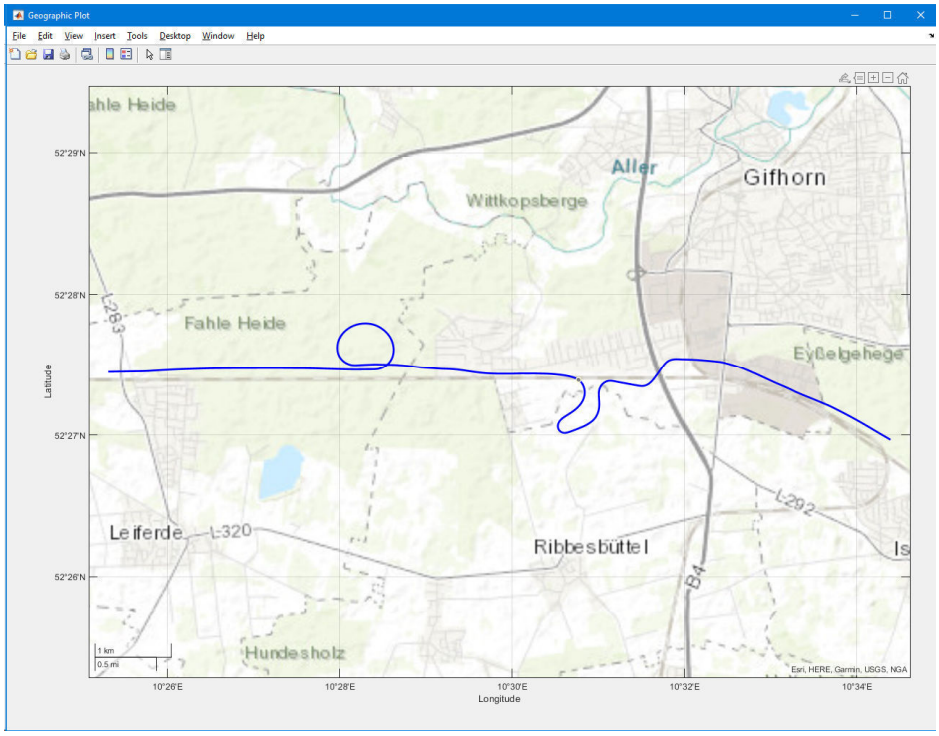


Figure 7.7.: Example for a geographic plot

More than 10 different maps as described in the documentation for the Matlab function `geobasemap` are available. Only the 'darkwater' map is included with Matlab, all other maps require internet access. Figure 7.7 shows an example for a geographic plot using the 'topographic' map.

7.7. Hardread Plot

For flight data saved in R-CDF format, it is sometimes useful to plot only the measured data and no interpolated values. This can be done with the **Hardread Plot**. The

options and the plot itself are similar to a **Report Time Domain** plot, but only one plot page can be defined and the data is by default plotted with markers instead of lines.

7.8. Quick Bode Plot

Once frequency responses have been defined via the **Load Frequency Responses from File** panel (see figure 4.5), the **Quick Bode Plot** allows to create Bode plots of all currently active frequency responses. The first diagram shows the amplitude of the frequency responses in dB and the second the phase in radians or degrees on a semilog scale.

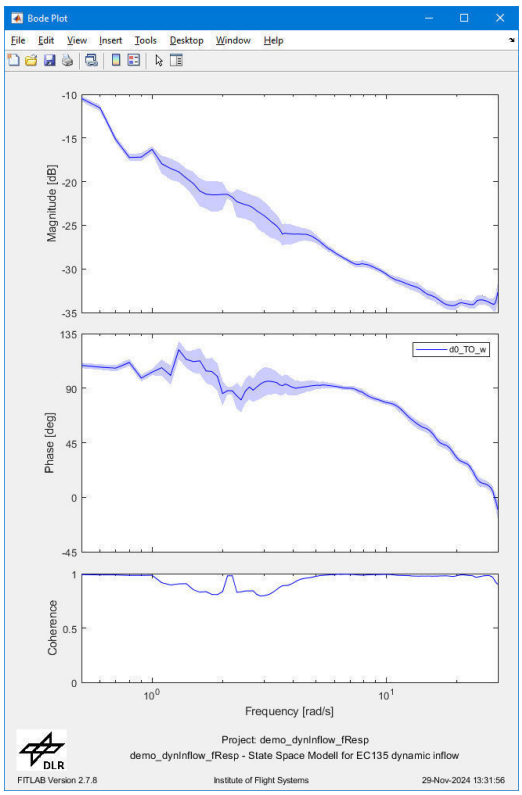


Figure 7.8.: Example for a quick Bode plot with uncertainty bounds

If the frequency responses have a coherence, the coherence can be displayed in a third diagram below the amplitude and phase and, if desired, the uncertainty bounds for magnitude and phase (see A.6) can be shown. Figure 7.8 shows an example of a quick Bode plot with uncertainty bounds.

7.9. Report Bode Plot

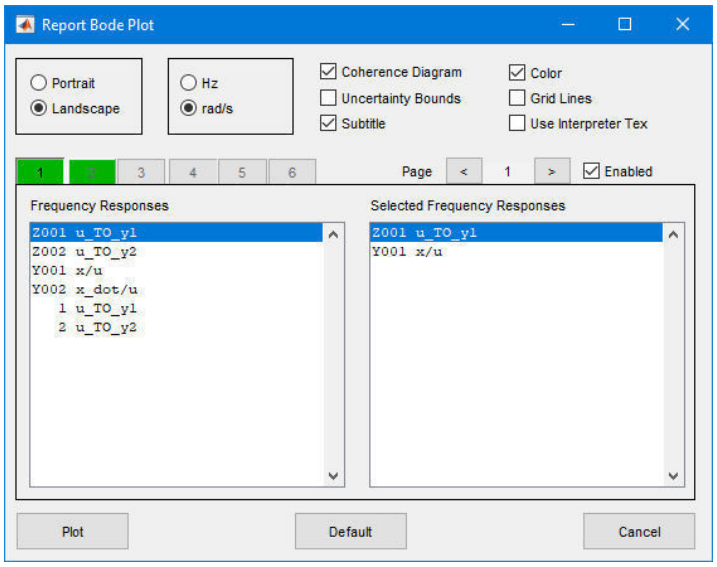


Figure 7.9.: Panel for the report Bode plot

Similar to the **Report Time Domain** plot, the **Report Bode Plot** allows to define several plot pages where each page can contain up to six frequency response diagrams with up to four frequency responses in each diagram. The corresponding panel is shown in figure 7.9.

On the top of the panel, some settings that are valid for all plot pages, are selected. On the right, arrows allow to switch between plot pages and each page can individually be enabled or disabled. On the left, tabs are used to select the diagrams on each page and the frequency responses to be displayed in each diagram are selected from the list of all frequency responses that are available. The resulting plots are scaled such that all plots on all active pages have the same frequency scaling.

Once a simulation or identification with the 'polynomial' or 'linear frequency response' model type has been performed, the measured (`Zxx`) and calculated (`Yxx`) frequency responses are available at the top of the selection list. The left part of figure 7.10 is an example for a Bode plot that illustrates the match of an identified model.

7.10. Spectral Plot

When frequency responses were generated with Tischler's method (see 4.10), the input and output spectra, the cross spectrum and the random error are saved in addition to amplitude, phase and coherence of the frequency response. The **Spectral Plot** allows to plot all of this information.

Care should be taken, that when MISO conditioning was used in the frequency response generation (by selecting secondary inputs), the spectra that are saved with the frequency response are the conditioned spectra. To derive, for example, the amplitude spectrum of a control input signal, the unconditioned spectrum is of interest and thus a frequency response that was generated without secondary input must be used to derive this information.

If the optional add-on Heli-HQ [12] is installed, a power spectrum plot can be generated with the **RMS / cutoff frequency** menu item.

7.11. Mismatch Envelope Plot

In the fixed-wing military handling qualities criteria standard MIL-STD-1797 [18], mismatch criteria have been defined to evaluate the match between an actual frequency response and its low-order equivalent system model. The boundaries correspond to limits on the maximum unnoticable added dynamics, beyond which a pilot will detect a deviation in the response characteristics. A plot showing the approximation errors in comparison to these limits can be created through the **Mismatch Envelope Plot**.

As long as no simulation or identification that creates frequency responses has been performed, the plotting routine assumes that the selected measured frequency response already contains the error that is to be evaluated. Otherwise, the difference between measured frequency response and the corresponding model output is calculated and displayed in the mismatch plot. The right part of figure 7.10 shows an example for such a plot.

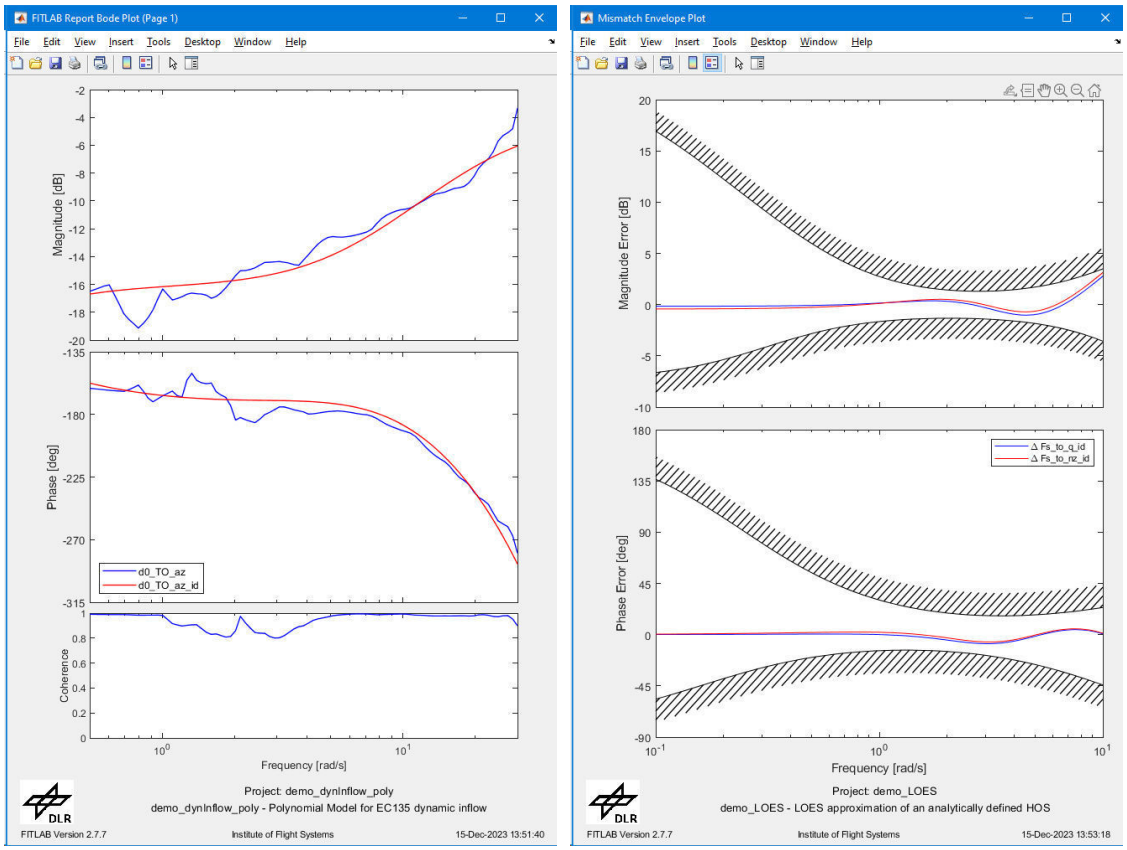


Figure 7.10.: Example for report Bode plot and mismatch envelope plot

7.12. Quick Plot Frequency Domain

Both the **Quick Plot Frequency Domain** and the **Report Frequency Domain** are only available if the model type 'linear frequency domain' has been selected in the panel from figure 5.1.

The **Quick Plot Frequency Domain** takes the time domain data, transforms it into the frequency domain using the utility function `fitlab_td2fd` (see section 11.4), and plots the results versus frequency. For each signal, an amplitude diagram and a phase diagram is generated.

7.13. Report Frequency Domain

The **Report Frequency Domain** allows display the same type of frequency domain data on several plot pages with up to four diagrams each. The panel is similar to the **Report Time Domain**. For each plot page, only data from one time section can be displayed, but different plot pages can correspond to different time sections.

Once a frequency domain identification with the ML output error method has been performed, this plot type allows to illustrate the match between the measured outputs and the model outputs in the frequency domain.

8. Help

The **Help** menu has five items:

FitlabGui Document allows to browse a pdf-version of this user's guide.

HQ-Tools Document allows to browse a pdf-version of the HQ-Tools user's guide [12]. (This item is only active if the HQ-Tools add-on is installed.)

RCDF Document allows to browse a pdf-version of the RCDF description [15].

About FitlabGui displays the version of FitlabGui used.

Version Information opens a window that lists the updates that were made between the last versions of FitlabGui.

9. Utilities

Several utility functions that can be used in the channel arithmetic are provided with FitlabGui:

`arinclabel`	display the ARINC label of a signal
`bitsignal`	extract a bitsignal from a discrete signal
`cutfilt`	cutoff filter
`differ`	numerical differentiation
`glatt`	remove outliers from a measured signal
`myunwrap`	remove jumps of $\pm 2\pi$ from a signal
`smooth`	smoothing filter

All routines are delivered as m-files and are thus only described briefly in the following sections.

9.1. ARINC Label

The function call `LabelData = arinclabel(Data)` allows to determine the ARINC label of a signal.

`Data`	column vector with channel data
`LabelData`	ARINC label data

9.2. Bitsignal

The function call `BitData = bitsignal(Data,BitMask)` allows to extract a bitsignal from a discrete.

`Data`	column vector with channel data
`BitMask`	String of length 8 containing the bitmask
`BitData`	Bit signal data

9.3. Cutoff Filter

The function `cutfilt` takes the fourier transform of a time domain signal and sets all elements above the cutoff frequency to zero. An inverse fourier transformation returns the signal to the time domain. Options allow the signal to be differentiated or integrated.

The call is `[output,realcutoff] = cutfilt(input,time,cutfreq,o1,o2,o3)` with

`input`	matrix with data to be filtered in columns
`time`	corresponding time axis or sampling interval
`cutfreq`	cutoff frequency, rad/s
`o1`	filter order `Inf` or `0` is a complete cut-off (the default)
`o2`	order of the trend to be removed `0` = remove constant, `1` = remove linear trend, default = `1`
`o3`	selection of differentiation (`1`) or integration (`-1`) of the signal. Default is no differentiation or integration (`0`
`output`	matrix with filtered signals in columns
`realcutoff`	frequency where the response is half the original response (-6dB point), rad/s

9.4. Numerical Differentiation

The function `differ` performs numerical differentiation using the five point method. This is the routine that is used when differentiation is chosen as postprocessing in the channel arithmetic (see section 4.6).

The function call is `yp = differ(dt,y)` with

`dt`	sampling interval
`y`	signal to be differentiated
`yp`	differentiated signal

9.5. Outlier Removal

The function `glatt` allows to remove outliers (spikes) from measured data.

The function call is `y = glatt(x,dx_lim,x_min,x_max)` with

`x`	column vector with channel data
`dx_lim`	max. difference from one x-value to the next (default = 1)
`x_min`	minimum value of x without spikes
`x_max`	maximum value of x without spikes
`y`	`x` without the spikes

`glatt` replaces `remove_spikes` which is still included for compatibility.

9.6. Remove Jumps of 2π

The function `myunwrap` allows to remove jumps of $\pm 2\pi$ from a measured angular signal, e.g. the heading angle. Unlike the MATLAB function `unwrap`, this function also removes jumps that have been interpolated from data sampled at a lower rate.

The function call is `y = myunwrap(x,dx_lim)` with

`x`	vector of input data (in rad)
`dx_lim`	max. difference from one value to the next (default: $\pi/6$ = 30 deg)
`y`	return vector with jumps removed

`myunwrap` replaces `remove_2pi_jumps` which is still included for compatibility.

9.7. Data Smoothing

The function `smooth` implements data smoothing with a symmetric filter that was widely used in the tool DA2 [19]. The first and last 2 points of the signal remain unfiltered. The next 5 points are filtered with a 5-point filter and for the rest of the signal, the full 15-point filter is used.

The function call is `y = smooth(x)` with

`x` column vector with data
`y` return vector with smoothed data

10. Examples

Several identification examples are provided with FitlabGui that can either be started by loading the corresponding project file from the `fitlabDemo` subdirectory or by typing `fitlab_demo` on the command line to start the command line demo program. The examples illustrate the capabilities and different options of the program and are described in the following sections.

10.1. Linear 2nd Order System

The response of a simple linear 2nd order system with natural frequency ω_n and damping ζ to an external input u is described by the following differential equation:

$$\ddot{x} = -\omega_n^2 x - 2\zeta\omega_n \dot{x} + K_u u$$

To simulate this system with the solvers for ordinary differential equations (ODEs) it must be transformed into two first order differential equations. Assuming that both x and $\dot{x}$ are measured, this yields the following state and observation equations in matrix notation

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ K_u \end{bmatrix} \cdot u$$

$$\begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \cdot u$$

Four demonstration examples use simulated data for this 2nd order system to illustrate the basic features of the program:

- In `demo_2ndOrder_nonlin.mat`, the model is implemented with a nonlinear model that uses a Simulink state-space system block.
  (This corresponds to the first example of the command line demo.)
- In `demo_2ndOrder_linTD.mat`, the same equations are formulated as a linear system. Data from three time sections is used and different bias parameters are estimated. Also, the startup mode is used.
  (This corresponds to the second example of the command line demo.)

- `demo_2ndOrder_linFD.mat` is basically the same as before but now using data in the frequency domain. To obtain values for the bias parameters, one final iteration is run in the time domain.
  (This corresponds to the third example of the command line demo.)
- In `demo_2ndOrder_fResp`, the frequency response method is used to identify the system parameters. To obtain values for the bias parameters, some final iterations are run in the time domain.
  (No corresponding example is available in the command line demo.)

10.2. Equivalent Models

For handling qualities research, linear models that describe the reaction of the augmented aircraft to pilot inputs are often used. The demonstration for FitlabGui contains two such examples using data from the ATTAS research aircraft:

- `demo_eqLon.mat` performs modeling of the longitudinal motion
  (see example 4 of the command line demo)
- `demo_eqLat.mat` performs modeling of the lateral motion
  (see example 5 of the command line demo)

For the longitudinal motion, the reaction of the aircraft to pilot pitch command δ_{PC} is assumed to be described through

$$\begin{bmatrix} \dot{u} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\Theta} \end{bmatrix} = \begin{bmatrix} X_u & X_\alpha & X_q - u_0\alpha_0 & -g\cos\Theta_0 \\ Z_u & Z_\alpha & Z_q + 1 & -g\sin\Theta_0/u_0 \\ M_u & M_\alpha & M_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} u \\ \alpha \\ q \\ \Theta \end{bmatrix} + \begin{bmatrix} X_{PC} \\ Z_{PC} \\ M_{PC} \\ 0 \end{bmatrix} \cdot \delta_{PC}$$

Output variables are the states and additionally the longitudinal and vertical accelerations. Therefore, the observation equations are in matrix notation:

$$\begin{bmatrix} u \\ \alpha \\ q \\ \Theta \\ a_x \\ a_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ X_u & X_\alpha & X_q & 0 \\ u_0 Z_u & u_0 Z_\alpha & u_0 Z_q & 0 \end{bmatrix} \cdot \begin{bmatrix} u \\ \alpha \\ q \\ \Theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ X_{PC} \\ u_0 Z_{PC} \end{bmatrix} \cdot \delta_{PC}$$

In the ATTAS data set, no angle of attack is provided, so the weighting for this variable is set to off for the evaluation. Bias parameters are estimated for some of the equations

to improve the match. Identification can be performed both in the time and in the frequency domain.

The corresponding state equations for the lateral-directional motion due to roll and yaw control inputs, δ_{RC} and δ_{YC}, are

$$\begin{bmatrix} \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} Y_\beta & Y_p + \alpha_0 & Y_r - 1 & g\cos\Theta_0/V_0 \\ L_\beta & L_p & L_r & 0 \\ N_\beta & N_p & N_r & 0 \\ 0 & 1 & \tan\Theta_0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \beta \\ p \\ r \\ \Phi \end{bmatrix} + \begin{bmatrix} Y_{RC} & Y_{YC} \\ L_{RC} & L_{YC} \\ N_{RC} & N_{YC} \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \delta_{RC} \\ \delta_{YC} \end{bmatrix}$$

Measurements are usually the state variables and the lateral acceleration

$$\begin{bmatrix} \beta \\ p \\ r \\ \Phi \\ a_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ V_0 Y_\beta & V_0 Y_p & V_0 Y_r & 0 \end{bmatrix} \cdot \begin{bmatrix} \beta \\ p \\ r \\ \Phi \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ V_0 Y_{RC} & V_0 Y_{YC} \end{bmatrix} \cdot \begin{bmatrix} \delta_{RC} \\ \delta_{YC} \end{bmatrix}$$

In the ATTAS data set the sideslip measurement is missing and therefore the weighting for this variable has to be switched off. Also, only roll control δ_{RC} was used, so all yaw control derivatives are not identifiable and have thus to be set to zero. As for the longitudinal motion, both time and frequency domain identification can be performed.

10.3. Compatibility Check Using Flight Path Reconstruction

As an example for a truly nonlinear model, a simple compatibility check using X-31 flight test data is provided in `demo_FPR.mat` or example 6 of the command line demo. The compatibility check uses the 6-Dof kinematic equations of aircraft motion as state equations

$$\begin{aligned} \dot{u} &= a_x - g\sin(\Theta) + rv - qw \\ \dot{v} &= a_y + g\sin(\Phi)\cos(\Theta) + pw - ru \\ \dot{w} &= a_z + g\cos(\Phi)\cos(\Theta) + qu - pv \end{aligned}$$

$$\begin{aligned} \dot{\Phi} &= p + (q\sin(\Phi) + r\cos(\Phi))\tan(\Theta) \\ \dot{\Theta} &= q\cos(\Phi) - r\sin(\Phi) \\ \dot{\Psi} &= (q\sin(\Phi) + r\cos(\Phi))/\cos(\Theta) \end{aligned}$$

$$\dot{h} = u\sin(\Theta) - v\sin(\Phi)\cos(\Theta) - w\cos(\Phi)\cos(\Theta)$$

Inputs are the measured linear accelerations a_x, a_y, a_z and the angular rates p, q, r. Output variables are the measured airspeed and the airflow angles

$$V = \sqrt{u^2 + v^2 + w^2}$$
$$\alpha = \arctan(w/u)$$
$$\beta = \arctan(v/\sqrt{u^2 + w^2})$$

as well as the aircraft attitude angles Φ, Θ, Ψ and the altitude h.

In a compatibility check, the above system of kinematic equations is driven by the measured inputs and the output of the model is compared to the measured output variables. If discrepancies are found, the different measurements are not compatible and correction terms for one or more of the measurements have to be identified. In the present case it is assumed that the measured inputs have unknown zero offsets, i.e.

$$a_x = a_{x,m} - \Delta a_x \qquad p = p_m - \Delta p$$
$$a_y = a_{y,m} - \Delta a_y \qquad q = q_m - \Delta q$$
$$a_z = a_{z,m} - \Delta a_z \qquad r = q_m - \Delta r$$

Identification of these error parameters ($\Delta a_x, \Delta a_y, \Delta a_z, \Delta p, \Delta q, \Delta r$) considerably improves the match for the three time sections under consideration. The small demonstration example stops at this point, a full flight path reconstruction would of course require more parameters, especially for also modeling errors in the output variables.

10.4. Low Order Equivalent System Approximation

This example for approximating an analytical high order system by a low order equivalent system (LOES) is taken from Mitchell/Hoh [20] and implemented as `demo_LOES.mat` and example 7 of the command line demo.

The high order transfer functions for pitch attitude θ and normal load factor n_z due to longitudinal stick force F_s are

$$\frac{\theta}{F_s} = \frac{(s+1.25)}{s(s+2)[0.7, 4.9][0.75, 63]}$$
$$\frac{n_z}{F_s} = \frac{1}{(s+2)[0.7, 4.9][0.75, 63]}$$

where $[\zeta, \omega_n]$ denotes a pair of complex zeros with damping ζ and natural frequency ω_n.

The corresponding low order equivalent system models are

$$\frac{\theta}{F_s} = \frac{K_\theta(s - 1/T_{\theta_e})}{s[\zeta_e, \omega_e]} e^{-\tau_e s}$$
$$\frac{n_z}{F_s} = \frac{-U_0 K_\gamma}{[\zeta_e, \omega_e]} e^{-\tau_e s}$$

As stated in the paper, the time constant of the lag between flight path and attitude responses ($1/T_{\theta_e}$) is not identified correctly when only the transfer function for θ/F_s is matched (the theoretical value is $1/T_{\theta_e} \doteq -Z_w = 1.25$). By simultaneously matching attitude and normal load factor, however, this time constant is determined correctly. Alternatively, the transfer function for θ/F_s can be matched with $1/T_{\theta_e}$ fixed at the expected value. The identified values listed in the paper are

match	$1/T_{\theta_e}$	ζ_e	ω_e	τ_e
θ/F_s	1.25 (fixed)	0.80	2.56	0.126
θ/F_s	4.08	0.52	3.80	0.098
θ/F_s and n_z/F_s	1.32	0.79	2.59	0.125

To be able to match two transfer functions simultaneously in FitlabGui, they must have the same denominator. This can be achieved by using $q/F_s = s\theta/F_s$ instead of θ/F_s. (The right part of figure 7.10 shows the resulting mismatch plots when both transfer functions are approximated together.)

10.5. EC 135 Dynamic Inflow

`demo_dynInflow_poly.mat` (corresponding to example 8 of the command line demo) compares two models for the vertical motion of the EC 135 helicopter in hover. Usually, the model for the vertical velocity w due to collective input δ_0 is

$$\frac{w}{\delta_0} = \frac{Z_{\delta_0}}{s - Z_w} e^{-\tau_{\delta_0} s}$$

As the coherence for a_z/δ_0 is often better than that for w/δ_0, the approximation $a_z = sw$ is usually used and the transfer function for vertical acceleration is approximated. It can be seen that for the EC 135 hover data, this simple model is not able to capture the rising amplitude with increasing frequency which is caused by the unmodeled dynamic inflow.

In [21] it is suggested to augment the model by a first-order lead-lag filter which leads to a model of

$$\frac{a_z}{\delta_0} = \frac{sZ_{\delta_0}(s+A)}{(s-Z_w)(s+B)} e^{-\tau_{\delta_0} s}.$$

In the paper the values for A and B are given as 4.8 and 12.9 respectively. Applying this augmented model to the EC 135 frequency response leads to a very good match (see the left part of figure 7.10) which shows that the dynamic inflow can be described by this type of model.

In `demo_dynInflow_fResp.mat`, the augmented model is formulated as a state space system

$$\begin{pmatrix} \dot{\tilde{w}} \\ \dot{w} \end{pmatrix} = \begin{bmatrix} Z_w & 0 \\ Z_w + A & B \end{bmatrix} \begin{pmatrix} \tilde{w} \\ w \end{pmatrix} + \begin{bmatrix} Z_{\delta_0} \\ Z_{\delta_0} \end{bmatrix} \delta_0. \tag{10.1}$$

The model is identified using the 'linear frequency response' model type and by matching the frequency response for w/δ_0. (There is no corresponding example in the command line demo.)

11. Command Line Interface for FITLAB

In addition to using FitlabGui for starting an identification, the unterlying parameter estimation routine FITLAB can also be run directly from the command line.

The calling sequence for FITLAB is

```
[par_id,bias_id,data_id,fdata_id,cost,lti,fitErr] = ...
    fitlab(model,par_0,bias_0,output,input,state,options,data,fdata)
```

The input and output arguments are as follows:

`model`	name of the model m-file (string) or model structure (for polynomial models)
`par_0`	parameter structure with the initial values
`bias_0`	bias parameter structure with the initial values (ignored for polynomial models)
`output`	output structure
`input`	input structure
`state`	state structure
`options`	options structure
`data`	time domain data structure with the measured data (required for time domain SysID, optional for frequency domain SysID, ignored for polynomial models)
`fdata`	frequency domain data structure with the measured data (required for frequency domain SysID and polynomial models)
`par_id`	parameter structure with the identified values
`bias_id`	bias parameter structure with the identified values
`data_id`	time domain data structure with measured and calculated data (if `data` was not empty)
`fdata_id`	freq. domain data structure with measured and calculated data (if `fdata` was not empty)
`cost`	structure with cost function information corresponding to the identification results
`lti`	identified model(s) as LTIs (SS-LTI for linear models, cell array of TF- or ZPK-LTIs for transfer functions, empty for nonlinear models)
`fitErr`	if `NaN`s occurred during simulation with the initial parameters, the corresponding time section indices and the time indices of the first occurrence of `NaN` are saved in `fitErr.i_TS_NaN` and `fitErr.i_first_NaN`

FITLAB uses MATLAB structures for most of its arguments. Names of parameters and input, output, and state variables are saved as cell arrays of strings. The different structures with their fields are described in the following sections. More information can also be gained from running the command line demo of FITLAB by typing `fitlab_demo` at the command line.

11.1. Model Formulation

Depending on whether the model whose parameters are to be estimated is nonlinear, linear, or a polynomial transfer function model, the user provided model must have a different structure. The default option in FITLAB is the identification of nonlinear models. If a model of another type is to be identified, this has to be specified in the `options` structure (see section 11.5). The definition of the user provided model files

for nonlinear and linear models is found in section 5.1.

For polynomial transfer function (TF) models, the model argument is a structure defining the model and not an m-file. Furthermore, the parameters (see section 11.2) have to be given in a specific order. No bias parameters are used for polynomial models.

For a polynomial model of the numerator/denominator type (see equation (B.29)), the model structure must have the following fields:

`model.degNum`	degree of the numerator(s) (integer vector)
`model.degDen`	degree of the denominator (integer)
`model.denLowOne`	=1 if the normalization is lowest denominator coeff. =1 =0 if the normalization is highest denominator coeff. =1
`model.nDel`	number of time delays (integer) must be either 0 (no delay), or 1 (common time delay for all TFs), or equal to the number of numerators

The corresponding parameters (see section 11.2) have to be given in the following order:

1. coefficients of the numerator(s) in descending order
2. coefficients of the denominator in descending order
3. time delay(s) (optional)

For a polynomial model of the pole/zero type (see equation (B.30)), the model structure must have the following fields:

`model.nSimpleNum`	number of simple roots in the numerator(s) (integer vector)
`model.nSimpleDen`	number of simple roots in the denominator (integer)
`model.nQuadNum`	number of quadratic roots in the numerator(s) (integer vector)
`model.nQuadDen`	number of quadratic roots in the denominator (integer)
`model.nDel`	number of time delays (integer) must be either 0 (no delay), or 1 (common time delay for all TFs), or equal to the number of numerators

The corresponding parameters (see section 11.2) have to be given in the following order:

1. for each numerator:
   a) the gain
   b) the simple roots

   c) damping and frequency pairs for all quadratic roots
2. the simple roots of the denominator
3. damping and frequency pairs for all denominator quadratic roots
4. time delay(s) (optional)

Examples for both types of polynomial models can be found in the examples 7 and 8 of the command line demo. The corresponding subroutines are `fitlab_demo_LOES` and `fitlab_demo_dynInflow_poly`.

The specification of polynomials models is more error prone than the other model types due to the necessary correspondence of model structure and parameter order. Therefore, it is suggested to identify polynomial models only via FitlabGui and not through the command line version of FITLAB.

11.2. Parameters and Bias Parameters

The model coefficients are divided into parameters and bias parameters (see section B.1.2). Parameters are valid for all time intervals, that are to be evaluated, whereas bias parameters have the same function but different numerical values for each time interval. Examples for bias parameters are initial conditions or trim settings.

For each parameter, a name and an initial value has to be given. The third component of the parameter structure determines which of the parameters are to be estimated and which are held fixed at their initial values. The fourth and fifth component represent parameter bounds (bounds can be set to $\pm$`Inf` for unbounded parameters).

`par.name`	names of the parameters (cell array of strings)
`par.value`	values of the parameters (vector)
`par.on`	=1 for parameters to be identified
	=0 for parameters that are to be held fixed
`par.min`	parameter lower bounds (vector, optional)
`par.max`	parameter upper bounds (vector, optional)

The structure for the bias parameters is almost identical to the one for the parameters. The only difference is that initial values for all time sections have to be specified if several time intervals are evaluated together. Bounds for bias parameters as well as the on/off information are valid for all time intervals.

`bias.name`	names of the bias parameters (cell array of strings)
`bias.value`	values of the bias parameters for all time sections (matrix, each time section in one column)
`bias.on`	=1 for bias parameters to be identified =0 for bias parameters that are to be held fixed
`bias.min`	bias parameter lower bounds (vector, optional)
`bias.max`	bias parameter upper bounds (vector, optional)

After an identification run, the returned structures `par_id` and `bias_id` have fields `par_id.stddev` and `bias_id.stddev` added, that contain the standard deviations (see equation B.41) of the corresponding parameters resp. bias parameters.

11.3. Output / Input / State Variables

The output structure must contain the names of the output variables and the information about which of the output variables are weighted in the cost function.

`output.name`	names of the output variables (cell array of strings)
`output.on`	=1 for output variables that are weighted =0 for output variables that are not weighted

For polynomial transfer function models, the output structure contains the names of the transfer functions to be approximated.

The input and state structures consist only of the names of the input respectively state variables (`input.name` resp. `state.name`). The two structures are required only for the model type `f_resp` (see section 11.5). They are ignored for nonlinear and polynomial models. For all model types that create SS-LTI models (`lin_td`, `lin_fd` and `f_resp`, the names are used to label the inputs and outputs of the identified LTI model (return argument `lti` of FITLAB).

11.4. Measured Data

11.4.1. Output Error Method

The measured time domain and/or frequency domain data has to be provided as a vector of structures with each element of the vector corresponding to one time

interval. For time domain data the structure is built up as

`data(iz).t`	vector with sampling times
`data(iz).u`	measured input variables (each variable in one column)
`data(iz).z`	measured output variables (each variable in one column)
`data(iz).x`	measured state variables (each variable in one column, required only for startup calculation of linear systems)

After completion of the parameter estimation, the output data structure `data_id(iz)` consists of the fields `data_id(iz).dt, .t, .u, .z`, that are identical to the corresponding fields of the input data structure `data(iz)`, and of `data_id(iz).y` which contains the calculated output variables (model output, based on the identified values of the parameters).

The frequency domain data structure is very similar:

`fdata(iz).w`	frequency vector
`fdata(iz).u`	measured input variables transformed into the freq. domain
`fdata(iz).z`	measured output variables transformed into the freq. domain
`fdata(iz).x`	measured state variables transformed into the freq. domain (required only for startup calculation of linear systems)

As for the time domain case, the output data structure `fdata_id(iz)` contains a field `fdata_id(iz).y` with the calculated output variables in addition to the frequency information and the measured input and output variables as taken from `fdata(iz)`.

For creating a frequency domain data structure from time domain data the following utility program is provided:

```
[fdata] = fitlab_td2fd(data,fmin,fmax,units,linlog,df)
```

with

`data`	time domain data structure
`fmin`	minimum frequency to be returned
`fmax`	maximum frequency to be returned
`units`	string specifying the units of fmin and fmax ('Hz' or 'rad/s')
`linlog`	string identifing desired frequency spacing ('lin' or 'log')
`df`	value defining frequency spacing for 'lin' spacing and values per decade for 'log' spacing
`fdata`	frequency domain data structure

The routine uses a chirp-z Transform (CZT) to transform the measured time domain data to the frequency domain. If logarithmic spacing is desired, the frequency domain

data is first smoothed by a moving average algorithm and then interpolated to the desired frequency axis.

11.4.2. Frequency Response Method

For the frequency response method, the data has to be in a different format. Each frequency response is stored in a separate element of `fdata` to enable different frequency axes for the different frequency responses. Amplitude, phase, and optionally coherence are stored in the other fields of the structure.

`fdata(iz).w`	frequency in rad/s (vector)
`fdata(iz).zMag`	amplitude in dB (vector)
`fdata(iz).zPhase`	phase in deg (vector)
`fdata(iz).coh`	coherence (vector, optional)
`fdata(iz).weight`	weighting (vector, 0=off, 1=on, for `system = f_resp` only)

After completion of an estimation run, the output data structure `fdata_id(iz)` consists of the fields `fdata_id(iz).w`, `.zMag`, `.zPhase`, that are identical to the corresponding fields of the input data structure `fdata(iz)`, and of `fdata_id(iz).yMag` and `fdata_id(iz).yPhase`, which contain the magnitude and phase of the identified model for the given frequency vector. Additionally, the weighting function w_γ (see equation B.25) is contained in `fdata_id(iz).wgt`.

When the identified model is linear (i.e. `system = f_resp`), additional fields are appended to the structure that contain the indices of the corresponding input and output in the linear model (`fdata_id(iz).iy` and `fdata_id(iz).iu`) and the model name of the frequency response (`fdata_id(iz).name`).

11.5. Options

Several parameter settings used by FITLAB are contained in an options structure. This options structure is similar to the one used by the MATLAB optimization routines. Default settings in this structure can be changed by the routine `fitlab_set` (type `help fitlab_set` for more information).

The different options and their default settings are listed in the following table. All defaults are used when `options=[]` is used in the call of FITLAB.

Option	Default	Meaning
System	nonlin	System class: nonlin nonlinear lin_td linear time domain lin_fd linear frequency domain poly polynomial transfer function model f_resp linear model with frequency response method
Optimizer	1	Optimization method: 1 Gauss-Newton 2 Subplex 3 fmincon (constrained minimization) 4 lsqnonlin (nonlinear least-squares) Options 3 and 4 work only if the MATLAB optimization toolbox is installed.
MaxIter	0	Maximum number of iterations allowed
MaxIterStart	□	Maximum number of startup iterations allowed. (for System=lin_td or lin_fd only)
MaxIterTime	□	Maximum number of time domain iterations allowed for identification of bias parameters. (for System=lin_fd only)
TolCost	1e-3	Termination tolerance for the relative change of the cost function value
TolPar	1e-4	Termination tolerance for the relative change of the parameters
EpsSvd	1.e-12	Values below EpsSvd are regarded as zero for the singular value decomposition (Gauss-Newton only)
DiffChange	1.e-4	Relative/minimum change in parameters for finite difference gradients (eq. B.38, Gauss-Newton only)
MaxCostEval	10	Maximum number of cost function evaluations during line search cycle (Gauss-Newton only)
RelStepSize	1e-3	Relative startup step size for the optimizer's parameter variation (Subplex only)
AmplitudePhase	0.01745	Relative weighting between amplitude and phase error (system = poly or f_resp only)
CohWeight	off	Specifies, if coherence weighting is to be used (on/off, system = poly or f_resp only)

Option	Default	Meaning
`Display`	`iter`	Level of display: `none` no display `cost` display only cost function (for simulation) `final` display only the final results `short` display cost function and parameter values for each iteration, display bias parameter values only for the final results `iter` display cost function as well as parameter and bias parameter values for each iteration `extended` same as `iter` plus display of the cost for each time interval and the (root) mean square error for each output variable `mixed` same as `short` during iteration and same as `extended` for final display
`CorrLim`	0.9	In the display of the final results, only correlations above `CorrLim` are displayed
`LogFile`	□	Name of a logfile for saving the results. If a logfile is specified, all screen output of FITLAB is also written to that file.

Table 11.1.: **Components of the FITLAB options structure**

12. Summary

FitlabGui is a MATLAB tool for flight data analysis and parameter estimation. The software has developed over the years and this report describes version 2.7.

The report first describes how to install and call the software. Then, the different panels and their options are presented. Several utilities that come with FitlabGui and the demonstration examples for the parameter estimation are also described. A separate chapter explains how to use the underlying parameter estimation software FITLAB from the command line instead of starting it via FitlabGui.

In the appendix, the mathematical background for the three implemented frequency response generation methods is given. A second appendix explains the parameter estimation methods implemented in FitlabGui.

Bibliography

[1] Peter Hamel and Ravindra Jategaonkar. Evolution of Flight Vehicle System Identification. *Journal of Aircraft*, 33(1):9–28, Jan-Feb 1996.

[2] Peter Hamel and Ravindra Jategaonkar. The Role of System Identification for Flight Vehicle Applications - Revisited. In *RTO-MP-11*, Mar 1999. Paper 2.

[3] Ermin Plaetschke and D. B. Mackie. Maximum-Likelihood Schätzung von Parametern linearer Systeme aus Flugversuchsdaten - Ein FORTRAN-Programm. Mitt. 84-10, DFVLR, Jan 1984.

[4] Ravindra Jategaonkar and Ermin Plaetschke. Maximum-Likelihood Estimation of Parameters in Linear Systems with Process and Measurement Noise. FB 87-20, DFVLR, Jun 1987.

[5] Kuang-Hua Fu. Systemidentifizierung im Frequenzbereich - Methodik und Anwendung. IB 111-88/48, DFVLR, Nov 1988.

[6] Kuang-Hua Fu. Ein Rechenprogramm für die Parameteridentifizierung nach der Methode der kleinsten Fehlerquadrate und der Maximum-Likelihood Methode. IB 111-90/01, DFVLR, Jan 1990.

[7] Ravindra Jategaonkar. User's Manual for NLMLKL, a General FORTRAN Program for Parameter Estimation in Time Domain. IB 111-95/03, DLR, Mar 1995.

[8] Ravindra Jategaonkar. ESTIMA - a Modular and Integrated Software Tool for Parameter Estimation and Simulation of Dynamic Systems. IB 111-2001/29, DLR, Jul 2001.

[9] NN. *MATLAB Version 7.5 Documentation*. The Mathworks.

[10] NN. *SIMULINK Version 7.0 Documentation*. The Mathworks.

[11] Susanne Weiß. PENSUM - Parameter Estimation of Nonlinear Systems Using Matlab - Version 1.0. IB 111-1999/32, DLR, Dec 1999.

[12] Susanne Seher-Weiß. HQ-Tools: An Add-On to FITLAB for Helicopter Handling Qualities Analysis. Technical Report DLR-FT-BS-2026-10, DLR, Jan 2026.

[13] National Space Science Data Center NASA/Goddard Space Flight Center, Greenbelt, Maryland 20771. *CDF User's Guide Version 3.0*, 2005.

[14] Helmut Reetz. MeDaLib - Programm zur Aufbereitung von Messdaten - Anwendungshandbuch. IB 111-2001/01, DLR, Jan 2001.

[15] Achim Jäkel. RCDF - Institutsstandard für Flugversuchsdateien. IB FT-BS-2017-33, DLR, Mar 2017.

[16] NN. *Control System Toolbox Version 8.0.1 Documentation*. The Mathworks.

[17] Mark B. Tischler and Robert K. Remple. *Aircraft and Rotorcraft System Identification - Engineering Methods with Flight Test Examples*. AIAA Education Series. AIAA, 2nd edition, 2012.

[18] Dept. of Defense Handbook. Flying Qualities of Piloted Aircraft. USAF MIL-STD-1797B, Dec 1997. USAF MIL-STD-1797B.

[19] H. Reetz. Datenaufbereitungsprogramme DA1 und DA2 - Anwendungshandbuch. IB 111-97/42, DLR, Nov 1997.

[20] David G. Mitchell and Roger H. Hoh. Low-Order Approaches to High-Order Systems: Problems and Promises. *Journal of Guidance*, 5(5):482–489, 1981.

[21] Jeffery A. Schroeder, Mark B. Tischler, Douglas C. Watson, and Michelle M. Eshow. Identification and Simulation Evaluation of a Combat Helicopter in Hover. *Journal of Guidance, Control, and Dynamics*, 18(1):31–38, Jan-Feb 1995.

[22] J.S. Bendat and A.G. Piersol. *Engineering Applications of Correlation and Spectral Analysis*. John Wiley & Sons, New York, 1980.

[23] L.R. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing*. Prentice Hall, Englewood Cliffs, New Jersey, 1975.

[24] W.H. Press, S.A. Teukolsky, W.A. Vetterling, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, New York, 1992.

[25] M.B. Tischler and M.G. Cauffmann. Frequency-Response Method for Rotorcraft System Identification: Flight Applications to BO 105 Coupled Rotor/Fuselage Dynamics. *Journal of the American Helicopter Society*, 37(3):3–17, 1992.

[26] J.K. Sridhar and G. Wulff. Application of Multiple-Input/Single-Output Analysis Procedures to Flight Test Data. *Journal of Guidance, Control, and Dynamics*, 14(3):645–651, May-June 1991.

[27] Carl J. Ockier. The Art of Frequency Response Calculation. IB 111-97/07, DLR, Feb 1997.

[28] J.S. Bendat and A.G. Piersol. *Random Data: Analysis and Measurement Procedures*. John Wiley & Sons, New York, 1971.

[29] Susanne Seher-Weiß. Vergleich zweier Verfahren zur Frequenzgangerzeugung. IB 111-2012/54, DLR, Jun 2012.

[30] R. Pintelon and J. Schoukens. *System Identification: A Frequency Domain Approach*. John Wiley & Sons, Inc., 2012. 2nd Edition.

[31] Benjamin Fragnière and Johannes Wartmann. Local Polynomial Method Frequency-Response Calculation for Rotorcraft Applications. In *AHS 71st Annual Forum*, Virginia Beach, VA, May 2015. May 5-7.

[32] P. Thummala and J. Schoukens. Estimation of the FRF Through the Improved Local Bandwidth Selection in the Local Polynomial Method. *IEEE Transactions on Instrumentation and Measurement*, 61(10):2833–2843, Oct 2012.

[33] J. Schoukens, G. Vandersteen, Y. Rolain, and R. Pintelon. Frequency Response Function Measurements Using Concatenated Subrecords With Arbitrary Length. *IEEE Transactions on Instrumentation and Measurement*, 61(10):2682–2688, Oct 2012.

[34] Martin Marchand and Kuang-Hua Fu. Frequency Domain Parameter Estimation of Aeronautical Systems with and without Time Delay. In *Proceedings of the 7th IFAC Symposium on Identification and System Parameter Estimation*, pages 669–674, York, UK, Jul 1985. July 3-7.

[35] Holger Duda. POLYKO: Ein Verfahren zur Bestimmung der Koeffizienten und Totzeiten von Übertragungsfunktionen aus Frequenzgangwerten unter MATLAB. IB 111-94/03, DLR, Jan 1994.

[36] NN. *Optimization Toolbox Version 7.5 Documentation*. The Mathworks.

[37] Lothar Thiel. Numerische Methoden der Systemidentifizierung - Givens-Transformationen. IB 111-92/37, DLR, Sep 1992.

[38] Ravindra Jategaonkar. Bounded-Variable Gauss-Newton Algorithm for Aircraft Parameter Estimation. *Journal of Aircraft*, 37(10):742–744, 2000.

[39] J.A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7:308–313, 1965.

[40] T. Rowan. *Functional Stability Analysis of Numerical Algorithms*. PhD Thesis, University of Texas at Austin, 1990.

[41] Ralf Beck. Parameteridentifizierung bei Systemen mit Unstetigkeiten. IB 111-2004/06, DLR, Jan 2004.

[42] M. B. Subrahmanyam. An Extension of the Simplex Method to Constrained Optimization. *Journal of Optimization Theory and Applications*, 62(2):311–319, 1989.

A. Frequency Response Generation

A.1. Frequency Response and Coherence

Let x_n and y_n, $n = 0, \cdots, N-1$ be the sampled input and output signals of the system under consideration. The length of the time interval is T and and the N data points are sampled with a sampling interval of Δt seconds

$$T = \Delta t(N-1). \tag{A.1}$$

The corresponding finite Fourier transforms $X(f_k)$ and $Y(f_k)$

$$X(f_k) = \Delta t \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N}, \quad Y(f_k) = \Delta t \sum_{n=0}^{N-1} y_n e^{-i2\pi kn/N} \tag{A.2}$$

are determined at discrete frequencies f_k

$$f_k = \frac{k}{T} = kf_1, \quad k = 0, \cdots, N-1. \tag{A.3}$$

f_1 is the fundamental frequency or frequency resolution and is the inverse of the length of the time interval. The Nyquist frequency $f_c = 0.5Nf_1 = (2\Delta t)^{-1}$ is the highest frequency that is available in the Fourier transform.

Once the Fourier transforms have been determined, the auto-spectra (or autospectral density functions) of the input and output signals are calculated as

$$G_{xx}(f_k) = \frac{2}{T} X^*(f_k)X(f_k) = \frac{2}{T}|X(f_k)|^2 \tag{A.4}$$

$$G_{yy}(f_k) = \frac{2}{T} Y^*(f_k)Y(f_k) = \frac{2}{T}|Y(f_k)|^2 \tag{A.5}$$

where $*$ denotes the conjugate complex value. The cross-spectrum (or cross-spectral density function) of the input/output signal is defined as

$$G_{xy}(f_k) = \frac{2}{T} X^*(f_k)Y(f_k) \tag{A.6}$$

and the frequency response H is now determined from

$$H(f_k) = \frac{G_{xy}(f_k)}{G_{xx}(f_k)}. \tag{A.7}$$

The coherence function γ^2_{xy} is an indicator of the correlation of the input and the output signals and is determined from

$$\gamma^2_{xy}(f_k) = \frac{|G_{xy}(f_k)|^2}{G_{xx}(f_k)G_{yy}(f_k)} \qquad 0 \leq \gamma^2_{xy}(f_k) \leq 1. \tag{A.8}$$

The coherence function is a quantification of the cross-spectrum inequality

$$|G_{xy}(f_k)^2| \leq G_{xx}(f_k)G_{yy}(f_k). \tag{A.9}$$

The proof of this inequality can be found on pages 54ff and 97ff of [22].

The standard technique for computing the discrete Fourier transforms is the Fast Fourier Transform (FFT). For the frequency response generation as implemented in FitlabGui, the Chirp-Z transform is used instead of the FFT. A description of the Chirp-Z transform (CZT) can be found on pp. 393-399 of [23]. Compared to the FFT, the CZT has the advantage that it works better for data point numbers that are multiples of large prime numbers and that the frequency points, for which the Fourier transform shall be returned, can be specified.

A.2. Segmenting and Windowing

One method to avoid frequency leakage is tapering or windowing of the data. (A good discussion of the phenomenon of frequency leakage can be found in [22] or [24].) In FitlabGui a Hanning window is used to process the input and output signals. This windowing function is defined by

$$w_k = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi k}{N}\right)\right] \tag{A.10}$$

where N is the number of data points in the record. It is important that both the input and the output signal are processed with the same window. In the case of the Hanning window, a correction factor of $(8/3)^{1/2}$ has then to be applied when determining the auto- and cross-spectra from (A.4) and (A.6).

Segmenting, i.e. subdividing the data record into segments of equal size, is a legitimate technique to reduce the random error of the data in the frequency response. Generally,

the segments are chosen such that they overlap each other by 50%. If the data record of length N is subdivided into n_d segments that overlap by 50%, the segments each have length M with

$$M = \frac{2N}{n_d + 1} \tag{A.11}$$

Because of the reduced length of the data segments, the fundamental frequency is increased to

$$f_1 = \frac{1}{Mh} = \frac{n_d + 1}{2N\Delta t} \tag{A.12}$$

Thus, by segmenting one has less error at the price of a coarser frequency resolution. (The Nyquist frequency, which depends solely on the sampling rate, is unchanged.) The random error ϵ_r is a function of the number of data segments and of the coherence of the input/output relationship and is calculated by

$$\epsilon_r\left(|H(f_k)|\right) = \frac{C_\epsilon \left[1 - \gamma_{xy}^2(f_k)\right]^{1/2}}{|\gamma_{xy}|\sqrt{2n_d}} \tag{A.13}$$

Here, C_ϵ is a constant that accounts for the degree of overlap between the windows ($C_\epsilon = \sqrt{0.55}$ for 50% and $C_\epsilon = \sqrt{0.50}$ for 80% overlap.).

The derivation of the above equation can be found in [22] for the non-overlapping version and in [25] for the function with 50% overlap.

The random error in (A.13)is normalized. Thus the standard deviation of the magnitude will be

$$\sigma(|H(f_k)|) = \epsilon_r(|H(f_k)|)/|H(f_k)| \tag{A.14}$$

and the standard deviation of the phase is given by

$$\sigma(\angle(H(f_k))) = \sin^{-1}(\epsilon_r(|H(f_k)|)). \tag{A.15}$$

A.3. Multi-Input Single-Output Conditioning

When multiple inputs that are partially correlated excite one output, it is necessary to use Multi Input / Single Output (MISO) conditioning to arrive at meaningful frequency responses and coherence functions. A system with two inputs $x_1(t)$ and $x_2(t)$ and one output $y(t)$ has the input/output relationship

$$Y = H_{1y}X_1 + H_{2y}X_2 + V \tag{A.16}$$

where H_{1y} and H_{2y} are the transfer functions between the first and second inputs and the output and V is the noise. If this equation is multiplied by $(2/T)X_1^*$ respectively

$(2/T)X_2^*$, taking into account equation (A.1) leads to

$$G_{1y} = H_{1y}G_{11} + H_{2y}G_{12} \tag{A.17}$$
$$G_{2y} = H_{1y}G_{21} + H_{2y}G_{22} \tag{A.18}$$

From these two equations, the conditioned frequency responses H_{1y} and H_{2y} are computed via

$$H_{1y} = \frac{G_{22}G_{1y} - G_{12}G_{2y}}{G_{11}G_{22} - |G_{12}|^2} = \frac{G_{1y}\left[1 - \frac{G_{12}G_{2y}}{G_{22}G_{1y}}\right]}{G_{11}[1-\gamma_{12}^2]} = \frac{G_{1y,2}}{G_{11,2}} \tag{A.19}$$

$$H_{2y} = \frac{G_{11}G_{2y} - G_{21}G_{1y}}{G_{11}G_{22} - |G_{21}|^2} = \frac{G_{2y}\left[1 - \frac{G_{21}G_{1y}}{G_{11}G_{2y}}\right]}{G_{22}[1-\gamma_{12}^2]} = \frac{G_{2y,1}}{G_{22,1}} \tag{A.20}$$

Here, $G_{1y,2}$, $G_{11,2}$, $G_{2y,1}$, and $G_{22,1}$ are the conditioned auto- and cross-spectra and γ_{12}^2 is the coherence between the two inputs $x_1(t)$ and $x_2(t)$.

$$\gamma_{12}^2 = \frac{|G_{12}|^2}{G_{11}G_{22}} \tag{A.21}$$

If the two inputs are totally uncorrelated, i.e. if $\gamma_{12}^2 = 0$, then the conditioned frequency responses reduce to the unconditioned ones.

The ordinary coherence functions between the two inputs and the output are given by

$$\gamma_{1y}^2 = \frac{|G_{1y}|^2}{G_{11}G_{yy}} = \frac{|H_{1y}G_{11} + H_{2y}G_{12}|^2}{G_{11}G_{yy}} \tag{A.22}$$
$$\gamma_{2y}^2 = \frac{|G_{2y}|^2}{G_{22}G_{yy}} = \frac{|H_{2y}G_{22} + H_{1y}G_{21}|^2}{G_{22}G_{yy}} \tag{A.23}$$

These coherence functions represent the correlation between the input and output signals. Unfortunately, the ordinary coherence uses both paths to determine coherence (directly from x_1 over H_{1y} to y and from x_1 through its correlation with x_2 over H_{2y} to y). Therefore, for the analysis of frequency responses from partially correlated inputs, the ordinary coherence cannot be used. Instead the partial coherence has to be used, which is computed from the conditioned auto- and cross-spectra via

$$\gamma_{1y,2}^2 = \frac{|G_{1y,2}|^2}{G_{11,2}G_{yy,2}} = \frac{|G_{1y}G_{22} - G_{2y}G_{12}|^2}{G_{22}^2G_{11}G_{yy}(1-\gamma_{12}^2)(1-\gamma_{2y}^2)} \tag{A.24}$$
$$\gamma_{2y,1}^2 = \frac{|G_{2y,1}|^2}{G_{22,1}G_{yy,1}} = \frac{|G_{2y}G_{11} - G_{1y}G_{21}|^2}{G_{11}^2G_{22}G_{yy}(1-\gamma_{12}^2)(1-\gamma_{1y}^2)} \tag{A.25}$$

If MISO conditioning is used, all frequency responses, spectra, and coherences returned by FitlabGui are conditioned ones.

An extension of the determination of the conditioned auto- and cross-spectra and partial coherences to cases with more than two inputs can be found in [22] or in [26] (the latter also covers some computational aspects). In FitlabGui, the method described in [26] is used to determine the conditioned frequency responses.

A.4. Composite Frequency Responses

Long segments are best for determining the frequency response at low frequencies whereas short segments are needed for the high frequencies. The logical conclusion is that the optimal frequency response is a composite that is made up of several frequency responses with different numbers of segments (few segments at the low frequencies and many segments at higher frequencies). In FitlabGui two different methods for deriving composite frequency responses from the results for different segments are implemented.

The first method was developed by Ockier and is described in [27]. It is based on a weighting function that states that the frequency response derived by dividing the data record into n_d windows is most trustworthy in the following frequency range

$$f_1\left[\left(\frac{n_d}{10}\right)^2+1\right] < f_k < f_1\left[\left(\frac{n_d}{10}\right)^2+1+14\frac{n_d}{10}\right] \tag{A.26}$$

If a higher resolution of the frequency response is desired, this equation for the frequency range is changed to

$$f_1\left[\left(\frac{n_d}{10}\right)^4+1\right] < f_k < f_1\left[\left(\frac{n_d}{10}\right)^4+1+14\frac{n_d}{10}\right] \tag{A.27}$$

These two equations were developed empirically from helicopter response data obtained with frequency sweeps.

For the Ockier method the computation of the composite frequency responses is performed with the following steps:

1. For each number of windows n_d, the data is segmented using 50% overlap and the frequency responses including MISO conditioning are computed and stored.
2. For each frequency point, the standard deviations of the amplitude and the phase are determined from (A.14) resp. (A.15).
3. For each segmentation, (A.26) or (A.27) is used to determine, which data points are trustworthy and which can be discarded.
4. For each frequency point that is to be considered, the optimal frequency response is computed by maximizing the likelihood of that data point. A Cauchy

or Lorentzian distribution of the data (see [28], chapter 15.7 'Robust Estimation') is assumed.

5. Any points that are not trustworthy are removed.

With the above defined weighting function, the validity range for each segmentation (number of windows) is relatively small so that in general many different segmentations are needed to cover the frequency range of interest. Furthermore, this method does not yield input, output, and cross spectra that might also be needed. Therefore a second method for frequency response generation was implemented in FitlabGui.

The second method was developed by Tischler and is described in chapter 10 of [17]. With this method, the conditioned spectra for several segmentations (usually up to 5) are used to derive composite spectra by minimizing a common cost function. The composite spectra are then used to calculate the composite frequency response.

The weighting of the results for each segmentation $i = 1, \ldots n_w$ is based on the corresponding random error (see (A.13)).

$$W_i = \left[\frac{(\epsilon_r)_i}{(\epsilon_r)_{min}}\right]^{-4} \tag{A.28}$$

Thus the results with the lowest random error get a weighting of 1 and the other segmentations are deweighted.

Starting values for the composite spectra are derived by simple averaging. e.g.

$$G_{xx} = \sum_{i=1}^{n_w} W_i^2 G_{xx_i} / \sum_{i=1}^{n_w} W_i^2 \tag{A.29}$$

for the input autospectrum. The corresponding coherence is calculated by

$$\gamma_{xy}^2(f) = \frac{|G_{xy}(f)|^2}{|G_{xx}(f)|\,|G_{yy}(f)|} \tag{A.30}$$

The final composite spectra (index c) are those that minimize the following weighted least-squares cost function L for each frequency f:

$$\begin{aligned} L(f) = \sum_{i=1}^{n_w} W_i \Bigg\{ &\left(\frac{G_{xx_c} - G_{xx_i}}{G_{xx}}\right)^2 + \left(\frac{G_{yy_c} - G_{yy_i}}{G_{yy}}\right)^2 + \left(\frac{\Re(G_{xy_c}) - \Re(G_{xy_i})}{\Re(G_{xy})}\right)^2 \\ &+ \left(\frac{\Im(G_{xy_c}) - \Im(G_{xy_i})}{\Im(G_{xy})}\right)^2 + 5\left(\frac{\gamma_{xy_c}^2 - \gamma_{xy_i}^2}{\gamma_{xx}^2}\right)^2 \Bigg\} \end{aligned} \tag{A.31}$$

Including the coherence term in the cost function ensures that the coherence function of the composite frequency response will track the coherence of the most reliable

windows over the entire frequency range. Due to the coherence term, the cost function depends nonlinearly on the desired composite spectral quantities and thus the cost function must be minimized iteratively.

For the Tischler method the computation of the composite frequency responses is performed with the following steps:

1. For each window length the spectra are computed using 80% overlap between the windows.
2. MISO conditioning is performed for all segmentations.
3. The weighting function from (A.28) is calulated for each window length.
4. Starting values for the composite results are determined from (A.29).
5. For each frequency point the cost function (A.31) is optimized to arrive at the composite spectra.
6. The composite frequency response and coherence are derived from the composite spectra.

Both methods for deriving composite frequency responses from the results for different segmentations are compared in [29].

A.5. Local Polynomial Method

Whereas segmenting and windowing has been a standard for frequency response calculation since the 1980s, the so-called Local Polynomial Method (LPM) was developed at the end of the 2000s [30]. LPM is presented as an alternative to the windowing methods, with better performance due to an improved reduction of the leakage error. The performance of the LPM when applied to rotorcraft data is assessed in [31].

In contrast to the windowing methods, the LPM does not eliminate the leakage term through the application of windows, but it considers the leakage as an unknown function that has to be determined. Hence, the LPM assumes that the discrete Fourier transforms (DFT) of the input $u(t)$ and output $y(t)$ measurements

$$\begin{aligned} U(k) &= \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} u(t) e^{-j2\pi kt/N} \\ Y(k) &= \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} y(t) e^{-j2\pi kt/N} \end{aligned} \tag{A.32}$$

are linked by the so-called extended transfer function model

$$Y(k) = G(\omega_k)U(k) + T(\omega_k) + V(k) \tag{A.33}$$

for each frequency $\omega_k = 2\pi k f_s/N$ ($k = 1, ..., N$). Here f_s is the sampling frequency, N the number of samples, $G(\omega_k)$ the frequency response (FR) of the system, $T(\omega_k)$ the leakage term and $V(k)$ the DFT of the disturbing noise, which is assumed to be a filtered white noise, uncorrelated over the DFT lines k and circular complex distributed.

The estimation of the frequency response with the LPM is based on the assumption that the FR $G(\omega)$ and the leakage $T(\omega)$ are smooth functions of frequency. Therefore, they can be approximated by complex polynomials within a narrow frequency band. The polynomial approximations at frequencies ω_{k+r} ($r = ..., -2, -1, 0, 1, 2, ...$) of $G(\omega)$ and $T(\omega)$ of the order R and centered around frequency ω_k are given by

$$\begin{aligned} G(\omega_{k+r}) &= G(\omega_k) + \sum_{s=1}^{R} g_s(k) r^s, \\ T(\omega_{k+r}) &= T(\omega_k) + \sum_{s=1}^{R} t_s(k) r^s, \end{aligned} \tag{A.34}$$

where g_s and t_s are the Taylor coefficients of G and T respectively. R, the order of the polynomials, is a parameter of the method that has to be chosen by the user. In the literature, usually $R = 2$ is used and thus this value is also used in FitlabGui.

Considering (A.33) at frequency $k + r$ and using (A.34) leads to

$$\begin{aligned} Y(k+r) &= \left(G(\omega_k) + \sum_{s=1}^{R} g_s(k) r^s \right) U(k+r) + \left(T(\omega_k) + \sum_{s=1}^{R} t_s(k) r^s \right) + V(k+r) \\ &= \Theta K(k+r) + V(k+r) \end{aligned} \tag{A.35}$$

where Θ is the matrix of the unknown complex parameters, namely the Taylor coefficients of G and T at frequency k,

$$\Theta = [G(\omega_k)\ g_1(k)\ g_2(k)\ \ldots\ g_r(k)\ T(\omega_k)\ t_1(k)\ t_2(k)\ \ldots\ t_r(k)]\,. \tag{A.36}$$

$K(k + r)$ contains the input data

$$K(k+r) = \begin{bmatrix} K_1(r) \otimes U(k+r) \\ K_1(r) \end{bmatrix} \tag{A.37}$$

with

$$K_1(r) = \begin{bmatrix} 1 \\ r \\ \vdots \\ r^R \end{bmatrix}$$

and $\otimes$ denoting the Kronecker product of the matrices.

Collecting (A.35) for the $2n+1$ neighbouring frequencies k ($r = 0, \pm 1, \ldots, \pm n$) gives the following set of equations

$$Y_n = \Theta K_n + V_n \tag{A.38}$$

where Y_n, K_n and V_n are matrices of the form

$$Y_n = [Y(k-n)\, Y(k-n+1)\, \ldots Y(k)\, \ldots Y(k+n)]\,. \tag{A.39}$$

n is a parameter of the method that has to be chosen by the user under the constraint $2n+1 \geq (R+1)(n_u+1)$, with n_u being the number of inputs. The constraint ensures that (A.38) is an over-determined set of equations for the unknown estimate $\widehat{\Theta}$ that is to be solved in the least-squares sense

$$\min_{\Theta} \|Y_n - \Theta K_n\|\,. \tag{A.40}$$

The choice of the parameter n is a trade-off between an effective noise reduction (n big) and a low interpolation error (n small). In [32], a method is described that allows choosing the optimal n at every frequency k. This method is used in the LPM algorithm implemented in FitlabGui.

Thus $\widehat{\Theta}$ provides the best fitting complex polynomials for G and T around frequency k. The FR estimate at the k-th frequency is obtained from $\widehat{\Theta}$ using

$$\widehat{G}(\omega_k) = \widehat{\Theta} \begin{bmatrix} I_{n_u} \\ 0 \end{bmatrix} \tag{A.41}$$

with I_{n_u} being the identity matrix of dimension n_u. The steps above are repeated for every frequency k of the spectrum. Note that special attention has to be paid to the lower and higher edge frequencies of the spectrum. There, the $2n+1$ neighbouring frequencies cannot be centered around the frequency k, but have to be shifted to the right or to the left respectively.

LPM is available for both SISO and MIMO systems and is described in details in [30]. It has been shown in [33] that using concatenated data leads to a reduced bias and variance error of the FR estimate. The MIMO algorithm of the LPM can be used unchanged for concatenated data.

A.6. Uncertainty Bounds

For frequency responses with a corresponding coherence, i.e. those that have been generated using either the Tischler or the Ockier method, the uncertainty bounds

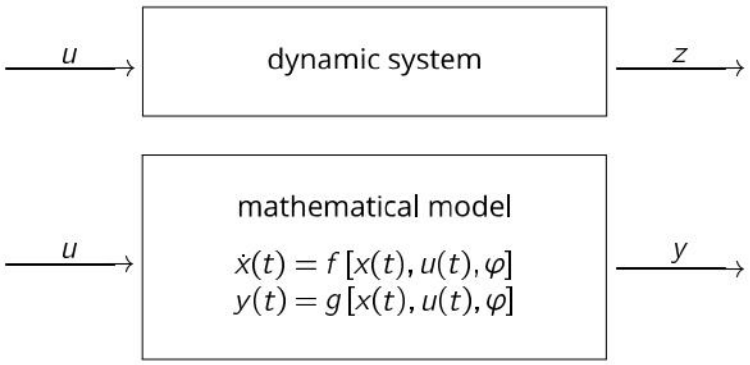
which are the 95% confidence bounds on magnitude and phase can be calculated. They are defined as

$$\begin{aligned} |H|(1-2\epsilon_r) \leq |H| \leq |H|(1+2\epsilon_r) \\ (\phi_{rad}-2\epsilon_r) \leq |\phi_{rad}| \leq (\phi_{rad}+2\epsilon_r) \end{aligned} \tag{A.42}$$

where $|H|$ is the magnitude in linear units and ϕ_{rad} the phase in radians and ϵ_r is the random error. If the frequency response has been generated with the Tischler method, the random error is directly available. For frequency responses generated with the Ockier method, an approximate random error is calculated from the coherence via eq. A.13 assuming a number of non-overlapping windows of $n_d = 15$ and setting $C_\epsilon = \sqrt{(}0.55)$ (80% overlap).

B. Parameter Estimation

It is assumed that the response z of a dynamic system to an input u has been measured. The goal is to develop a mathematical model that describes the system behavior. It is assumed that the model structure (system state function f and observation function g) is known correctly and only the model parameters φ have to be adjusted so that the simulated model output y matches the measured output z for the same input history u.



(B.1)

B.1. Maximum Likelihood Output Error Method

The dynamical system whose parameters are to be estimated is assumed to be described by the following mathematical model:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), \varphi); \qquad x(t_0) = x_0 \\ y(t) &= g(x(t), u(t), \varphi) \end{aligned} \tag{B.2}$$

Here, x and u denote the state and control input vectors. The model structure (f, g) is given and the coefficients φ and the initial conditions x_0 form the vector θ of the unknown parameters.

$$\theta = [\varphi; x_0] \tag{B.3}$$

The state variables cannot be measured directly. Instead, other output or observation variables y can be measured that depend on the unknown parameters and the state and control variables. Measurements z of these output variables that are corrupted with noise v exist for N discrete sampling times t_k.

$$z(t_k) = y(t_k) + v(t_k); \qquad k = 0, \ldots, N-1 \tag{B.4}$$

It is assumed that the state and observation equations correctly describe the dynamic system, i.e. that no modeling errors are present. The measurement errors are usually assumed to be characterized by stationary zero-mean Gaussian white noise with covariance matrix R.

$$E\{v(t_k)\} = 0, \qquad E\{v(t_i)v^T(t_j)\} = \delta_{ij}R \tag{B.5}$$

Here, $E\{\ldots\}$ denotes the expected value and δ_{ij} the Kronecker delta. This means that the measurement errors at different sample times are uncorrelated (i.e. white) and that the covariance matrix is time invariant.

With these assumptions for the error distribution, the probability of making the measurements z can be determined from the law of joint probability as

$$p(z|\theta) = [(2\pi)^m \det(R)]^{N/2} \exp\left\{-\frac{1}{2}\sum_{k=0}^{N-1}[z(k) - y(k)]^T R^{-1}[z(k) - y(k)]\right\} \tag{B.6}$$

Here, $p(z|\theta)$ denotes the probability of the measurements z for given parameters θ; m denotes the number of observation variables (dimension of z resp. y). In this equation, like in the following ones, k was used as a shorthand notion for t_k.

This function p is called the likelihood function. It is the probability distribution not of the unknown parameters but of the measurements. Maximum Likelihood estimation means to determine the parameter vector $\hat{\theta}$ that maximizes the function $p(z|\theta)$. This parameter vector $\hat{\theta}$ is called the most plausible because it gives the measurements that were made the maximum probability.

An equivalent task to maximizing $p(z|\theta)$ is the minimization of the negative logarithm L of the likelihood function:

$$L(z|\theta) = \frac{1}{2}\sum_{k=0}^{N-1}[z(k) - y(k)]^T R^{-1}[z(k) - y(k)] + \frac{N}{2}\log(\det(R)) \tag{B.7}$$

This cost function L consists of two terms that both contain the measurement error covariance matrix R. If R is known, the second term is constant and the variable part reduces to

$$\sum_{k=0}^{N-1}[z(k) - y(k)]^T R^{-1}[z(k) - y(k)] \tag{B.8}$$

Thus, the cost function in this case is a quadratic criterion. The sum of squares of the differences between model outputs and measurements, weighted by the measurement error covariances, are to be minimized.

If the measurement error covariance matrix R is unknown, as is usually the case, it has to be estimated like the parameters. Minimizing the cost function with respect to the elements in R^{-1} leads to the following Maximum Likelihood estimation for R (see [3]):

$$R = \sum_{k=0}^{N-1} [z(k) - y(k)] [z(k) - y(k)]^T \tag{B.9}$$

This means that the output error covariance matrix is the most plausible estimate for R.

With this estimate for R the first term of L reduces to a constant and the variable part reduces to

$$\det(R) \tag{B.10}$$

In FITLAB, as in the other identification programs used at the Institute, R is implemented as a diagonal matrix. This means that the measurement errors are assumed to be uncorrelated. As shown in [3], this also leads to faster convergence of the identification.

The Maximum Likelihood cost function in the frequency domain is derived analogously as

$$L(z|\theta) = \frac{1}{2} \sum_{k=0}^{N-1} [z(\omega_k) - y(\omega_k)]^* S^{-1} [z(\omega_k) - y(\omega_k)] + \frac{N}{2} \log(\det(S)) \tag{B.11}$$

This cost function is very similar to the one in the time domain (see equation (B.7)) with only the output error covariance matrix R replaced by the spectral density matrix S of the measurement noise.

B.1.1. Cost Function

In the case of an unknown output error covariance matrix, the Maximum Likelihood (ML) cost function for the output error method reduces to the determinant of the covariance matrix. If this matrix is assumed to be diagonal, i.e. neglecting the error covariances, the cost function reduces to the product of the output error variances

(mean square error) of all m output variables.

$$Cost = \prod_{i=1}^{m} \sigma^2(z_i - y_i) \tag{B.12}$$

with

$$\sigma^2(z_i - y_i) = \frac{1}{N} \sum_{k=0}^{N-1} (z_i(t_k) - y_i(t_k))^2 \tag{B.13}$$

Two important characteristics of this cost function should be pointed out here:

- If a 10% improvement in the match of one observation variable can be gained at the cost of a 10% deterioration in another, it will be done (as $0.9 * 1.1 = 0.99 < 1.00$).
- The minimum of zero in the cost function is reached when one output achieves an ideal match (irrespective of the match in the other outputs).

The second aspect rarely applies in practice as the presence of noise and measuring errors usually precludes an ideal match.

B.1.2. Model Extensions and Linear Systems

The initial conditions x_0 of the state variables are usually not known and thus have to be estimated together with the system parameters φ. Furthermore, the measured input and output variables are usually corrupted by unknown systematic errors Δu resp. Δz. These constant offsets shall also be included into the estimation as additional unknown parameters. Equations (B.1) then transform into

$$\begin{aligned} \dot{x}(t) &= f\,[x(t), u(t) - \Delta u, \varphi]\,; \qquad & x(t_0) &= x_0 \\ y(t) &= g\,[x(t), u(t) - \Delta u, \varphi] + \Delta z \end{aligned} \tag{B.14}$$

In most cases it will not be possible to estimate all components of x_0, Δu and Δz, as they are linearly dependent or at least highly correlated with each other.

Often it is necessary to use several time slices of measured data to estimate one set of parameters. In this case, the offsets Δu and Δz as well as the initial conditions x_0 can have different values for the different time intervals. This leads to a parameter vector of

$$\theta = \left[\varphi^T, x_{0,1}, \ldots, x_{0,NZ}, \Delta u_1, \ldots, \Delta u_{NZ}, \Delta z_1, \ldots, \Delta z_{NZ}\right]^T \tag{B.15}$$

in the general case of NZ time slices.

If the system state and observation functions are linear in the state and control variables, a different model is suitable. By substituting the state and control input variables with their deviations from the initial condition and by accounting for time delays, equations (B.14) are transformed into

$$\begin{array}{lll} \dot{x}(t) = A(\varphi) \cdot x(t) & +B(\varphi) \cdot u(t - \tau_{in}) & +b_x; \quad x(t_0) = 0 \\ y(t) = C(\varphi) \cdot x(t - \tau_{out}) & +D(\varphi) \cdot u(t - (\tau_{in} + \tau_{out})) & +b_y \end{array} \tag{B.16}$$

In this linear model, b_x and b_y are the lumped bias parameters of the state and observation equations respectively. They are linear combinations of the original initial conditions and zero-offsets. All components of b_x and b_y can be estimated if the system is observable.

τ_{in} is the time delay between the input u and the state vector x and τ_{out} denotes the time delay between x and the output y. These time delays correspond to the `InputDelay` and `OutputDelay` properties of MATLAB LTI models (see the documentation of the MATLAB Control System Toolbox [16]). An input delay can for example be an equivalent time delay whereas sensor delays correspond to output delays.

To estimate time delays in nonlinear Simulink models, a Transport Delay Block (see "Transport Delay" in the Simulink documentation [10]) can be used with the time delay of this block as an unknown parameter. It has to be noted, that in this case the value of the time delay must not be less than the sample time. Therefore, 0 cannot be used as the initial value and Δt should be used as lower bound.

B.1.3. Output Error Method in the Frequency Domain

A discretely sampled time dependent variable

$$x_n = x(n\Delta t); \qquad n = 0, \ldots, N-1 \tag{B.17}$$

can be transformed into a frequency dependent variable using the Fourier transform

$$x(\omega_k) = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{-j\omega_k n \Delta t}; \qquad k = 0, \ldots, N-1. \tag{B.18}$$

with

$$T = (N-1)\Delta t, \qquad \omega_k = k \cdot 2\pi/T \tag{B.19}$$

The variables $\dot{x}, x, u, y, z$ of a linear model are transformed into the frequency domain in this way. One has to note that the state variables do not always fulfil the condition

of periodicity. In this case, the Fourier transform of $\dot{x}$ is approximately given by [34]

$$\begin{aligned} &\dot{x}(\omega) = j\omega x(\omega) - b_p \bar{u}(\omega) \\ &b_p = \tfrac{1}{2T}\left[(x_{N-1} + x_N) - (x_{-1} + x_0)\right]; \qquad \bar{u} = e^{\frac{1}{2}j\omega\Delta t} \end{aligned} \tag{B.20}$$

which requires two additional data points x_{-1} and x_N not used in equation (B.18). As the data used for system identification usually starts and ends at a trim condition, the parameter b_p is ignored in FITLAB.

The model equations in the frequency domain are therefore

$$\begin{aligned} j\omega \cdot x(\omega) &= A(\varphi) \cdot x(\omega) + B(\varphi) \cdot u(\omega) \\ y(\omega) &= C(\varphi) \cdot x(\omega) + D(\varphi) \cdot u(\omega) \end{aligned} \tag{B.21}$$

The Maximum Likelihood function in the frequency domain is very similar to the one in the time domain (see equation B.11) and thus leads to a similar cost function

$$Cost = \prod_{i=1}^{m} \sigma^2(z_i - y_i) \tag{B.22}$$

with

$$\sigma^2(z_i - y_i) = \frac{1}{N} \sum_{k=0}^{N-1} \left(z_i(\omega_k) - y_i(\omega_k)\right)^* \left(z_i(\omega_k) - y_i(\omega_k)\right). \tag{B.23}$$

Parameter estimation in the frequency domain has the advantage that it is possible to significantly reduce the amount of data to be evaluated by restricting the evaluation to the frequency range of interest. The higher frequencies can often be omitted safely because they correspond to measurement noise and negligible higher order dynamics.

When the lowest frequency ($\omega = 0$) is omitted, the estimation of bias parameters is suppressed thus leading to much fewer unknown parameters, especially when several time intervals are evaluated together. One possibility to include the estimation of bias parameters is to first identify the system parameters φ using frequency domain identification and afterwards to only identify the bias parameters using time domain identification with φ fixed at the identified values.

As the model equations (B.21) are algebraic, no integration is necessary to calculate the output variables. This makes frequency domain models very suitable for unstable systems.

The substantial disadvantage of frequency domain identification is that it is restricted to linear systems. An approximate way of including nonlinear terms is to calculate nonlinear terms using measured signals and use them as additional inputs.

Frequency domain identification yields a different weighting compared to time domain identification. In the time domain, the lowest frequencies have the highest weighting because they correspond to more time samples and thus have more influence in the cost function. Obtaining the frequency domain data through a Fourier transform leads to linearly spaced frequency points (see (B.19)). Thus, the higher frequencies have comparably more data points and therefore more influence in the cost function. To provide an option to reduce the weighting of the higher frequencies, FitlabGui allows for using logarithmically spaced data in the ML frequency domain identification (see section 5.2 and the decription of the routine `fitlab_td2fd` in section 11.4).

B.1.4. Startup Algorithm for Linear Systems

When the state variables are measured, starting values for the unknown parameters can be obtained by running the identification in regression mode. In FITLAB, the model is reformulated as follows for the startup calculation:

$$\begin{aligned} \dot{x} &= \begin{bmatrix} B & A \end{bmatrix} \cdot \begin{bmatrix} u \\ x_m \end{bmatrix} + b_x \\ y &= Cx + \begin{bmatrix} D & 0 \end{bmatrix} \cdot \begin{bmatrix} u \\ x_m \end{bmatrix} + b_y \end{aligned} \tag{B.24}$$

This means that the measured states are used instead of the calculated ones in the state equations and that the observation equations remain unchanged. Thus the reformulated model is linear in the unknowns (parameters and calculated states) and can be solved by regression.

B.2. Frequency Response Method

In section B.1.3, identification based on matching the Fourier transforms of the measured time histories has been described. Another method, the frequency response method, is based on matching the frequency responses, i.e. the ratio of the output per unit of control input as a function of control input frequency.

The frequency response method can be used to approximate an analytically derived high-order model by a low-order model (LOES - Low Order Equivalent System). The method can also be used to approximate frequency responses that were derived from measured time histories of the input and output variables.

A good overview of system identification using the frequency response method can be found in [17].

B.2.1. Cost Function

An effective format for looking at a frequency response H is the Bode plot, which displays log-magnitude ($20\log_{10}|H|$, dB) and phase ($\angle H$, deg) vs log-frequency (ω, rad/s) on a semilog scale. Thus, the quadratic cost function L to be minimized for the frequency response method is

$$L = \frac{20}{N_\omega} \sum_{k=1}^{N\omega} w_\gamma(k) \left[(|H_m(k)|_{dB} - |H(k)|_{dB})^2 + w_{ap} (\angle H_c(k) - \angle H(k))^2 \right] \tag{B.25}$$

When several frequency responses are approximated together, the overall cost function is the average of the individual cost functions. In (B.25) N_ω is the number of frequency points in the frequency interval $[\omega_1, \omega_{N\omega}]$. H_m is the frequency response of the data to be approximated (either generated from measured time history data as described in the preceding subsection or analytically generated in the case of a given high-order system. H is the frequency response of the model. $|.|_{dB}$ is the amplitude in dB and $\angle(.)$ the phase in deg.

w_γ is an optional weighting function based on the coherence between the input and the output at each frequency. If coherence data is available and if coherence weighting is used, then

$$w_\gamma(k) = \left[1.58(1 - e^{\gamma_{xy}^2(k)}) \right]^2, \tag{B.26}$$

otherwise $w_\gamma \equiv 1$. This is the same weighting function as used e.g. in CIFER® [17].

w_{ap} is the relative weight between amplitude and phase errors. The normal convention is $w_{ap} = 0.01745$ and this value is the default used in FITLAB. The military standard on flying qualities [18] suggests a value of $w_{ap} = 0.02$.

Compared to the ML output error cost function from section B.1.3, the following differences should be pointed out:

- The overall cost is based on the sum of the individual cost functions and not the product. Thus a perfect match in one transfer function does not necessarily lead to the best overall cost.
- As amplitude and phase errors always have the same scaling, the absolute value of the cost function is a direct measure of the goodness of fit. A cost function of $J \leq 100$ usually yields an acceptable level of accuracy for flight-dynamics modeling.

The cost function from equation B.25 can be written in matrix formulation as

$$L = \frac{20}{N_\omega} \epsilon^T(\theta) W \epsilon(\theta) \tag{B.27}$$

where ϵ is a vector of the magnitude and phase errors between the identified model and the flight data, namely

$$\epsilon(\theta) = \begin{bmatrix} (|H_c|_{dB} - |H|_{dB})_1 \\ (|H_c|_{dB} - |H|_{dB})_2 \\ \cdots \\ (|H_c|_{dB} - |H|_{dB})_{N_{TF}} \\ (\angle H_c - \angle H)_1 \\ (\angle H_c - \angle H)_2 \\ \cdots \\ (\angle H_c - \angle H)_{N_{TF}} \end{bmatrix} \tag{B.28}$$

Each frequency response row in (B.28) is associated with one frequency response to be approximated and is actually composed of N_ω rows corresponding to the number of frequency points. The weighting parameters w_γ (see equation (B.26)) and w_{ap} in the cost function are collected into the single diagonal weighting matrix W that is dependent on the frequency but not on the unknown parameters θ.

The matrix formulation of the cost function in (B.27) is formally equivalent to a Maximum-Likelihood cost function for known output error covariance matrix (compare to (B.8)). Thus the same optimization methods as for the Maximum-Likelihood method can be used for determining the unknown parameters.

B.2.2. Transfer Function Models

A transfer function model can be specified by numerator and denominator polynomials

$$F(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0} \, e^{-\tau s} \tag{B.29}$$

This model has to be normalized by either setting a_n or a_0 to 1. In the model, τ is an equivalent time delay that can be used to account for unmodeled higher order dynamics.

Alternatively, the transfer function can also be displayed in the factored form with poles and zeros.

$$F(s) = k \frac{(s - z_1)(s - z_2) \ldots (s - z_m)}{(s - p_1)(s - p_2) \ldots (s - p_n)} \, e^{-\tau s} \tag{B.30}$$

Here, z_i and p_i respectively denote the zeros and poles of the transfer function and k is the high frequency gain. When the zeros or poles are complex conjugate pairs, they are usually displayed in terms of the damping ratio ζ and natural frequency ω_n as

$$[\zeta, \omega_n] \Leftrightarrow [s^2 + 2\zeta\omega_n s + \omega_n^2]$$

which is also the notation used in the examples in section 10.

In FITLAB both model types (numerator/denominator and pole/zero) can be used in the specification of transfer function identification models. Several frequency responses can be approximated together, if the corresponding transfer function models have the same denominator.

When the frequency response method is used with a polynomial model of the numerator/denominator type, a startup calculation can be used when no starting values for the coefficients are available. The startup algorithm uses a damped Gauss-Newton method to minimize the cost function

$$L = \sum_{k=1}^{N_\omega} w_\gamma \left| H_c(k) - \frac{num(k)}{den(k)} \right| \tag{B.31}$$

where $num(k)$ and $den(k)$ are the numerator and denominator polynomials of the identification model. If the model contains a time delay, it is approximated by a first order linearization. Thus the degree of the numerator polynomial is increased by one and increments to a startup time delay are determined in each iteration. (More detail on accounting for time delays in this way can be found in [35]). After the startup calculation, normal iteration with the cost function from (B.26) is used.

Pole/zero models are more convenient when the approximate location of at least some of the poles and/or zeros is known and bounds can be placed on the corresponding parameters. No startup calculation is available for pole/zero models.

Once the identification run has ended, FITLAB will display the frequency and damping of all poles and zeros of the identified models.

B.3. Minimizing the Cost Function

Adjusting the model parameters so that they minimize the cost function is an optimization problem. The optimization method most widely used for parameter estimation is the Gauss-Newton method (see appendix B.3.1) where, starting from an initial guess, the parameters are obtained iteratively. In each iteration, a system of linear equations for the parameter improvement has to be solved. The Gauss-Newton method

gives information about the accuracy and the correlation of the parameter estimates. The Gauss-Newton method as implemented in FITLAB is enhanced by a line search algorithm for improved performance and allows for parameter bounds.

For systems with discontinuities, where gradient based methods like the Gauss-Newton algorithm fail, the Subplex algorithm (see appendix B.3.2), a variant of the Nelder-Mead simplex minimum search algorithm, is provided as an alternative optimization method. The FITLAB implementation of the Subplex algorithm also allows to specify bounds for the unknown parameters.

If the MATLAB Optimization Toolbox [36] is installed, the optimization routines `fmincon` and `lsqnonlin` can also be used.

The optimization process is stopped when either the relative change of the cost function or the relative change of the unknown parameters from one iteration to the next is smaller than a specified limit or when the specified maximum number of iterations is reached.

All optimization algorithms are described in more detail in the following sections.

B.3.1. Gauss-Newton Method

Finding the parameter vector $\hat{\theta}$ that minimizes the function L from (B.7), (B.11) or (B.27) is an optimization problem. A necessary condition for the optimality is that the partial derivative of L with respect to θ vanishes for $\theta = \hat{\theta}$.

$$\left.\frac{\partial L(z|\theta)}{\partial \theta}\right|_{\theta=\hat{\theta}} = L'(z|\hat{\theta}) = 0 \tag{B.32}$$

Applying Newton's method to this equation yields an iterative solution starting from an initial guess θ_0 via

$$\theta_{i+1} = \theta_i + \Delta\theta = \theta_i - \frac{L'(z|\theta_i)}{L''(z|\theta_i)} \tag{B.33}$$

with

$$L''(z|\theta_i) = \left.\frac{\partial^2 L(z|\theta)}{\partial \theta^2}\right|_{\theta=\theta_i} \tag{B.34}$$

For given R as well as for the Maximum Likelihood estimate from (B.9), the partial derivative of L with respect to R vanishes so that the derivatives of L with respect to θ

are

$$L'(z|\theta) = -\sum_{k=0}^{N-1} \left[\frac{\partial y(k)}{\partial \theta}\right]^T R^{-1} [z(k) - y(k)] \tag{B.35}$$

$$L''(z|\theta) = \sum_{k=0}^{N-1} \left[\frac{\partial^2 y(k)}{\partial \theta^2}\right]^T R^{-1} [z(k) - y(k)] + \sum_{k=0}^{N-1} \left[\frac{\partial y(k)}{\partial \theta}\right]^T R^{-1} \left[\frac{\partial y(k)}{\partial \theta}\right] \tag{B.36}$$

The first term in (B.36) requires the second derivative of y with respect to θ that can only be determined with great computational effort. It can be shown that this derivative vanishes at the optimum, i.e. for $\theta = \hat{\theta}$. Therefore, the first term can be neglected without affecting the convergence when the starting values are good enough.

The optimization algorithm with this approximation for $L''(z|\theta)$ is called Gauss-Newton or modified Newton-Raphson algorithm. The resulting system of linear equations for the parameter improvement in each iteration is

$$\left\{\sum_{k=0}^{N-1} \left[\frac{\partial y(k)}{\partial \theta}\right]^T R^{-1} \left[\frac{\partial y(k)}{\partial \theta}\right]\right\} \cdot \Delta\theta = \left\{\sum_{k=0}^{N-1} \left[\frac{\partial y(k)}{\partial \theta}\right]^T R^{-1} [z(k) - y(k)]\right\} \tag{B.37}$$

In equation (B.37), $\partial y/\partial \theta$ is the so called sensitivity matrix. For nonlinear systems, the elements of this matrix are suitably approximated by finite differences

$$\frac{\partial y}{\partial \theta} \approx \frac{y(\theta + \delta\theta) - y(\theta)}{\delta\theta} \tag{B.38}$$

The Gauss-Newton method for solving the optimization problem automatically yields information about the accuracy of the estimates. The information matrix J is equal to the matrix on the left hand side of the linear system of equations (B.37) for the parameter update,

$$J = \sum_{k=0}^{N-1} \left[\frac{\partial y(k)}{\partial \theta}\right]^T R^{-1} \left[\frac{\partial y(k)}{\partial \theta}\right] \tag{B.39}$$

For a bias free and efficient estimation method, the inverse of the information matrix is equal to the estimation error covariance matrix (Cramér-Rao).

$$P = J^{-1} \tag{B.40}$$

As the Maximum Likelihood estimation is asymptotically bias free and efficient, this equation holds true when the number of data samples is large enough. Thus the

standard deviations and correlation coefficients of the parameters can be calculated as

$$\sigma(\theta_i) = \sqrt{p_{ii}} \tag{B.41}$$

and

$$\rho(\theta_i, \theta_j) = \frac{p_{ij}}{\sqrt{p_{ii} p_{jj}}} \tag{B.42}$$

As the cost function in the frequency domain is very similar to the one in the time domain, applying the Gauss-Newton method leads to a similar system of equations for the parameter improvement vector $\Delta\theta$.

$$\left\{ \sum_{k=0}^{N-1} \left[\frac{\partial y(k)}{\partial \theta} \right]^* S^{-1} \left[\frac{\partial y(k)}{\partial \theta} \right] \right\} \cdot \Delta\theta = \left\{ \sum_{k=0}^{N-1} \left[\frac{\partial y(k)}{\partial \theta} \right]^* S^{-1} \left[z(k) - y(k) \right] \right\} \tag{B.43}$$

Line Search

Generally, the standard Gauss-Newton method using the full step of the parameter update as computed from equation (B.37) or (B.43) performs very well. However, due to initial parameter values far from the optimum in combination with a non-quadratic cost function (with regard to the parameters to be identified), the Gauss-Newton algorithm may show intermediate local divergence. To overcome this difficulty, a line search algorithm is added where the parameter step $\Delta\theta$ serves as search direction. The update equation (B.33) is thus replaced by

$$\theta_{i+1} = \theta_i + \alpha_i * \Delta\theta \tag{B.44}$$

where the scaling factor α_i is determined as follows.

First, two values $\alpha_{rear}, \alpha_{front}$ are sought, that bracket a minimum α_{min}, i.e.

$$\alpha_{rear} \leq \alpha_{min} \leq \alpha_{front} \tag{B.45}$$

such that

$$L(\alpha_{rear}) \geq L(\alpha_{min}) \quad \text{and} \quad L(\alpha_{min}) \leq L(\alpha_{rear}) \tag{B.46}$$

holds for the corresponding cost function values.

Brent's algorithm (see chapter 10.2 of [24]) to determine the optimal value α_i for the scaling factor.

Solving for the Parameter Improvement

In [37] it is illustrated that equation (B.37) for the parameter improvement can be interpreted as the normal equations of the following weighted least squares problem:

$$\begin{bmatrix} \tilde{R} & & \\ & \ddots & \\ & & \tilde{R} \end{bmatrix} \cdot \begin{bmatrix} H(1) \\ \vdots \\ H(N) \end{bmatrix} \Delta\theta = \begin{bmatrix} \tilde{R} & & \\ & \ddots & \\ & & \tilde{R} \end{bmatrix} \cdot \begin{bmatrix} z(1) - y(1) \\ \vdots \\ z(N) - y(N) \end{bmatrix}; \tilde{R}_{ij} = 1/\sqrt{R_{ij}} \tag{B.47}$$

Therefore, solving this system is equivalent to solving (B.37). Equations (B.47) can be solved by any method feasible for least squares problems, with QR decomposition and SVD being the numerically best methods. Solving (B.47) with SVD is essentially a Singular Value Decomposition of the sensitivity matrix and the results are, therefore, easily interpretable.

FITLAB uses QR decomposition for each time slice to reduce the number of equations and SVD for solving the final system of equations.

Considering Bounds

The parameters to be estimated are frequently subject to simple bounds. Typical applications are parameters with a physical meaning, which are often constrained to lie within a certain range (e.g. damping parameters that have to be negative if the system is known to be stable).

The following sections describe the extension of the two optimization algorithms to bounded problems. The linear-constraint optimization problem where the constraints are simple bounds is formulated according to [38]:

$$min(L(\theta)) \quad \text{subject to} \quad \theta_{min} \leq \theta \leq \theta_{max} \tag{B.48}$$

An active set strategy is implemented in FITLAB to solve problem (B.48). Starting from the initially specified parameter values θ_0, an active set containing the indices of the variables reaching the bounds is formed and updated in every iteration. A variable is called a free variable, if it is within the permissible bounds, and hence not in the active set. The Gauss-Newton search directions for the free variables are computed according to equation (B.37) for the free parameters:

$$\left\{ \sum_{k=0}^{N-1} \left[\frac{\partial y(k)}{\partial \theta_{free}} \right]^T R^{-1} \left[\frac{\partial y(k)}{\partial \theta_{free}} \right] \right\} \cdot \Delta\theta_{free} = \underbrace{\left\{ \sum_{k=0}^{N-1} \left[\frac{\partial y(k)}{\partial \theta_{free}} \right]^T R^{-1} [z(k) - y(k)] \right\}}_{-G_{free}} \tag{B.49}$$

The parameter updates resulting from equation (B.49) are checked for the specified bounds, and any violation leads to inclusion of the corresponding parameter into the active set. For such parameters the values are set to the respective bounds and the search direction of equation (B.49) to zero. For the remaining parameters a new point is computed using a line search.

An important aspect of the active set strategy is to appropriately alter the active set as the optimization progresses. The active set is changed whenever a free variable hits its bounds during an iteration. Furthermore, if the Kuhn-Tucker optimality conditions

$$\begin{array}{llll} (G_i < 0, & \text{for} & \theta_i = \theta_{i,max}) & \text{or} \\ (G_i > 0, & \text{for} & \theta_i = \theta_{i,min}) & \end{array} \tag{B.50}$$

are not satisfied for any of the variables in the active set, then those variables are dropped from the active set and are made free.

B.3.2. Subplex Algorithm

For systems with discontinuities, gradient based optimization methods like the Gauss-Newton method usually fail and simplex based methods are used instead. The standard Simplex algorithm [39] for the case of n unknown parameters starts with an n-dimensional simplex formed by $n+1$ points in the parameter space. Iteratively, the worst point is replaced by a better point through reflection, expansion, or contraction of the simplex. The algorithm ends when all points are close to each other.

The standard Simplex algorithm works well for problems with few unknowns but often fails for more than five unknown parameters. Thus, the Subplex algorithm developed by Rowan in [40] is implemented in FITLAB in a slightly modified version. The Subplex algorithm subdivides the problem space into orthogonal subspaces of lower dimension (between 2 and 5) and then uses the Simplex algorithm on these subspaces. The main subspace is chosen such that the direction of main improvement of the last iteration step is included in this subspace. More detail about the Subplex algorithm can be found in [40, 41].

The basic Simplex and Subplex algorithms do not account for parameter bounds. In [42] a method for considering bounds in the Simplex algorithms is presented. This approach has been successfully integrated into the Subplex algorithm as used in FITLAB [41].

B.3.3. Optimization Toolbox Routines fmincon and lsqnonlin

The routine `fmincon` from the MATLAB optimization toolbox allows to find a minimum of a function subject to constraints. For the implementation in FITLAB, only the lower and upper bounds for the variable parameters are utilized. The maximum number of iterations is taken from the FITLAB options structure. All other options for `fmincon` are left at their default values.

The routine `lsqnonlin` solves a nonlinear least-squares problem. Minimizing the cost function of the frequency response method is such a least-squares problem as shown in (B.27). The cost function of the Maximum Likelihood method (see (B.7)) only reduces to a least-squares problem if the measurement error covariance matrix R resp. the spectral density matrix S of the measurement noise is given (see (B.8)). This option is, however, not implemented in FITLAB.

This means that `lsqnonlin` can properly be used only for identification with the frequency response method, i.e. only for polynomial models and for linear models that are identified with the FR method.

To enable the use of `lsqnonlin` also for the ML method, the following approach was used: A simulation with the starting values of the parameters is used to determine a starting value for R from (B.9). During the iteration, R is updated using the same equation after each iteration. This approach, however, does not implement a true ML method and convergence is not guaranteed.