ETAS ASCMO-MOCA V5.15

User Guide

www.etas.com

ASCMO-MOCA V5.15 I User Guide R01 EN I 04.2025

Contents

1 Introduction

In this chapter, you can find information about the intended use, the addressed target group, and information about safety and privacy related topics.

Please adhere to the ETAS Safety Advice (**Help > Safety Advice**) and to the safety information given in the user documentation.

ETAS GmbH cannot be made liable for damage which is caused by incorrect use and not adhering to the safety messages.

1.1 Demands on Technical State of the Product

The following special requirements are made to ensure safe operation:

- Take all information on environmental conditions into consideration before setup and operation (see the documentation of your computer, hardware, etc.).

1.2 Intended Use

The ETAS ASCMO tool family is intended for offline data-based modeling, model-based calibration, or efficient optimization of parameters in physics-based models. It is not intended to operate directly in a running system.

With ASCMO-STATIC and ASCMO-DYNAMIC, it is possible to accurately model the behavior of complex systems based on a small set of measurement data. This model can either be used to analyze and optimize input parameters or as a black box plant model in other simulation environments. In contrast, ASCMO-MOCA typically uses existing physics based-models with a defined structure to calibrate and optimize the parameters of the model itself. The results are a suggestion and must be additionally validated before further processing.

ETAS GmbH cannot be held liable for damage which is caused by incorrect use and not adhering to the safety information. See **Help > Safety Advice**.

1.3 Target Group

This product is intended for trained and qualified personnel in the development and calibration sector of motor vehicle ECUs. Technical knowledge in measuring and control unit engineering is a prerequisite.

1.4 Classification of Safety Messages

Safety messages warn of dangers that can lead to personal injury or damage to property:

DANGER

DANGER indicates a hazardous situation that, if not avoided, will result in death or serious injury.

WARNING

WARNING indicates a hazardous situation that, if not avoided, could result in death or serious injury.

CAUTION

CAUTION indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.

NOTICE

NOTICE indicates a situation that, if not avoided, could result in damage to physical property.

ATTENTION

ATTENTION indicates a situation that, if not avoided, could result in damage to digital property like data loss, data corruption and system vulnerability.

1.5 Safety Information

NOTICE

Damage due to wrong calibration data

Wrong usage of calibrations derived from ASCMO-MOCA model can lead to engine or test bench damage.

Compare measured data and model created data with Residual Analysis feature after the optimization or before exporting at the latest. Feature is accessible via Analysis › Residual Analysis › Training and Test Data › Absolute Error Analysis.

See "Performing the optimization" on page 124, export options in **Parameters** Step or **Optimization** Step, and 6.9 "Step 7: Export" on page 125.

NOTICE

Potential malicious code from external source

For FMU, ASCET and TSiM model types, ASCMO-MOCA executes an external runnable during model evaluation.

Make sure that the external runnable of the model comes from a trustworthy source.

1.6 Data Protection

If the product contains functions that process personal data, legal requirements of data protection and data privacy laws shall be complied with by the customer. As the data controller, the customer usually designs subsequent processing. Therefore, he must check if the protective measures are sufficient.

1.7 Data and Information Security

To securely handle data in the context of this product, see the next sections about data and storage locations as well as technical and organizational measures.

1.7.1 Data and Storage Locations

The following sections give information about data and their respective storage locations for various use cases.

1.7.1.1 License Management

When using the ETAS License Manager in combination with user-based licenses that are managed on the FNP license server within the customer's network, the following data are stored for license management purposes:

Data

- Communication data: IP address
- User data: Windows user ID

Storage location

- FNP license server log files on the customer network

When using the ETAS License Manager in combination with host-based licenses that are provided as FNE machine-based licenses, the following data are stored for license management purposes:

Data

- Activation data: Activation ID
  - Used only for license activation, but not continuously during license usage

Storage location

- FNE trusted storage

  `C:\ProgramData\ETAS\FlexNet\fne\license\ts`

1.7.2 Technical and Organizational Measures

We recommend that your IT department takes appropriate technical and organizational measures, such as classic theft protection and access protection to hardware and software.

2 About ASCMO-MOCA

ASCMO-MOCA is a tool for **Mo**deling and **Ca**librating functions with given data. These functions consist of mathematical operations on changeable parameters, such as lookup tables. The goal is to minimize the deviation of the function's output from the given data. The function's parameters are adapted (calibrated) with an optimizer to minimize this deviation. Additional constraints, such as smoothness and gradients of curves/maps, can be considered.

The results can be visualized in different views, such as scopes and scatter plots. A residuals analysis allows to detect problems, e.g., outliers.

ASCMO-MOCA comes in two versions: the full version and the runtime version. The full version allows modeling of the function, definition of an optimization sequence, and the optimization itself. The runtime version opens existing projects from the full version, allows data import, and enables the start of the optimization, but not the definition of the function or the optimization sequence.

The building blocks of the function in ASCMO-MOCA are scalars, lookup tables, RBF(Radial Basis Function)-Nets, and models from other sources like Simulink®.

A time-independent function without inner states and loops can be directly modeled in ASCMO-MOCA. More complex, time-dependent functions are to be modeled in other tools, such as Simulink®. ASCMO-MOCA then uses the external tool during the optimization.

2.1 Basics

ETAS ASCMO-MOCA enables the optimization of parameters in physics-based models, such as those used in the ECU and simulation environments. Various plant models and controller models can be loaded, connected, or modeled for this purpose. It is also possible to load measurement data, import and export model parameters, and define optimization tasks. ASCMO-MOCA provides a wide range of functions and options for visualizing and analyzing the data, as well as the models used. Powerful algorithms can optimize a large number of free parameters simultaneously while considering constraints such as smoothness or monotonicity.

A common use for ASCMO-MOCA is in optimizing the prediction quality of ECU models (virtual sensors) for e.g., torque or exhaust-gas temperature, minimizing the deviation of the model prediction from real measurements on the engine test bench or in the vehicle for all measuring points.

Another application is optimizing emissions and fuel consumption for complex internal combustion engines in dynamic/transient driving cycles. This requires linking classic ASCMO data-based models to parts of the ECU software. Such linkages and the joint optimization of different subcomponents are straightforward in ASCMO-MOCA.

Since the methodology used in ASCMO-MOCA is not limited to the internal combustion engine, the tool is also used in areas such as electric mobility (e.g., charging strategy) and component development.

2.2 Finding Out More

In addition to this User Guide, the Online Help is recommended, especially when working with the user interface. It can be accessed via **Help › Online Help** or context-sensitive with F1 in the currently open operating window.

For help on the P-code version functions, use **Help › Interface Help**.

3 Installation

Before installing, make sure your computer meets the system requirements (see System requirements MOCA ASCMO). You must ensure that you have the necessary user rights and a network connection.

If you want to use the product offline, you need to borrow the license in the ETAS License Manager (**LiMa** main window > **License** > **Borrow selected licenses / Borrow all licenses**). See Licensing for more information.

3.1 System Requirements

The following minimum system requirements must be met:

Required Hardware	1,0 GHz PC 4 GB RAM Graphics with a resolution of at least 1024 x 768, 32 MB RAM
Required Operating System	Windows® 10, Windows® 11
Required Free Disk Space	4 GB (not including the size for application data)

The following system requirements are recommended:

Recommended Hardware	4,0 GHz Quad-Core PC or equivalent 32 GB RAM Graphics with a resolution of 1680 x 1050, 128 MB RAM
Recommended Operating System	Windows® 10, Windows® 11
Recommended Free Disk Space	> 4 GB

3.2 Software Requirements

ETAS ASCMO requires and installs the MATLAB® Compiler Runtime 2022b. It also requires the .Net Framework V4.6, which is included with Windows® 10/11.

There are no additional software requirements for the installation of the ETAS ASCMO base product and add-ons. Any missing software components will be installed during the installation.

To use the ETAS ASCMO add-on *Software Developer Kit (SDK)*, a MATLAB® version R2021b up to R2023b and the MATLAB® *Optimization Toolbox* and *Statistics Toolbox* are required.

3.3 Installing

Install ETAS ASCMO

1. Go to the directory where the ASCMOinstallation file is located.
2. Double-click `Setup_ETAS-ASCMO_Vx_x_20xxx.exe`.

   The **License Agreement** window opens.

3. Read the license agreement carefully, then activate the **I accept the agreement** option.
4. Click **Next**.

   If you have already installed an ASCMO version, the path (destination location and start menu folder) of the initial installation will be used and steps 5 to 8 will not be available.
5. In the **Set Destination Location** window, accept the default folder or click **Browse** to select a new directory.
6. Click **Next**.
7. In the **Select Start Menu Folder** window, accept the default folder or click **Browse** to select a new folder.
8. Click **Next**.

   i. Activate the **Create a desktop icon** checkbox if you want to create an icon on the desktop.
   ii. Choose whether to force the installation of the MATLAB® Compiler Runtime or to install it only if it is not already installed.
   iii. If necessary, activate**Install Prerequisites**.
9. Click **Next**.
10. In the **Ready to Install** window, click **Install** to start the installation.

   or

   If you want to change the settings, click **Back**.

   The installation process begins. A progress indicator shows the installation's progress. When the installation is complete, the **Completing the ETAS ASCMO Setup Wizard** window opens.
11. Click **Finish**.

⇨ The installation is complete. ASCMO can be started.

3.4 Files and Directories

All files belonging to the program are located in the `<installation>` directory selected during the installation, and in additional subfolders of this directory.

By default, `<installation>` is `C:\Program Files\ETAS\ASCMO x.x`.

Start Menu

After successful installation, the folder you specified in the **Select Start Menu Folder** window with the following entries is added to the **Windows Start** menu.

- **ASCMO Desk V5.15**
  Starts the ASCMO-DESK window, where you can start the ETAS ASCMO components.
- **ASCMO Dynamic V5.15**
  Starts ASCMO-DYNAMIC.
- **ASCMO ExpeDes Dynamic V5.15**
  Starts ASCMO-DYNAMIC ExpeDes.
- **ASCMO ExpeDes V5.15**
  Starts ASCMO-STATIC ExpeDes.
- **ASCMO MOCA Runtime V5.15**
  Starts the ASCMO-MOCA Runtime environment with limited functionality.
- **ASCMO MOCA V5.15**
  Starts ASCMO-MOCA.
- **ASCMO Static V5.15**
  Starts ASCMO-STATIC.
- **Manuals and Tutorials**
  Opens the ASCMO documentation directory (`<installation>\Manuals`), which contains the following information and documents.
  - `ASCMOInterfaceDoc` – a folder with interface documentation.
  - `Examples` – a folder with different example data (e.g., ASCMO projects, MF4, DCM, XLS or FMU files, templates, plugins, etc.).
  - `HTML folder` – online help files for the installed components (available via ‹F1›).
  - `ASCMO-DYNAMIC_V5.15_User_Guide_*.pdf` – User Guide with tutorials for the basic functions of ASCMO-DYNAMIC.
  - `ASCMO-STATIC_V5.15_User-Guide_*.pdf` – User Guide with tutorials for the basic functions of ASCMO-STATIC.
  - `ASCMO-MOCA_V5.15_User-Guide_*.pdf` – User Guide with a tutorial for the basic functions of ASCMO-MOCA.

P-code Files

Of special interest are the P-code files for MATLAB® and Simulink® in the *<installation>*`\pCode\ascmo` directory.

For more information, see "P-Code Version" below.

3.5 P-Code Version

The P-code version also allows you to start ETAS ASCMO within MATLAB®.

Prerequisites

The P-code version requires an installation of MATLAB® R2021b up to R2023b. In addition, the following MATLAB® toolboxes are required:

- Optimization Toolbox™
- Statistics and Machine Learning Toolbox ™

Executing ETAS ASCMO

In MATLAB®, change to the directory *<installation>*`\pCode\ascmo`. In the command window, enter one of the following commands:

command	action
`AscmoDesk`	Starts ASCMO-DESK.
`ascmo static`	Starts ASCMO-STATIC.
`ascmo expedes`	Starts ASCMO-STATIC ExpeDes.
`ascmo dynamic`	Starts ASCMO-DYNAMIC.
`ascmo expedesdynamic`	Starts ASCMO-DYNAMIC ExpeDes.
`ascmo moca`	Starts ASCMO-MOCA.
`ascmo mocaruntime`	Starts ASCMO-MOCA Runtime.
`ascmo cyclegenerator`	Starts the standalone ASCMO-Cycle Generator.
`ascmo essentials`	Starts ASCMO Essentials.

All further steps in an ETAS ASCMO tool can be automated using commands whose description can be found in the main menu under **Help › Interface Help**.

3.6 Licensing

A valid license is required to use the software. You can obtain a license in one of the following ways:

- from your tool coordinator
- via the self-service portal on the ETAS website at www.etas.-com/support/licensing
- via the ETAS License Manager

To activate the license, you must enter the Activation ID that you received from ETAS during the ordering process.

For more information about ETAS license management, see the ETAS License Management FAQ or the ETAS License Manager help.

To open the ETAS License Manager help

The ETAS License Manager is available on your computer after the installation of any ETAS software.

1. From the Windows Start menu, select **E** > **ETAS** > **ETAS License Manager**.

   The ETAS License Manager opens.
2. Click in the ETAS License Manager window and press F1.

   The ETAS License Manager help opens.

3.7 Uninstalling

Note

You cannot uninstall specific components. The procedure uninstalls **all** ETAS ASCMO components.

Uninstall ETAS ASCMO

1. Go to the directory where the ASCMOinstallation file is located.

   Start the uninstall procedure.

   A warning message opens.
2. Double-click `unins000.exe`.

   A warning message opens.
3. To completely remove ETAS ASCMO and all its components, click **Yes**.

   The uninstallation process begins. When the process is complete, a message window opens.
4. Click **OK** to complete the uninstallation.

⇨ ETAS ASCMO and all its components are successfully uninstalled.

4 Basics of ASCMO-MOCA

ASCMO-MOCA enables optimization of model parameters and minimizes the deviation of model prediction and desired output values.

E.g. modern vehicle ECUs contain physics based models to replace or monitor real sensors. Such a physics based model is generic, but must be adapted to an actual engine. Parameters (maps/curves/scalars) are optimized using real measurements, e.g., from test bench or vehicle.

The model can be represented in ASCMO-MOCA as a set of formulas entered by the user. Alternatively, existing models, e.g. from Simulink®, can be used.

In this chapter, you can find a description of the basic concepts of ASCMO-MOCA.

These are the following:

- **"Fields of Application of ASCMO-MOCA" on the next page**

  This section provides a general overview of the wide range of application fields in ASCMO-MOCA.
- **"Elements of the ASCMO-MOCA User Interface " on page 80**

  This section provides an brief overview of the user interface key elements of ASCMO-MOCA.
- **"Data" on page 19**

  This section provides information on import, analysis and preprocessing of measured data.
- **"Assessment of the Input Data" on page 19**

  In this section you will find information on how you can assess the quality of the input data used by ASCMO-MOCA for the parameter optimization.
- **"Models" on page 40**

  This section provides information on importing and using external models in ASCMO-MOCA.
- **"Function" on page 43**

  This section provides information on how to create a model by specifying a set of formulas that form a function.
- **"Parameters " on page 27**

  This section contains general information about the optimization of parameters within ASCMO-MOCA.
- **"Available Types of Parameters" on page 27**

  This section provides a brief overview of the various types of parameters that can be used in the function (see "Step 5: Build Up the Function" on page 115) for optimization (see "Step 6: Optimization" on page 122).

- **"Optimization" on page 51**

  This section contains a description of the different optimization methods and the optimization criteria that can be used for the parameter optimization.

4.1 Fields of Application of ASCMO-MOCA

This section provides a general overview of the wide range of application fields of ASCMO-MOCA.

4.1.1 Calibration of ECU Sensor Data

- Optimization of parameters
- Optimization of time-dependent (dynamic) functions
- Parameterization of ECU models (cylinder fill, torque, ...)

The use of ASCMO-MOCA in the area of *calibration* offers a series of advantages:

- Significant increase in efficiency through reduced measuring and analysis efforts
- Improved complexity handling
- Improved data quality
- Multiple use of models

4.1.2 Research, Function and System Development

- Quick calibration and evaluation of experimental engines
- Use of models of real engines for test and development of new functions (e.g., controller strategies)
- Analysis and optimization of unknown systems.

The advantages in the area of *research and development* lie primarily in a quicker and more improved system understanding, coupled with a variety of possibilities for impact analysis.

4.1.3 Fields of Application of ASCMO-MOCA Runtime

The Runtime version of ASCMO-MOCA is designed to fulfill the special requirements of using the software with limited access to special functionalities. Reasons for doing this are to hide away special IP or to avoid that an user changes something critical.

This version can be either installed and used in parallel to the main (Developer) version or as standalone.

> **Note**
>
> The Runtime version does not allow to create or modify functions.

The following activities can be carried out with ASCMO-MOCA Runtime:

- Import of stationary or transient data followed by name-mapping.
- Definition of conversion rules (Conversion Parameters / formulas).
- Import, export, creation, deletion and editing of parameters and system constants.
- Iterative optimization and calibration of parameters.

The installation of ASCMO-MOCA Runtime is particularly recommended if the one who has created the project with the optimization task is not the same as the one who executes the optimization.

This supports intellectual property protection and safety:

- You do not have to share special know-how about the function or the optimization logic with others.
- No critical parameters and settings are changed by the user who performs the optimization. Such changes could result in unexpected behavior.

4.2 Data

The first steps in ASCMO-MOCA are import, analysis and preprocessing of measured data. These steps are performed in the Data Step.

For more information, see the following subsections and the online help.

- "Assessment of the Input Data" below
- "Step 1: Data Import" on page 88 (tutorial)

4.2.1 Assessment of the Input Data

This section provides information on how you can assess the quality of the input data used by ASCMO-MOCA for the parameter optimization.

- "Tabular Representation of All Model-Related Data" on the next page
- "Checking the Relevance of the Inputs" on the next page
- "Function Assessment and Improvement" on page 21
  - "Graphical Analysis of Data and Function Nodes" on page 21
  - "Residual Analysis" on page 22
  - "Improving the Model Quality" on page 25
- "Variables RMSE and R2" on page 25
- "Function Evaluation Using RMSE and R2" on page 26

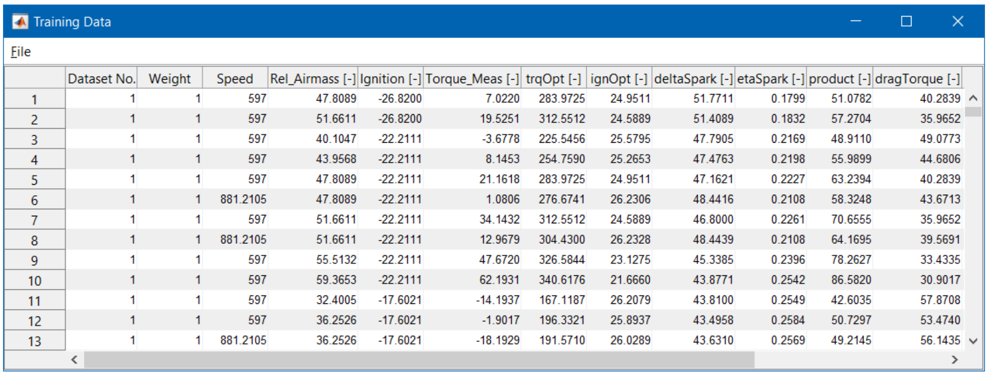
4.2.1.1 Tabular Representation of All Model-Related Data

The **Analysis › Data Table › Training Data/Test Data/Training and Test Data** menu options open a table that displays the imported data columns, converted data columns from conversion formulas and additionally calculated nodes from the function. If optimization criteria are defined, also the residuals are displayed.

> **Note**
>
> The data in the "All Data" window cannot be modified.

The following values are shown in the table in detail:

- imported data
- converted data (conversion rules)
- nodes (from functions)
- residuals (from optimization criteria)



Training Data

File

	Dataset No.	Weight	Speed	Rel_Airmass [-]	Ignition [-]	Torque_Meas [-]	trqOpt [-]	ignOpt [-]	deltaSpark [-]	etaSpark [-]	product [-]	dragTorque [-]
1	1	1	597	47.8089	-26.8200	7.0220	283.9725	24.9511	51.7711	0.1799	51.0782	40.2839
2	1	1	597	51.6611	-26.8200	19.5251	312.5512	24.5889	51.4089	0.1832	57.2704	35.9652
3	1	1	597	40.1047	-22.2111	-3.6778	225.5456	25.5795	47.7905	0.2169	48.9110	49.0773
4	1	1	597	43.9568	-22.2111	8.1453	254.7590	25.2653	47.4763	0.2198	55.9899	44.6806
5	1	1	597	47.8089	-22.2111	21.1618	283.9725	24.9511	47.1621	0.2227	63.2394	40.2839
6	1	1	881.2105	47.8089	-22.2111	1.0806	276.6741	26.2306	48.4416	0.2108	58.3248	43.6713
7	1	1	597	51.6611	-22.2111	34.1432	312.5512	24.5889	46.8000	0.2261	70.6555	35.9652
8	1	1	881.2105	51.6611	-22.2111	12.9679	304.4300	26.2328	48.4439	0.2108	64.1695	39.5691
9	1	1	597	55.5132	-22.2111	47.6720	326.5844	23.1275	45.3385	0.2396	78.2627	33.4335
10	1	1	597	59.3653	-22.2111	62.1931	340.6176	21.6660	43.8771	0.2542	86.5820	30.9017
11	1	1	597	32.4005	-17.6021	-14.1937	167.1187	26.2079	43.8100	0.2549	42.6035	57.8708
12	1	1	597	36.2526	-17.6021	-1.9017	196.3321	25.8937	43.4958	0.2584	50.7297	53.4740
13	1	1	881.2105	36.2526	-17.6021	-18.1929	191.5710	26.0289	43.6310	0.2569	49.2145	56.1435

Fig. 4-1: The "All Data" window

4.2.1.2 Checking the Relevance of the Inputs

During data import, you can check the inputs' relevance to the outputs (see also "To check the relevance of the inputs" on page 91).

If you do so, a polynomial stepwise regression is done with the inputs and outputs. The stepwise regression ignores inputs with a significance ‹ 5% and can find dependent inputs.

If, for example, the training data contains the inputs *speed*, *load* and *speed + load*, then one of the inputs has a low significance.

The order of the inputs is important. After the stepwise regression, the inputs are permuted column by column, and a pseudo RMSE is calculated per input, to get a heuristic of the input's relevance. The findings are then plotted in the "Relevance of Inputs" window.

4.2.1.3 Function Assessment and Improvement

The **Analysis** menu offers a number of functions to compare the model output prediction with the measured data of the function output. Specifically, these are:

- Graphical analysis of the measured data and the function nodes

  See "Graphical Analysis of Data and Function Nodes" below for details.
- Residual analysis

  See "Residual Analysis" on the next page for details.

Graphical Analysis of Data and Function Nodes

The scatter plots (**Analysis** Menu) in the following windows provide a graphical control of the measurement data and the function evaluation:

- "Data - Training Data/Test Data/Training and Test Data"
- "FunctionNode - Training Data/Test Data/Training and Test Data"
- "Data and Nodes - Training Data/Test Data/Training and Test Data"

When analyzing the measurement data, the following points should be considered particularly:

- Have all data been varied in accordance to the Design of Experiment (DoE) and has the measured system remained in the intended operating mode?
- Are the output values in a physically reasonable range?
- Are there outliers included which must be removed if appropriate?

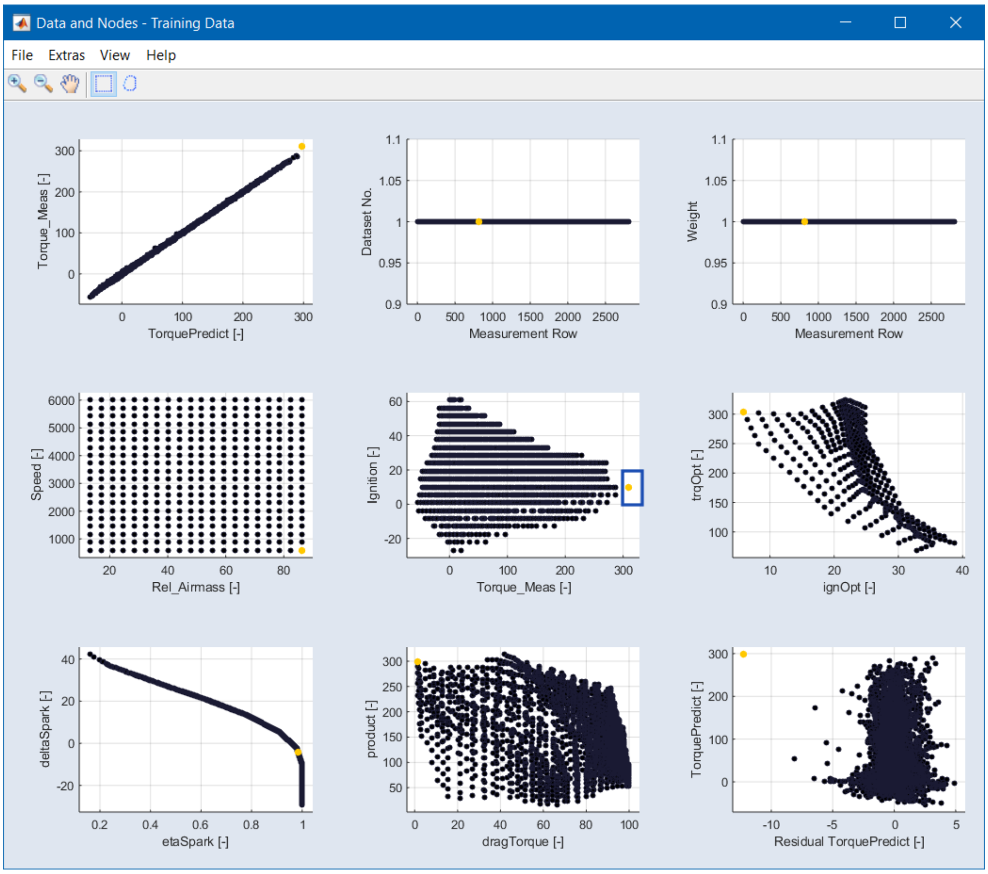


Fig. 4-2: The "Data and Nodes" window

Residual Analysis

Residuals are the deviation of the data calculated according to the optimization criteria to the measured data.

Three types of residual analysis are available:

- **Absolute Error Analysis**

  For the *Absolute Error Analysis*, all residuals are displayed:

  $$Y_{measured} - Y_{predicted}$$

- **Relative Error Analysis**

  For the *Relative Error Analysis* the quotient from the residue and the measured value is displayed:

  $$100 \cdot \left(\frac{Y_{measured} - Y_{predicted}}{Y_{measured}} \right)$$

  Therefore, a percentage deviation is displayed.

- **Studentized Error Analysis**

  When performing a *Studentized error analysis*, the quotient from the residual and the RMSE 4.2.2.1 "RMSE (Root Mean Squared Error)" on

page 25 is displayed:

$$\frac{Y_{measured} - Y_{predicted}}{RMSE}$$

Thus, the error based on the RMSE is shown.

Residual analysis is performed via the **Analysis › Residual Analysis › *** menu options. These menu options open four plot windows:

"Histogram" Window

The "Histogram" window displays the current error distribution (blue bars) on the total number of values for the predicted function output. The normal distribution fit (red line) is drawn additionally. This function enables you to validate whether the current error distribution fits to the normal distribution or not.

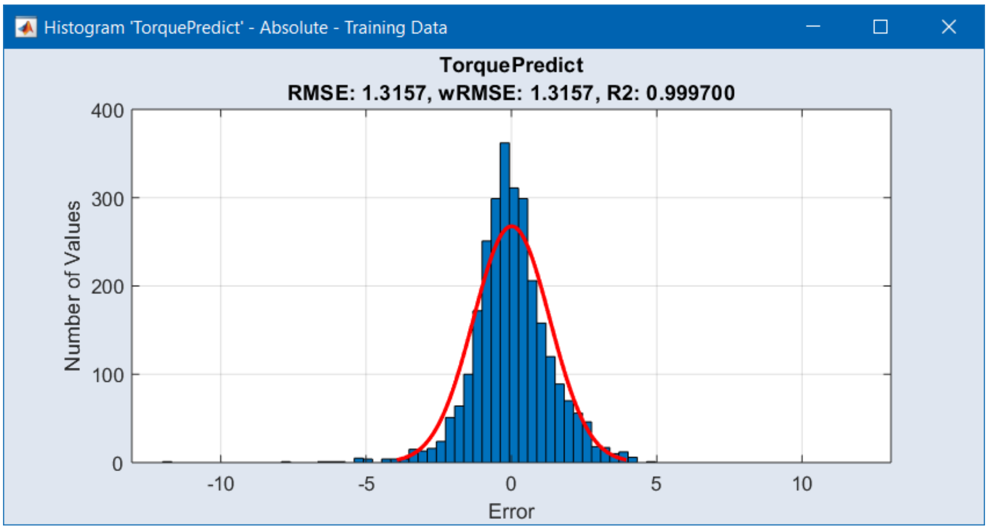


Fig. 4-3: The "Histogram" window

"Residuals over Inputs" Window

This window shows several scatter plots: data set number, Active flag and weight against measurement number, as well as the errors (absolute, relative, or studentized) of the computed data against the measured data. For a detailed description, see "Improving the Model Quality" on page 25.

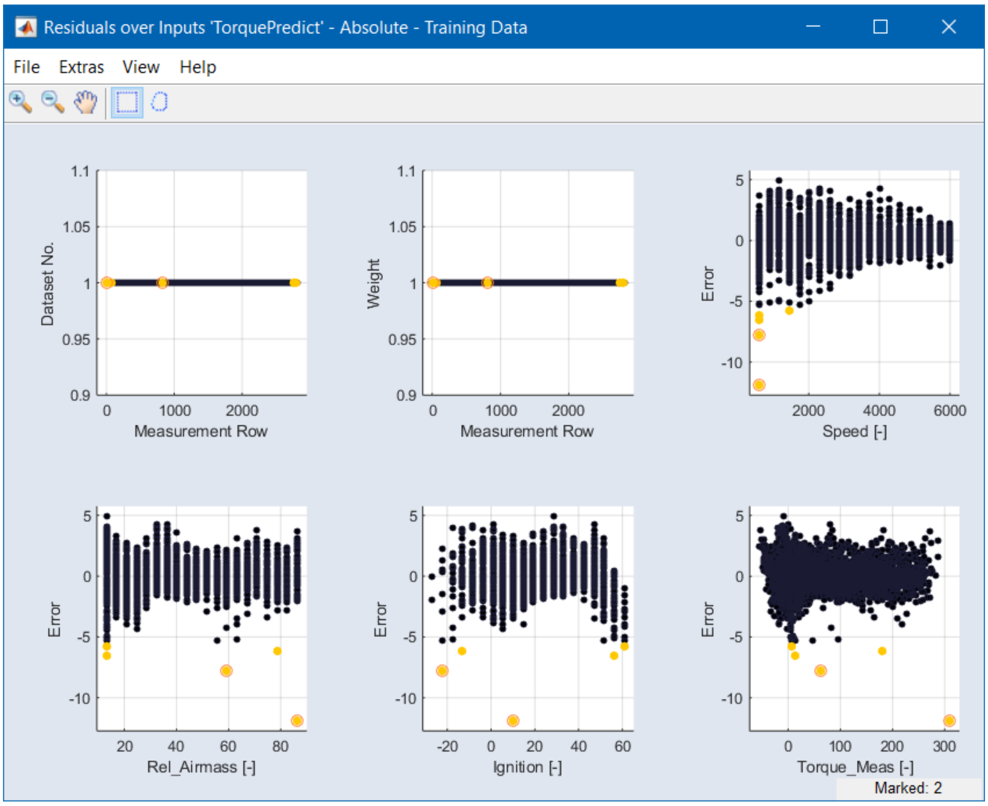


Fig. 4-4: The "Residuals over Inputs" window

"Residuals over Outputs" Window

This window shows scatter plots of the errors (absolute, relative, or studentized) of the computed data against the function nodes.

"Measured vs. Predicted" Window

In this window, the model output is displayed on the X axis and the measuring points are displayed on the Y axis. A perfect match between the two would result in a "pearl necklace" (y = x). The further the points are removed from the y = x line, the greater the difference between measurement and model output.

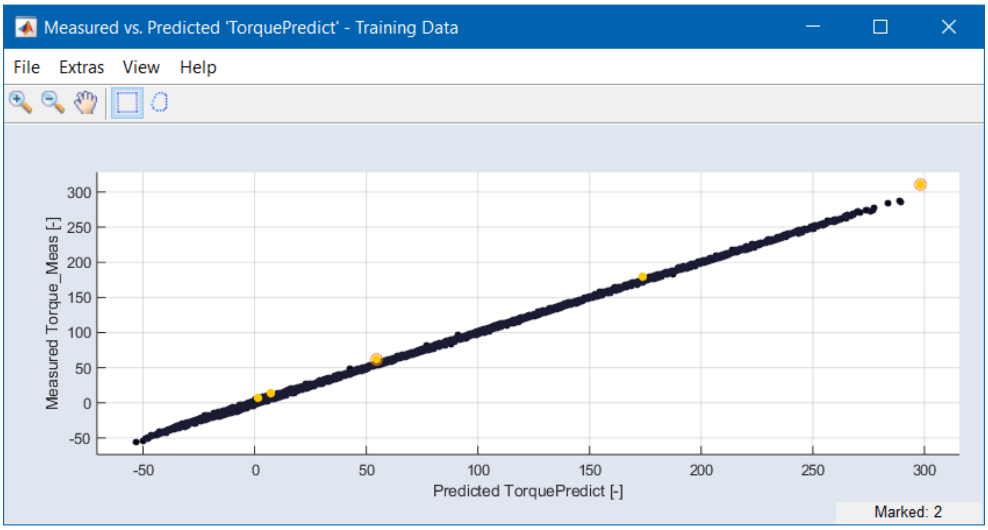


Fig. 4-5: The "Measured vs. Predicted" window

The "Residuals over *" and "Measured vs. Predicted" windows are described in detail in the online help.

Improving the Model Quality

Outliers can be caused by measurement errors or by insufficient function quality. The scatter plots mentioned in sections "Graphical Analysis of Data and Function Nodes" on page 21 and "Residual Analysis" on page 22 allow visually determining and improving the model quality. You can search for outliers, draw a rectangle to mark them, delete them, deactivate them or reduce their weight manually, or you can set an outlier threshold and detect outliers automatically.

4.2.2 Variables RMSE and R^2

A series of variables is used for quantifying the function quality. These variables are described in this section.

4.2.2.1 RMSE (Root Mean Squared Error)

The RMSE describes the variance to be expected (standard deviation) about the model: A second measurement falls less than 1 RMSE from the model prediction with a probability of 68% (with 95.5% < 2 RMSE, 99.7% < 3 RMSE, etc.).

The RMSE is defined as follows:

$$RMSE = \sqrt{\frac{SSR}{N}}$$

Equ. 4-1: Root Mean Squared Error (RMSE)

whereby N = the number of measuring data and

$$SSR = \sum_{i=1}^{N} \left(X_{i,predicted} - X_{i,measured}\right)^2$$

Equ. 4-2: Sum of Squared Residuals (SSR)

Therefore, SSR is the sum of squared residuals (SSR = **S**um of **S**quared **R**esiduals).

4.2.2.2 Coefficient of Determination R^2

The coefficient of determination R^2 is derived from the comparison of the variance that remains after the model training (SSR) with the variance concerning the mean value of all measuring data (SST)

$$R^2 = 1 - \frac{SSR}{SST}$$

Equ. 4-3: Coefficient of determination R^2 whereby

$$SST = \sum_{i=1}^{N} \left(X_{i,measured} - \bar{X}_{i,mean}\right)^{2}$$

Equ. 4-4: Total Sum of Squares (SST)

R^2 is a relative measure for evaluating the function output error – it indicates which portion of the total variance of the measuring data is described by the function.

4.2.3 Function Evaluation Using RMSE and R^2

Evaluation of R^2

The most important variable is the coefficient of determination R^2 ("Coefficient of Determination R2" on the previous page) . This measure results in the following evaluations:

- The coefficient of determination, R^2, can be maximal 1. In this case, the function prediction fits exactly to each measured value.
- If the function would simply predict the mean of the measured output for any input data, an R^2 of 0 would be the result. A negative R^2 would mean that the prediction is worse than that simple prediction.
- An R^2 of 1 means a perfect fit, every prediction of the function is the same as the measured data. Typically, the measured data has added noise. In this case, an R^2 of 1 means overfitting. You should be interested in a high R^2 with consideration of the noise.
- Keep in mind that different signals can be measured with different quality. There might be signals where an R^2 of 0.6 might already be a good value. In contrast, a model for a different signal can be seen as good only if the R^2 is above 0.99.

Evaluation of RMSE

The absolute error RMSE (see section "RMSE (Root Mean Squared Error)" on the previous page) must be evaluated individually:

- At best, the RMSE can be as good as the experimental repeatability.
- Despite a good R^2, the RMSE can be too low, e. g. in case of a very large variation range of the modeled variable.
- Despite a small R^2, the RMSE can be good enough, e. g. if the modeled variable features only a minor variance over the input parameters of the function.

4.3 Parameters

Modern ECUs contain many map-based physical models[1)] to replace or monitor real sensors, e. g.:

- Engine torque
- Air charge/Air mass
- Exhaust gas temperature
- Fuel system corrections

To provide an optimal prediction quality, these models contain parameters such as maps (see also "Maps" on the next page) and curves (see also "Curves" on page 29) that need to be calibrated using real measurement data (e. g., from test bench or vehicle).

The high number of actuators in modern engines leads to a continuous increase in the complexity and the number of parameters of these functions.

A manual calibration is either very time consuming or even impossible.

ASCMO-MOCA supports the calibration and optimization tasks in an efficient and user-friendly way.

4.3.1 Example

You can find an example of parameter optimization for a sensor in the chapter "Tutorial: Working with ASCMO-MOCA" on page 85. In this tutorial, different maps and curves will be optimized in order to reduce the deviation between the measured values and the model prediction.

4.3.2 Available Types of Parameters

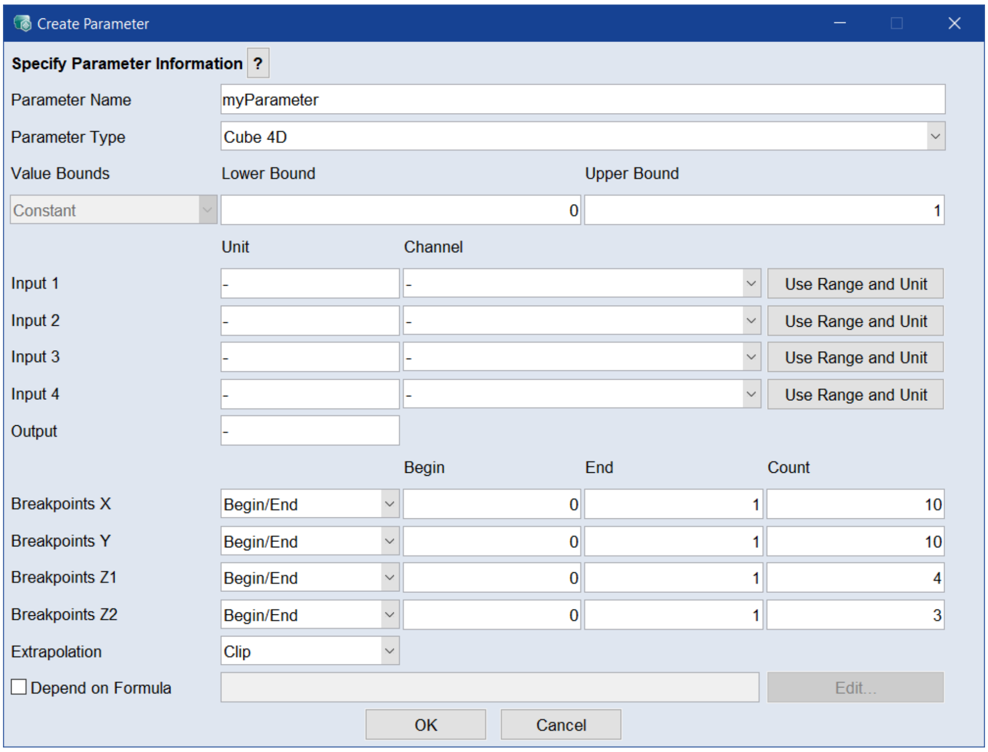
This chapter provides a brief overview of the various types of parameters that can be used in the function (see "Step 5: Build Up the Function" on page 115) for optimization (see "Step 6: Optimization" on page 122).

The parameters are divided in to the following classes:

- "Maps" on the next page
- "Curves" on page 29
- "Scalar" on page 30
- "3D- and 4D-Cubes" on page 30
- "Compressed Model" on page 30
- "Matrix " on page 31
- "Group Axis" on page 31
- " Text Scalar/Matrix/Curve/Group Axis" on page 31

1) Similar models are used in other environments such as HiL systems.

Scalar, Cube-3D and Cube-4D parameters are similar to curves and maps, except that they have no, three (X, Y, Z1), or four (X, Y, Z1, Z2) axes. See the online help for an instruction on how to create such a parameter.

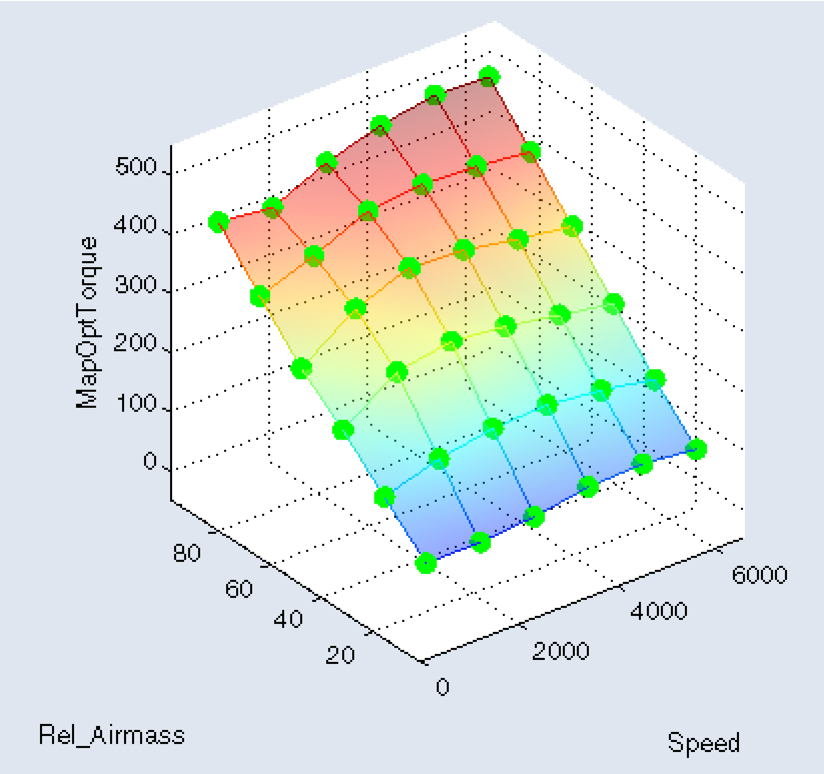


Maps

A map is represented by a set of Z values that are defined over a two-dimensional grid that represents the X and Y axes.

In between grid points, the corresponding Z values are calculated by bi-linear interpolation. Therefore, the functional dependency is given by `z = z(x, y)` and a map is stored in the form of a two-dimensional lookup table.

Outside the grid, either clip- or linear interpolation is applied.



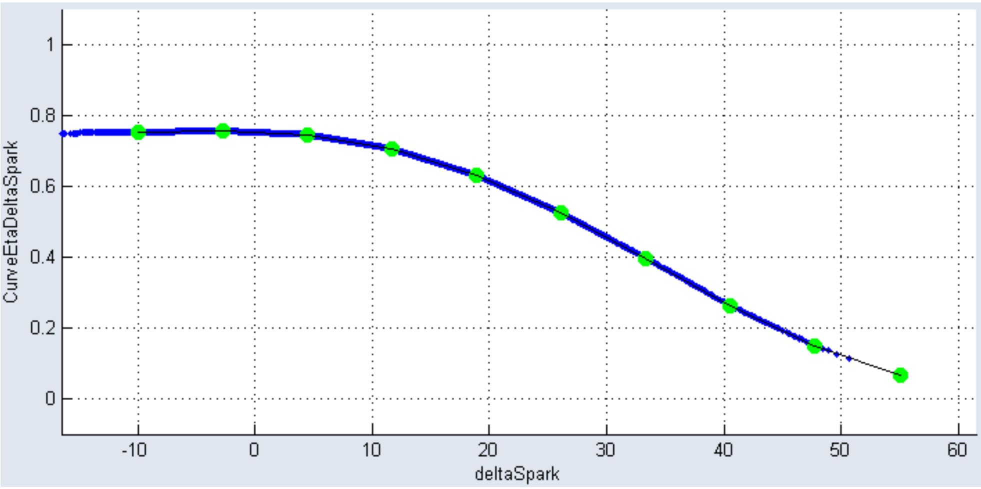
You can set up input-dependent bounds for map parameters. These can be edited in the **Parameter ‹parameter_name›** window; see the online help for more information.

Curves

A curve is represented by a set of Y values that are defined over a one dimensional grid, that represents the X axis.

In between grid points, the corresponding Y values are calculated by linear interpolation. Therefore, the functional dependency is given by `y = y(x)` and a curve is stored in the form of a one-dimensional lookup table.

Outside the grid, either clip- or linear interpolation is applied (cf. figure below).



You can set up input-dependent bounds for curve parameters. These can be edited in the **Parameter <parameter_name>** window; see the online help for more information.

Scalar

A scalar is a 0-dimensional calibration parameter.

3D- and 4D-Cubes

In addition to curves (one input) and maps (two inputs), ASCMO-MOCA supports also lookup tables with three and four inputs: Cube-3D and Cube-4D.

Compressed Model

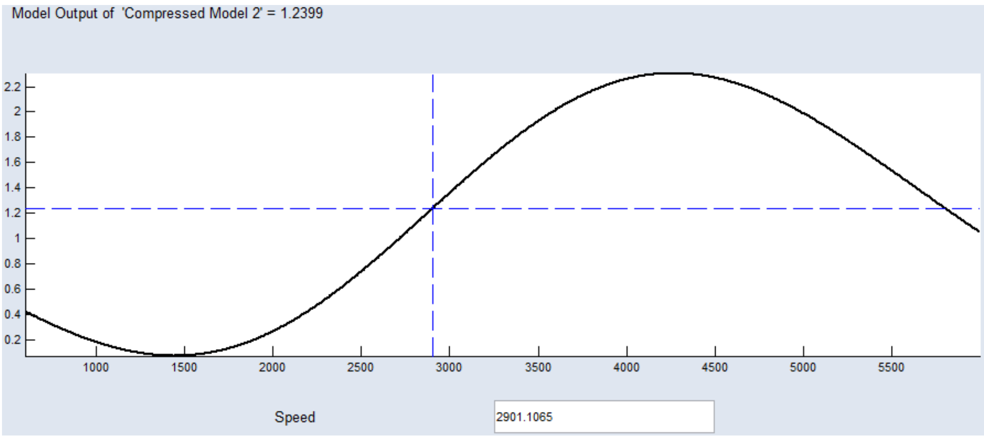
In addition to lookup tables (Curve, Map, Cube), ASCMO-MOCA also supports networks of radial basis functions with a squared exponential kernel (RBF Net-SE) as a parameter.

The number of inputs for such a parameter can be chosen by the user. Also the number of basis functions (kernels) must be chosen by the user. A higher number of inputs and kernels increases the computational complexity of the optimization and evaluation of such a parameter.

The evaluation function for the parameter is a superposition of Gaussian functions. A rough estimate of the computational complexity for the function is "Number of inputs" multiplied with "Number of basis functions" evaluations of the e-function.

It can be seen as a black-box data based model and is also available in ASCMO as "Compressed Model". It can replace a whole function consisting of multiple lookup tables and connections between them.

A higher number of kernels increases the fidelity of the model, but it can result in overfitting and should be tested with test data.



Matrix

ASCMO-MOCA supports matrix parameters. A matrix is a two-dimensional, indexed set of elements. The position of a scalar value within a matrix is determined by its associated index values (non-negative integer values).

Group Axis

ASCMO-MOCA supports group axes for shared axes, which are used by several parameters, e.g. multiple maps share the same axes. They are handled as a separate parameter type. Using group axes ensures, they are consistent. Group axes are especially useful for Simulink® or FMU parameter mappings. For example if Simulink® multiple variable mappings point to the same variable in a Simulink® model or calculated parameters point to the same value reference in an FMU model. Group axes can be exported and imported as DCM or CDFX file and are automatically detected and created using the scan or validate function in the Models Step.
Group axes cannot be used in a function. Group axes can be optimized. Using group axes speeds up the communication with external models.

Text Scalar/Matrix/Curve/Group Axis

Note

Text parameters can only be imported using a DCM and an A2L file; they cannot be created or edited. To import them, click **Import**, select the DCM file, and click **Open**. Then, click **Load A2L File** to select the corresponding A2L file.

Text Scalar: For text scalars, you can select the corresponding label in the **Enumeration** column drop-down. This label consists of the text part and its corresponding actual value in the control unit.

Text Matrix: Text matrices are similar to text scalars, but they have multiple cells with drop-downs where you can select the corresponding labels. These labels consist of the text and their respective actual values in the control unit.

Text Curve: For text curves, the **Label** column represents the X-axis and contains both the text part and its corresponding actual value in the control unit. In the **Value** column, you can find the corresponding Y-values. To edit a value, double-click the respective cell.

Text Group Axis: This is a group axis for multiple curves, where the X-axis is displayed as text in the **Label** column. The corresponding actual value in the control unit is shown in the **Index** column. The **#** column represents a continuous count.

4.3.3 System Constants

System constants can be used to provide default values for parameters. One or more parameters of any type can be assigned to a system constant, and a default value can be provided for each parameter. For non-scalar parameters, the same constant value is returned for each point.

By activating a system constant, you define that the default values of the assigned parameters are used.

System constants are created and managed in the "System Constant" tab of the Parameters Step.

See the online help for an instruction how to create a system constant.

4.3.4 Parametersets

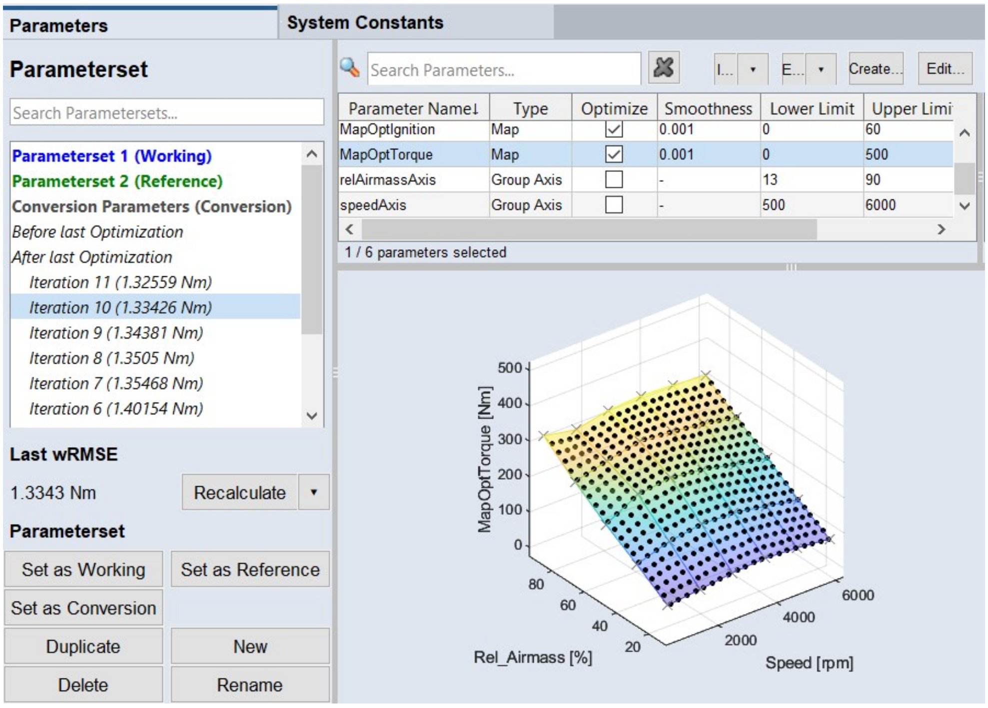
Multiple parametersets can be used and managed whereby one set is always defined as working and reference parameterset. The working parameterset is the one, which is optimized and used in the Optimization Step.

The different parametersets can have different parameters, parameters with the same name can have different support vectors.

In addition to working and reference set, the following parametersets are created while optimization:

- Before last optimization (might be empty, but the set is created and the default Reference Parameterset)
- After last optimization
- During all the iterations, i. e. parameterset Iteration 1, Iteration 2, ...

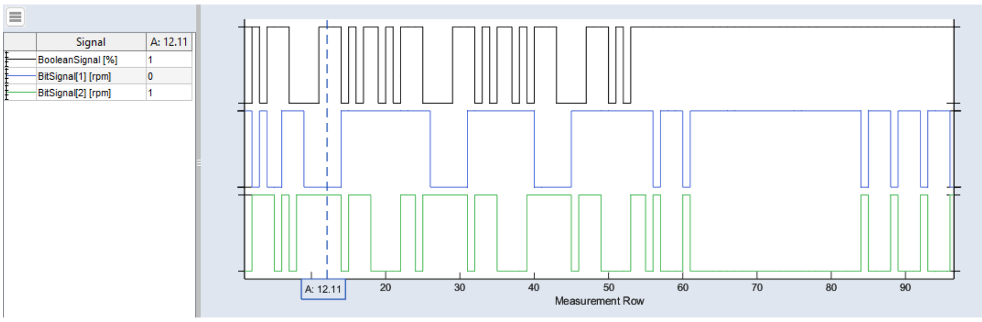
Iterations of the optimizer are automatically stored as separate Parametersets. This allows to analyze the optimization progress and also go back to a previous set.



4.4 Visualization

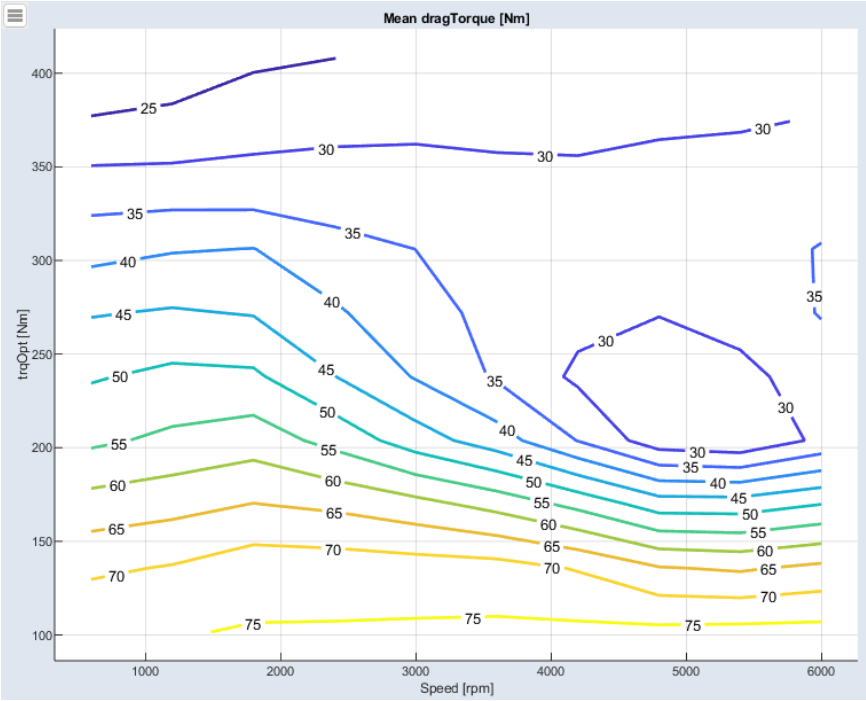
In addition to the performed steps from Data to Optimization, ASCMO-MOCA provides a Visualization Step for having results visually in one place. You can create data and parameter visualizations according to your own taste. The Visualization step allows to combine different plot types to create user defined representations:

Bit Plot to show bit plots where multiple bit signals can be selected and displayed as a scope on top of each other.



Data Contour Plot to show the mean, min, max, median, or data density of the selected data, using lines to connect points of equal value.

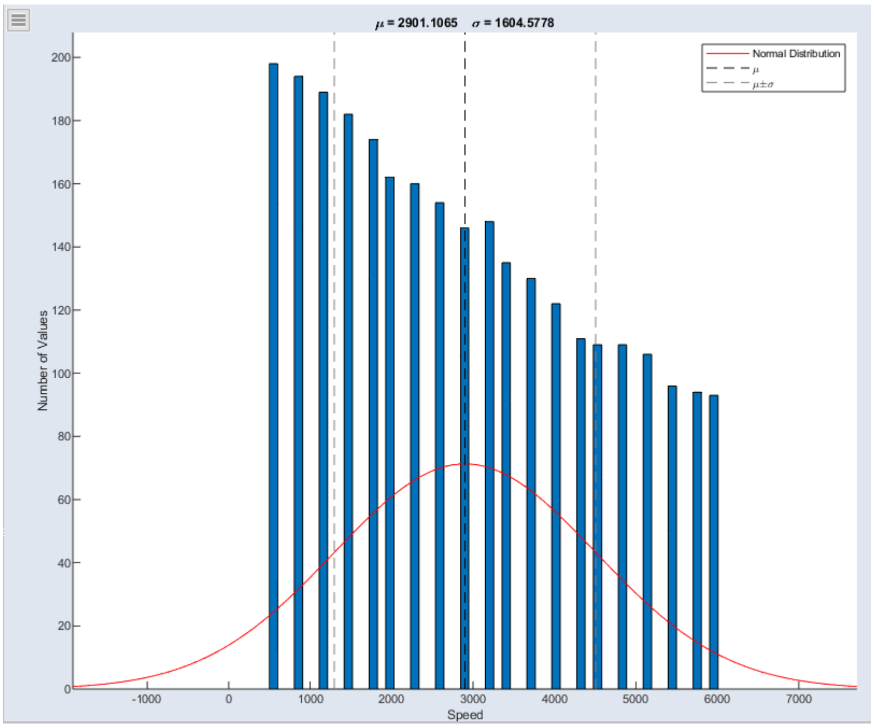
Data points are assigned to overlapping bins/intervals and the metric (mean, min, max) is calculated for data points per interval. The calculated value is then assigned to specific grid values. From the two-dimensional grid values, a map is calculated internally and visualized with contour lines.



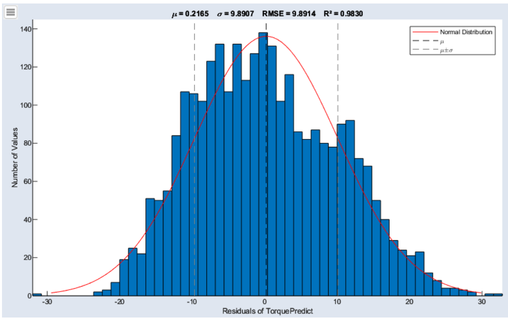
Data Table to show selections from other plots, values are highlighted in the associated color.

	Speed	Rel_Airmass	Ignition	Torque_Meas	TorquePredict
1	597.000	47.8089	-26.8200	7.02201	10.7943
2	597.000	51.6611	-26.8200	19.5251	21.3053
3	597.000	40.1047	-22.2111	-3.67776	-0.166347
4	597.000	43.9568	-22.2111	8.14535	11.3093
5	597.000	47.8089	-22.2111	21.1618	22.9555
6	881.211	47.8089	-22.2111	1.08058	14.6535
7	597.000	51.6611	-22.2111	34.1432	34.6904
8	881.211	51.6611	-22.2111	12.9679	24.6004
9	597.000	55.5132	-22.2111	47.6720	44.8292
10	597.000	59.3653	-22.2111	62.1931	55.6803
11	597.000	32.4005	-17.6021	-14.1937	-15.2673
12	597.000	36.2526	-17.6021	-1.90174	-2.74430
13	881.211	36.2526	-17.6021	-18.1929	-6.92899
14	597.000	40.1047	-17.6021	10.9310	9.98072
15	881.211	40.1047	-17.6021	-5.53035	4.35335
16	597.000	43.9568	-17.6021	23.8206	22.9078
17	881.211	43.9568	-17.6021	6.26536	15.5937
18	597.000	47.8089	-17.6021	38.2851	36.0368
19	881.211	47.8089	-17.6021	19.5687	26.7921
20	1165.42	47.8089	-17.6021	0.470531	18.0611

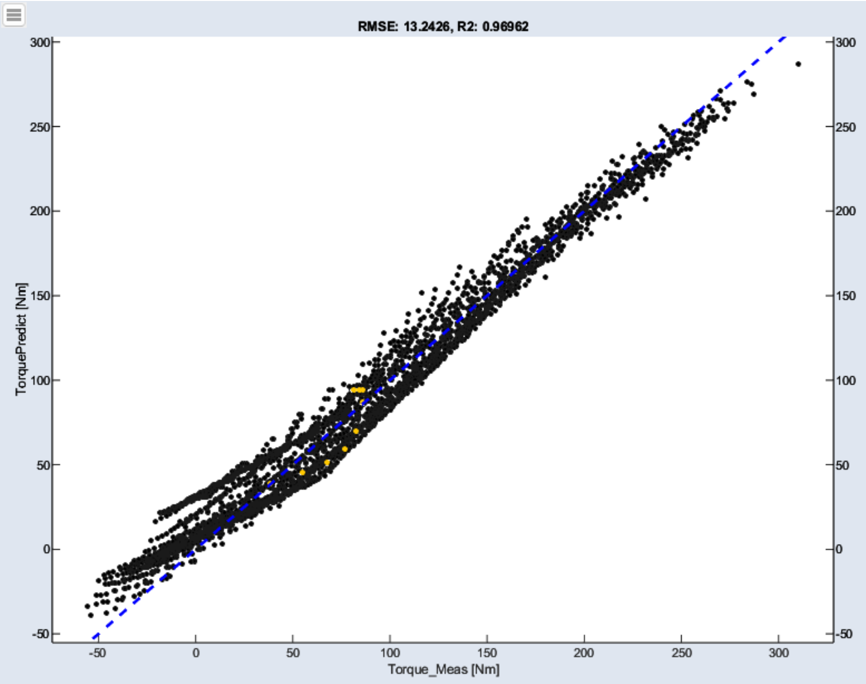
Histogram to show a bar plot where the values are assigned to bins



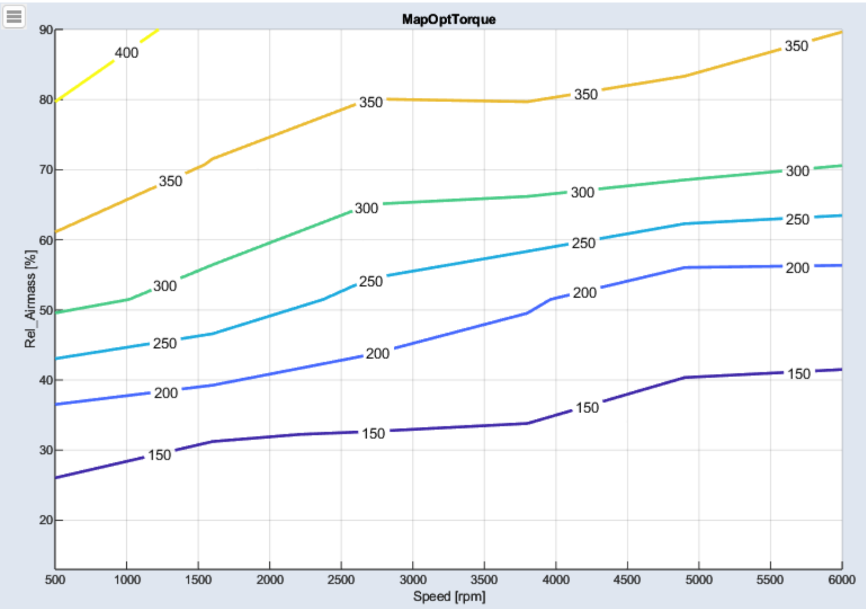
Residual Histogram to the distribution of residuals (differences between observed and fitted values), organized into bins.



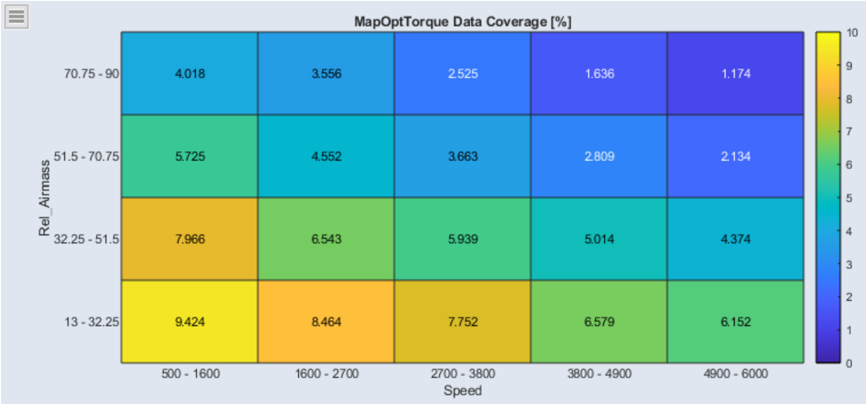
Measured vs. Predicted to show a comparison of measured and predicted signals. Plots include a bisector, RMSE and R^2.



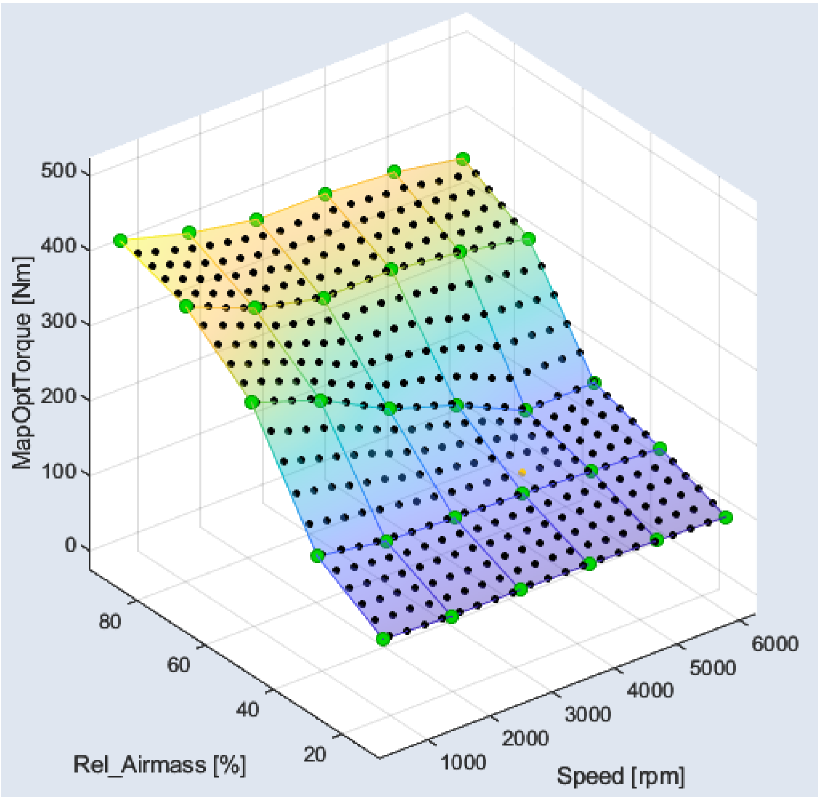
Parameter Contour Plot to show a graphical representation of maps and curves on a two-dimensional plane, using lines to connect points of equal value.



Parameter Heatmap to show the data coverage of a parameter.



Parameter Plot to show parameters such as maps and curves.

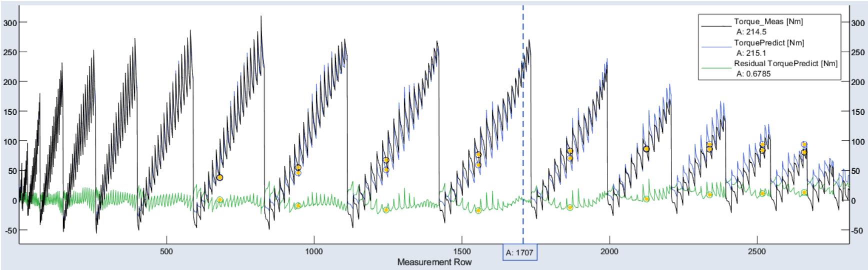


Parameter Table to show the value table for each parameter.

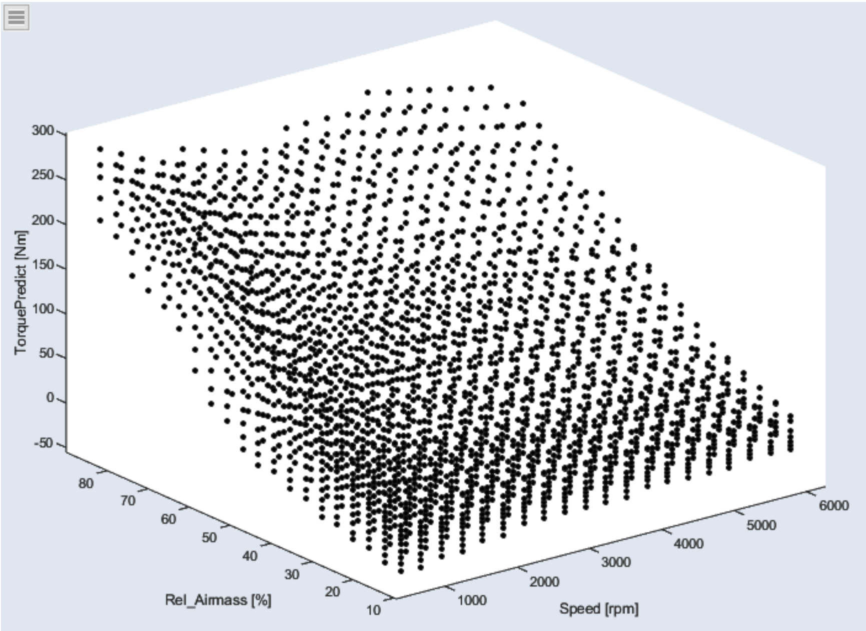
MapDragTorque (Calibration)

Y \ X	500	1600	2700	3800	4900	6000
13	77.0804	78.6306	78.7216	78.9457	79.6010	80.1531
32.25	57.2164	66.5855	68.2530	69.1787	71.0912	72.4950
51.5	34.8366	48.8347	39.5829	30.8083	15.6306	32.8890
70.75	22.7061	30.7942	35.3824	33.8130	35.0545	37.8163
90	16.5684	21.8265	25.3030	25.1882	26.5461	29.5359

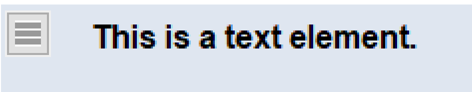
Signal Plot to show data as scatter and scope plots.



Signal Plot 3D to show data as 3D plots.



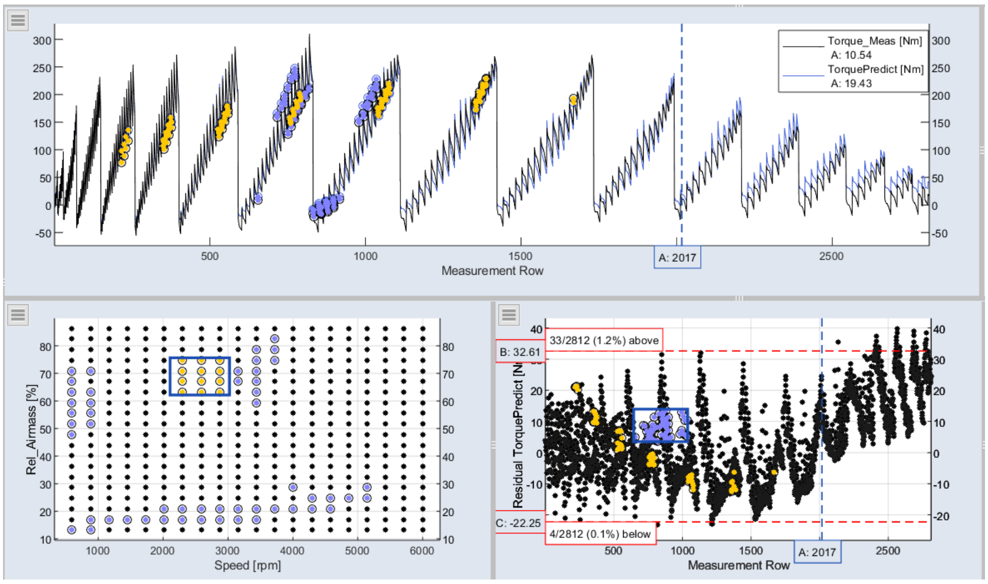
Text to show additional text-based information.



For all types showing imported or calculated data, one or more datasets can be selected to be visualized. Beside this also additional settings can be configured to show only data points, also the lines between the points or a grid in background.

For the types showing parameters, beside the selection of the parameter itself additional content can be chosen such as the reference values or bounds. Parameter plots cannot only show the current calibration, they can also be used to change it. Through this interaction, the Visualization Step acts like an experimental environment. The effect of changed values is directly visualized in all other plots.

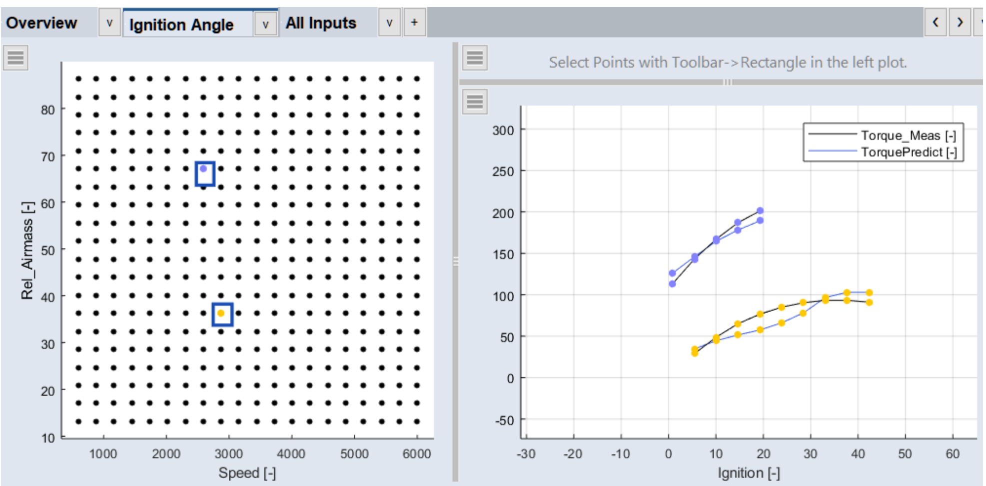
You can define the layout of multiple views. Selections (rectangles and lassoes) can be used in signalplots and highlight the data in any view. Cursors can show the value at a specific time. Cursors can be set as a limit to show the number of values outside the limit. Selections and cursors are saved with the project.



For each visualization tab the screen can be divided into different elements and separately filled. Therefore two modes are available at the bottom: View and Configuration. The View mode also provides the possibility to use a print mode and to export or copy images to the clipboard. Beside visualizing plots you can also insert text into the elements. The visualization tabs can be undocked and arranged on the monitor according to your wishes. Tabs can be easily renamed by double-clicking on it.

The Visualization step supports you in data comparison and to have your data visualized at a glance in one overview.

The **Only Marked Data** option (› **Configure Single Element** › **Only Marked Data**) gives you a better overview, e.g. prediction vs. ignition angle per operating point.



4.5 Models

In ASCMO-MOCA, you can work with models provided as a set of formulas, or you can import models created with ASCET, FMU, Simulink, ASCMO-STATIC or ASCMO-DYNAMIC. These models can then be used as function nodes in the ASCMO-MOCA project.

Importing and connecting external models is done in the Models Step.

- ASCET models

  If you want to use an ASCET model in ASCMO-MOCA, you have to create a `*.dll` file with ASCET and ASCET-PSL first. This `*.dll` file is then added to the ASCMO-MOCA project; see the online help for details.

  ASCET models are used as black boxes by ASCMO-MOCA. You cannot change the models, and no link to the ASCET model or to ASCET is created during import.

- FMU models

  If you want to use an FMU model in ASCMO-MOCA, you have to create an FMI `*.fmu` file first. This `*.fmu` file is then added to the ASCMO-MOCA project; see the online help for details.

  > **Note**
  >
  > FMU models that use FMI 2.x or FMI 3.x are supported by ASCMO-MOCA. FMU models that use FMI 1 cannot be used.

  Only the FMU file name is added to the ASCMO-MOCA project. You cannot open the model itself. During optimization, the FMU model is used as a black box: ASCMO-MOCA passes the inputs to the model, and receives the outputs from the model. The way the model computes the output values remains unknown to ASCMO-MOCA.

  The execution of Linux FMUs (i.e. no Win32 or Win64 binaries included in the FMU) is supported, an appropriate Linux image must exist. An FMU with Linux binaries can be run, if WSL2 (Windows Subsystem Linux) is installed. The virtual machine needs ZeroC Ice and libgomp.

  For example on Debian this can be installed by

  ```
  sudo apt install libgomp1
  sudo apt install libzeroc-ice3.7 libzeroc-ice zeroc-ice-compilers zeroc-ice-slice
  ```

  Take care that at least Ice version 3.7.6 is installed.

- Simulink® models

  Using Simulink® models in ASCMO-MOCA is described in detail in the tutorial, see "Step 4: Models" on page 105.

- ASCMO-STATIC and ASCMO-DYNAMIC models

These models are used as black boxes. After import of the models, the ASCMO project isn't linked anymore and the models become part of the ASCMO-MOCA project.

During import, you can select one, several or all outputs for import. Each output is added as a separate model. See also section "Importing ASCMO-STATIC/ASCMO-DYNAMIC Models" in the online help.

- TSiM Plugin

  If you want to use a TSiM Plugin in ASCMO-MOCA, you need a `*.mexw64` file. This proprietary file format of Bosch is similar to FMU and typically represents control unit functions in a compiled form (DLL).

  This format can be used in ASCMO-MOCA for simulations and optimization of parameters.

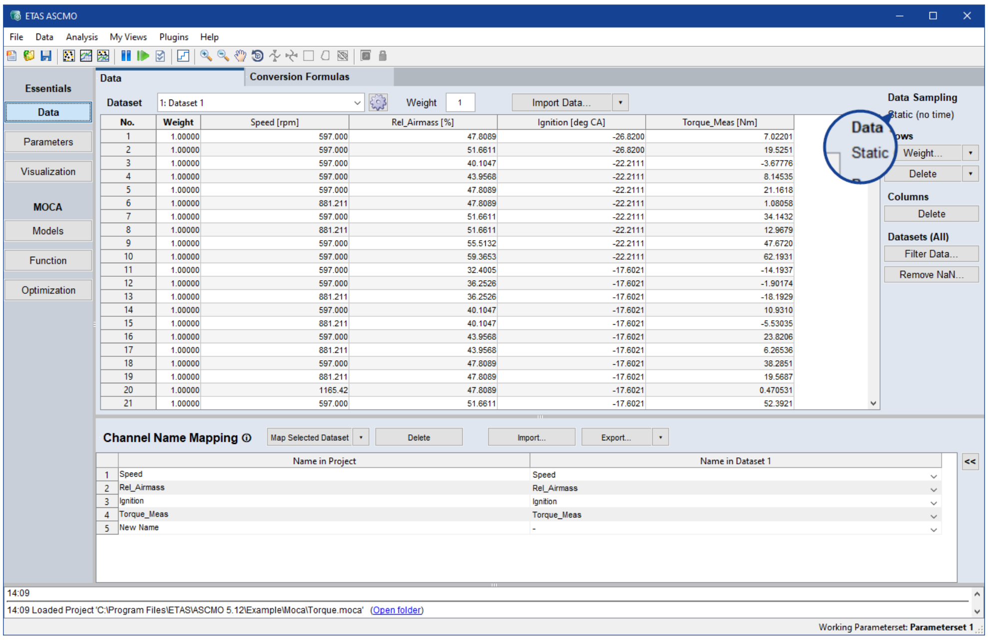
For more information, see the online help (F1).

4.5.1 Steady State

Steady State is a concept used in the Models Step in ASCMO-MOCA for following model types:

- FMU models
- Simulink models
- TSiM PlugIn

It can only be applied if the imported data is static. This can be checked in the Data Step (**Data Sampling: Static (no time)**).



Steady state is a state in which all relevant variables are constant relative to each other over time or grow at the same rate (steady development). That means the state of equilibrium. 3 sizes can be specified.

- **Simulation Step Size**: Define the simulation step size (base sample time). This is specified in the model and must match.
- **Time until Steady State**: This is the time span to wait until steady state is reached (worst case scenario).
- **Average Last**: Defining the duration of the average interval at the end of **Time until Steady State** phase. Enter time in seconds to average the last values and determine a mean value.

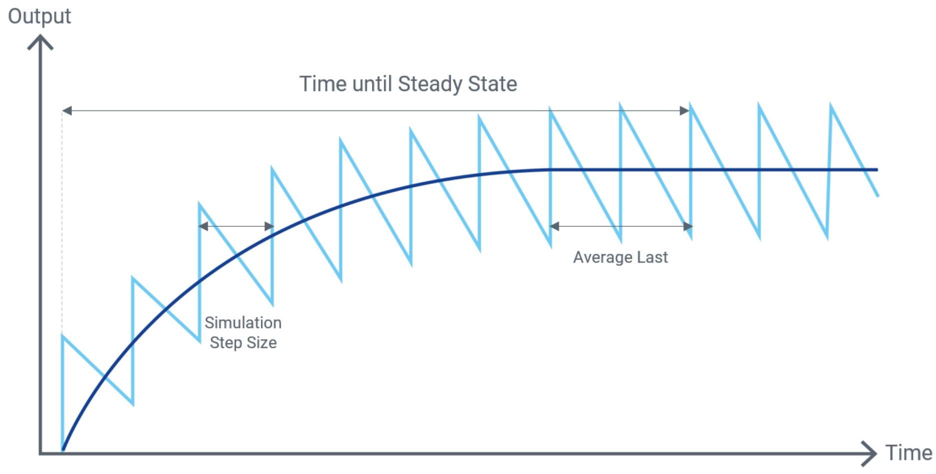


Fig. 4-6: Steady state visualization: the dark blue line is the average of the light blue line

4.6 Function

In ASCMO-MOCA, you can work with models provided as a set of formulas, or you can import models created with Simulink, ASCET, ASCMO-STATIC or ASCMO-DYNAMIC and connect them to the ASCMO-MOCA project.

Specifying a function formed by a set of formulas is done in the Function pane.

Data channels, parameters, other function nodes and imported models can be used to define the expression of a function node. Several operators are available; see " Mathematical Operators for Function Nodes" below.

You can export and import functions to and from text files that follow the formula syntax. A sample export file is given here.

```
trqOpt[-] = %MapOptTorque%(%Speed%,%Rel_Airmass%)
ignOpt[-] = %MapOptIgnition%(%Speed%,%Rel_Airmass%)
deltaSpark[-] = %ignOpt% - %Ignition%
etaSpark[-] = %CurveEtaDeltaSpark%(%deltaSpark%)
product[-] = %SubFunction%(%deltaSpark%, %trqOpt%,
%CurveEtaDeltaSpark%)
dragTorque[-] = %MapDragTorque%(%Speed%,%Rel_Airmass%)
TorquePredict[-] = %product% - %dragTorque%

function SubFunction(InDeltaSpark : Data, IntrqOpt : Data,
myCurve : Curve)
curveOut[-] = %myCurve%(%InDeltaSpark%)
functionOut[-] = %curveOut% .* %IntrqOpt%
```

For more information, see the following subsections and the online help.

- " Mathematical Operators for Function Nodes" below
- "Step 5: Build Up the Function" on page 115 (tutorial)

4.6.1 Mathematical Operators for Function Nodes

Function nodes can be added and edited in the Insert/Edit Node window. At the right side of that window, you can see buttons for common mathematical operators.

× / + -	Inserts the `*`, `/`, `+`, or `-` operator into the expression. This results in element-wise multiplication, division, addition, or subtraction.
√	Inserts the square root `sqrt(` operator into the expression. The expression must end with `)`..
x^a	Inserts the `^` operator into the expression. x^y means row-by-row **x** to the power of **y**.
abs	Inserts the absolute value `abs(` operator into the expression. The expression must end with `)`. Example[1]

bswitch	Inserts the `bswitch(` operator into the expression. The expression must end with `)`.
	Binary switch, element-wise:
	`y = bswitch (x, a, b)`
	`y = a for x <= 0`
	`y = b for x > 0`
	Example[2]

( )	Inserts an opening or closing bracket into the expression.
,	Inserts a comma into the expression.
min max	Inserts the `min(`/`max(` operator into the expression. The expression must end with `)`.. min = minimum of two inputs max = maximum of two inputs Example[3]
&	Inserts the `&` operator into the expression, which means a logical element-wise AND. Example[4]

1) `abs(-3) ≥ 3`

2) `bswitch (%speed% > 2000, %Y_1%, %Y_2%)`

3) `min(%in1%, %in2%)`, `max(%in1%, %in2%)`

4) `%Speed% > 2000 & %Load% > 6`

<table>
<tr><td>|</td><td>Inserts the | operator into the expression, which means a logical element-wise OR.
Example[1]</td></tr>
<tr><td>~</td><td>Inserts the ~ operator into the expression, which means a logical not.
Example~(%x1% & %x2%)</td></tr>
<tr><td>cumsum</td><td>Inserts the cumsum(operator into the expression, which means the cumulative sum of a numerical sequence. The expression must end with).
Example[2]</td></tr>
<tr><td><
<= == >=
></td><td>Inserts the </<=/==/>=/> operator into the expression, which means is less than/less than or equal to/equal to/greater than or equal to/greater than.</td></tr>
<tr><td>warnIf</td><td>Inserts the warnif(operator into the expression. The expression must end with).
It allows you to define checks that are automatically executed after each optimization run. If the defined condition is met, the specified warning text is displayed in the log window.
To use the warnif operator, you need to insert a condition in the expression field after clicking the button. The syntax for the warnif operator is as follows:
<code>y = warnIf(condition, 'warningText')</code>.
Examples
<code>warnIf(%MapDragTorque%(%Speed%, %Rel_Airmass%) > 0, 'Warning')</code>
In this example, if at least one of the values of the map parameter MapDragTorque is greater than zero after the optimization, the text "Warning" will appear in the log window.
<code>warnIf(%speed% < 0, 'speed less than zero')</code>
In this case, if the value of speed is less than zero after the optimization run, the warning "speed less than zero" will be issued.</td></tr>
</table>

[1] `%Speed% > 2000 | %Load% > 6`

[2] `cumsum(%y%): [1 2 4] ≥ [1 3 7]`

timeDelay	Inserts the **`timeDelay(`** operator into the expression. The expression must end with **`)`**. It allows you to delay a signal by one time step. After clicking the button, you need to insert a condition in the expression field using the following syntax: **`timeDelay(x, initialValue)`** In this expression, initialValue is returned in the first step, and it is the only way to access nodes further in the function. Examples **`timeDelay(%transferFcn%, %setpoint%(1))`** In this example, the signal from **`transferFcn`** is delayed by one time step, with the initial value set to the first element of the input **`setpoint`**. **`y = timeDelay(%y%, 0.0)`** Here, the value of **`y`** is delayed by one time step, with an initial value of 0.0.
dT	Inserts the delta T (sample time, **`dT`**) operator into the expression. It represents the sample time in the data and will be replaced with the corresponding value during execution. Examples **`dT ./ %filterConstant%`** In this example, **`dT`** is the value from the **time [s]** column in the **Data** step, while **`%filterConstant%`** is a parameter used in the calculation. **`y = timeDelay(%y%, 0.0) * dT`** In this case, the value of y is multiplied by the sample time dT after applying a time delay.

roundTo	Inserts the **`roundToDiscreteValues(`** operator into the expression, which rounds a node to a specific discrete value. The expression must end with **`)`**. After inserting the operator, you need to select a discrete parameter and enter the desired values. The syntax is as follows: **`roundToDiscreteValues(node, [value1, value2, ...])`** Examples **`roundToDiscreteValues(%calMap_SCV%(%speed%, %load%), [0, 1])`** In this example, the output of the calibration map **`calMap_SCV`** is rounded to the discrete values 0 and 1 based on the inputs **`speed`** and **`load`**. **`roundToDiscreteValues(%myTernary%, [0, 1, 2])`** Here, the value of **`myTernary`** is rounded to the discrete values 0, 1, and 2.
steady States	Inserts the **`steadyState_abs(`** operator into the expression, which calculates the steady state of a signal based on the window length and window height. The expression must end with **`)`**. The syntax is as follows: **`steadyState_abs(x, windowLength, windowHeight, sampleRate)`** Example **`steadyState_abs(%signal%, 10, 200, dT)`** In this example, the steady state of the **`signal`** is calculated using a window length of 10, a window height of 200, and the sample rate represented by **`dT`**.
←	Deletes the last entry in the expression field (backspace).

The following operators are supported. Select them from the burger menu (☰) or type them in manually:

`log(x)`	The natural logarithm (base e). Example[1]
`log10(x)`	The common logarithm (base 10). Example[2]

1) **`log(exp(2)) => 2`**

2) **`log10(10^2) => 2`**

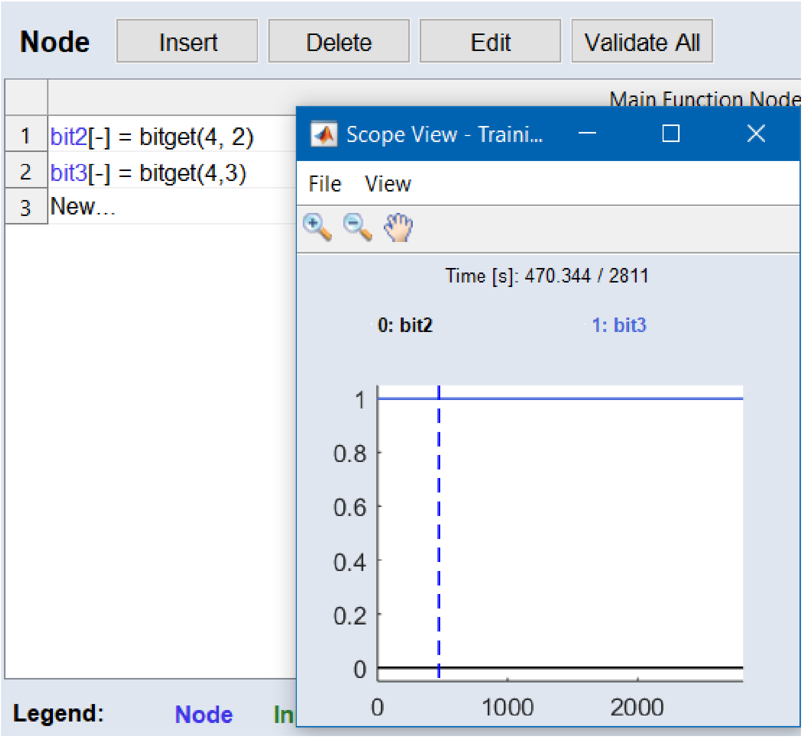
`exp(x)`	Euler's number raised to the power of e^x. Example[1)]
`sin(x), cos(x), tan(x), tanh(x), atan(x)`	The trigonometric functions, the input is in radians. Example[2)]
`atan2(x, y)`	The four-quadrant inverse tangent, input is given in radians. Example[3)]
`delayseq(data, n)`	Delay a signal by n time steps. The start of the signal is filled with zeros. Example[4)]

1) `exp(1)) => 2.718`

2) `sin(3.1416) ~> 0`

3) `atan2(%id%, %iq%)`

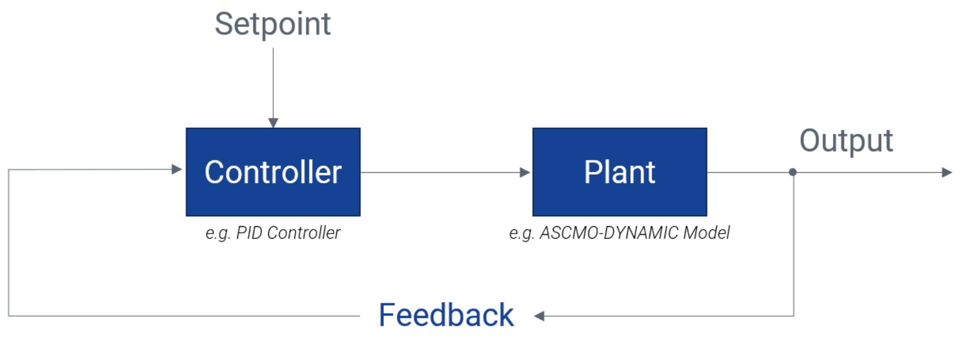
4) `delayseq([1; 2; 3; 4;], 2) => [0; 0; 1; 2]`

`bitget(x, bitPos)`	The MATLAB® **`bitget`** function returns a bit value at a specified position (**`bitPos`**). Example **`bitget(4, 3)`** returns **`1`** **`bitget(4, 2)`** returns **`0`** In this example, the integer 4 is represented in binary as 100. The function bitget(4, 3) retrieves the bit at position 3, which is 1, while bitget(4, 2) retrieves the bit at position 2, which is 0. 

Further notations are supported:	
`z = [x, y]`	Creates a multidimensional vector. A vector can be extracted with **`y = z(:, 2)`**. This can be useful, e.g., when a sub-function shall returns multiple nodes/vectors.
`y = zeros(size(x))`	Creates a vector of zeros.
`y = ones(size(x))`	Creates a vector of ones.

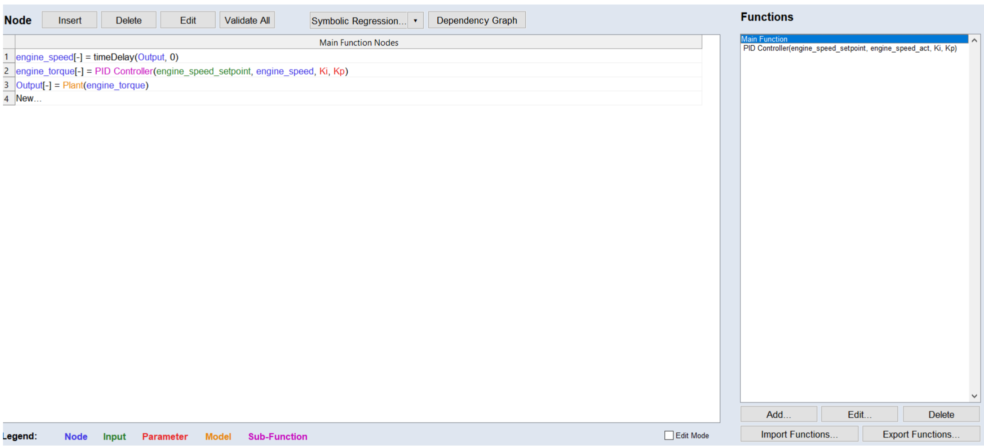
4.6.2 Feedback Loop

Within a node of the Function you can access a future node using a feedback loop with time delay. This can be also used with a dynamic model. See the following graphic and ASCMO-MOCA example.



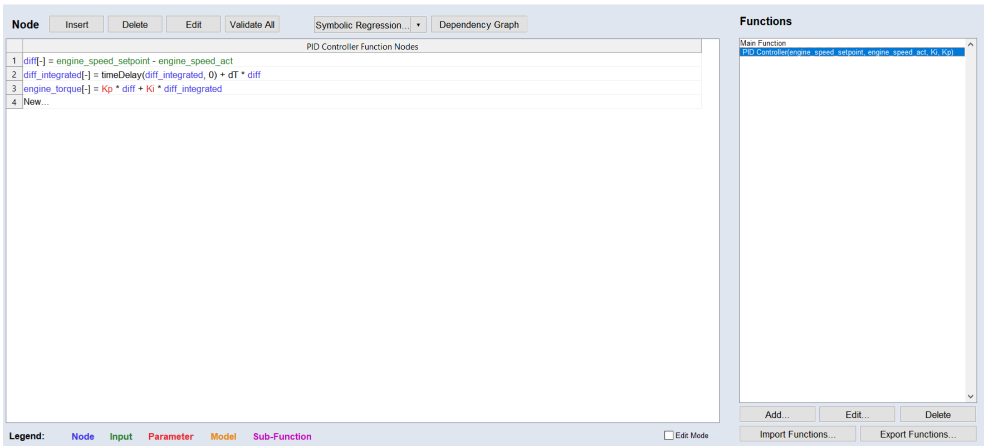
Main Function Nodes

```
engine_speed[-] = timeDelay(%Output%, 0)
engine_torque[-] = %PID Controller%(%engine_speed_setpoint%,
 %engine_speed%, %Ki%, %Kp%)
Output[-] = %Plant%(%engine_torque%)
```

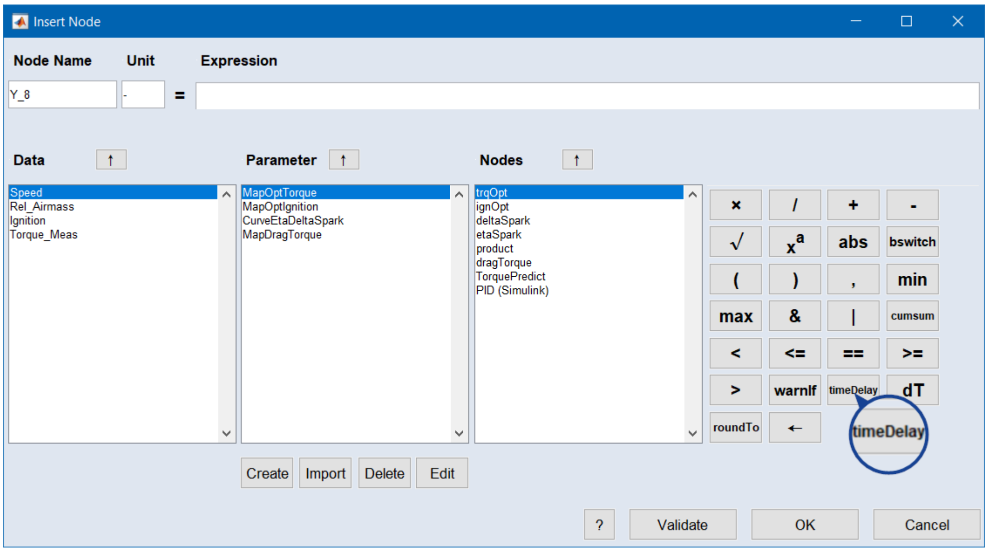


PID Controller Function Nodes

```
diff[-] = %engine_speed_setpoint% - %engine_speed_act%
diff_integrated[-] = timeDelay(%diff_integrated%, 0) + dT .*
 %diff%
engine_torque[-] = %Kp% .* %diff% + %Ki% .* %diff_integrated%
```



The `timeDelay` element is one of the mathematical operators in the "Insert/Edit Node" window.



4.7 Optimization

This section contains a description of the different optimization methods and the optimization criteria that can be used for the parameter optimization.

This section contains the following subsections:

- "Description of the Optimization Method" below
- "Consideration of the Roughness" on page 65
- "Optimization Criterion" on page 66
- "Optimization Without Sequence" on page 67
- "Optimization With a Sequence" on page 67
- "Parameter Correlation" on page 68
- "Parameter Sensitivity" on page 68

4.7.1 Description of the Optimization Method

The optimizer calibrates the *p* calibration values of the maps/curves with the goal to minimize the deviation between the measured, predetermined values and the predicted *n* values.

$$\underset{p}{\operatorname{argmin}} \sum_{i=1}^{n} \left(W_o * \left(Y_{i,predicted}(p) - Y_{i,measured}(p) \right)^2 + \cdots \right)$$

Equ. 4-5: Optimization method

where

p	calibration values
n	number of measurement points
$Y_{predicted}$	prediction of the function in ASCMO-MOCA/ASCMO-MOCA Runtime
$Y_{measured}$	the imported data
W_o	Weight of Optimization
W_c	Weight of Constraint
W_g	Weight of Gradient, 1..D dimensions
W_k	Smoothness factors, 1..D dimensions

The squared deviation is minimized, where the square has the effect that larger deviations are penalized even stronger.

Based on this general formula, smoothness, local constraints and gradient limits can be added. This can be expressed in the following formulas.

Smoothness

$$\underset{p}{\operatorname{argmin}} \sum_{i=1}^{n} \left(W_o * \left(Y_{i,predicted}(p) - Y_{i,measured}(p) \right)^2 + \sum_{k=1}^{D} \boldsymbol{W_k} * \boldsymbol{Roughness} + \cdots \right)$$

See 4.7.5 "Optimization Criterion" on page 66

Gradient Limits

$$\underset{p}{\operatorname{argmin}} \sum_{i=1}^{n} \left(W_o * \left(Y_{i,predicted}(p) - Y_{i,measured}(p) \right)^2 + \sum_{g=1}^{D} \boldsymbol{W_g} * \boldsymbol{Gradients} + \cdots \right)$$

See "Optimization Criterion" on page 66

Local Constraints

$$\underset{p}{\operatorname{argmin}} \sum_{i=1}^{n} \left(W_o * \left(Y_{i,predicted}(p) - Y_{i,measured}(p) \right)^2 + \boldsymbol{W_c} * \boldsymbol{Constraint} + \cdots \right)$$

4.7.2 Optimization Algorithms

In the **Optimization** step, you can choose from the following optimization algorithms:

- "Default (Gradient Descent)" on the next page
- "Respect Constraints (Gradient Descent)" on page 54
- "Gradient-free Optimizer" on page 54
- "Surrogate Optimizer (Global Optimization)" on page 55

- "Genetic Algorithm (Global Optimization)" on page 55
- "Simulated Annealing (Global Optimization)" on page 55
- "Particle Swarm (Global Optimization)" on page 56

Gradient Descent vs. Global Optimization

Gradient descent optimization starts with the working parameter set as initial values and uses the gradient of the cost function to iteratively follow the direction of the steepest descent to the minimum of the cost function. The gradient descent optimizer only finds a local minimum, so the starting position of the optimization is important.

ASCMO-MOCA calculates the gradients of the function analytically. Gradients from external models, such as Simulink or FMU models, are computed using the finite difference method. This allows the optimizer to find the local minimum quickly and with a minimum number of function evaluations. The memory consumption is at least the number of data points multiplied by the number of parameters and multiplied by two for the optimization algorithm itself.

Global optimizers are gradient-free optimizers that try to find a global optimum. They start with several random candidate solutions to the optimization problem spread over the entire search space. Then the search space is iteratively narrowed down to good, more accurate solutions. The optimization may not hit the optimum perfectly, so you can start with a global optimization to find the global optimum, and then continue with a gradient descent optimization to refine the result.

A typical optimization problem in ASCMO-MOCA is the optimization of maps and curves. Such an optimization problem usually has many parameters (e.g., a 20x20 map has 400 parameters), and a global optimizer may require many iterations to find a good solution.

Being gradient-free, global optimization can find a solution when the gradients of a function/model are not continuous. This happens when your model is implemented with a fixed-point representation of numbers or parameters and inputs or outputs are discrete. It can also happen if your model is implemented with 32-bit instead of 64-bit floating point numbers.

Default (Gradient Descent)

This is a gradient descent least squares optimizer. It was chosen as the default optimizer because it performs well when the optimization task has many parameters, which is likely when using maps and curves.

The residuals enter the optimization algorithm as a vector, so the optimizer gets 100 residuals for a dataset with 100 data points. This is computationally expensive, but leads to good optimization results.

The residuals are implicitly squared by the optimizer, so the difference from a reference value is always minimized. To do a minimization or maximization, you must explicitly provide a low/high value to optimize against.

EXAMPLE
The output is in the range 0 to 1000. To do a maximization, define the optimization criterion as **minarg(y(x)-1000)**.

Local constraints enter the optimization as part of the sum formula (`W * Constraint`). In the context of ASCMO, this is called a soft constraint. The weight of such a local constraint is increased every 10 iterations until the constraint is satisfied.

$$\underset{p}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\left(Y_{i,predicted} - Y_{i,measured} \right)^2 + W * Constraint \right)$$

During optimization, the weight of the constraint is increased if the constraint is violated. This may not be sufficient, and the constraint may still be violated after optimization.

When to use: The Gradient Descent Optimization is ideal for the typical problems MOCA addresses. It is fast and delivers good results, though it requires substantial memory. Other optimization algorithms should be considered only in specific situations outlined below.

Usage: Gradient descent optimization is ideal for the typical problems that MOCA addresses. It is fast and gives good results, although it requires a lot of memory. Other optimization algorithms should be considered only in the specific situations described below.

Respect Constraints (Gradient Descent)

This is also a gradient descent least squares optimizer, but it takes constraints into account.

The constraints are treated by the optimizer as hard bounds. The default optimizer should be preferred unless the constraints are violated. Use this algorithm if a constraint is violated after optimizing with the default optimizer.

Usage: This optimization algorithm is useful when local constraints are violated after running an optimization. It's recommended to first try the default optimizer and increase the weight of the local constraints (e.g. 10, 100, 1000, etc.). If this approach fails, this algorithm may find a solution within the constraints.

Gradient-free Optimizer

The gradient-free Optimizer uses a simplex algorithm for optimization. The algorithm does not depend on gradients and therefore requires more iterations than a gradient descent algorithm. The solutions are typically not as good as those of a gradient descent optimizer.

Use this gradient-free optimizer when the gradients of the function/model are not continuous. This happens when your model is implemented with a fixed-point representation of numbers or parameters and signals are discrete. It can also happen if your model is implemented with 32-bit instead of 64-bit floating point numbers.

Usage: This optimization algorithm is suitable for external models that are insensitive to small parameter changes, such as those that use fixed-point representation internally. If an external model uses single precision (32-bit) instead of double precision, consider increasing the finite difference factor to 20,000 and using the default optimizer.

Surrogate Optimizer (Global Optimization)

The Surrogate Optimizer tries to find a global optimum. It first builds a surrogate model and then optimizes it instead of the original function/model. This can be useful if the function/model evaluation takes a long time.

Usage: This optimization algorithm is useful when evaluating an external model is time-consuming. A surrogate model, which is evaluated quickly, is built for optimization purposes.

Genetic Algorithm (Global Optimization)

The Genetic Algorithm tries to find a global optimum. It is inspired by natural selection and the exchange of genomes. It starts with random candidate solutions, here called a population, see Wikipedia: Genetic Algorithmfor more details. The size of the population strongly influences the memory consumption. An optimization task with many signals and/or data can lead to out-of-memory problems. Reducing the population size frees up memory. The algorithm performs a vectorization with all candidate solutions and can perform model evaluations of FMU and TSim models in parallel.

Usage: This optimization algorithm is a gradient-free optimization algorithm suitable for insensitive models. Its primary purpose is to find a global minimum, especially when the default optimizer may get stuck in a local minimum. The algorithm uses less memory than the default optimizer, but may take longer to achieve similar results.

Simulated Annealing (Global Optimization)

Simulated Annealing tries to find a global optimum. It starts with random candidate solutions at the beginning of the optimization. Due to a high initial temperature, large parameter changes are possible. Over several iterations, the temperature decreases, limiting possible parameter changes and allowing more accurate solutions to be found. See Wikipedia: Simulated Annealing for more details. The number of particles strongly affects the memory consumption. An optimization task with many signals and/or data can lead to out-of-memory problems. Reducing the number of particles frees up memory.

Usage: This optimization algorithm is a gradient-free optimization algorithm suitable for insensitive models. Its primary purpose is to find a global minimum, especially when the default optimizer may get stuck in a local minimum. The algorithm uses less memory than the default optimizer, but may take longer to achieve similar results.

Particle Swarm (Global Optimization)

Particle Swarm tries to find a global optimum. It starts with random candidate solutions, called particles. The particles have a position and a velocity. It starts with a high velocity and over several iterations the velocity decreases, allowing more accurate solutions to be found, see Wikipedia: Particle Swarm Optimization for more information. An optimization task with many signals and/or data can lead to out-of-memory problems. Reducing the number of particles frees up memory. The algorithm performs a vectorization with all candidate solutions and can perform model evaluations of FMU and TSim models in parallel.

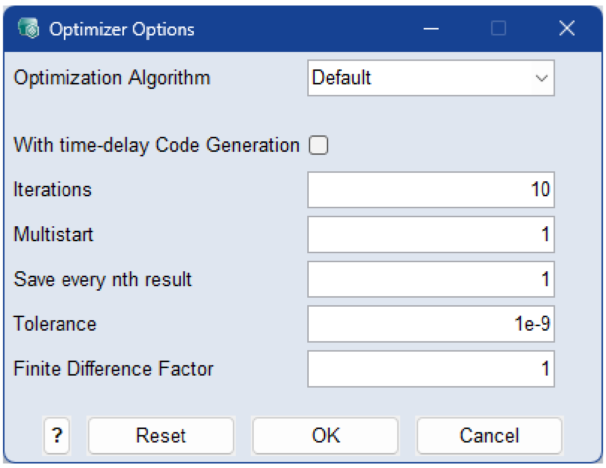
Usage: This optimization algorithm is a gradient-free optimization algorithm suitable for insensitive models. Its primary purpose is to find a global minimum, especially when the default optimizer may get stuck in a local minimum. The algorithm uses less memory than the default optimizer, but may take longer to achieve similar results.

4.7.3 Optimizer Options

Optimization step › **Configure**

From the **Optimization Algorithm** drop-down list, select the optimization algorithm for which you want to customize the settings:

Default



With time-delay Code Generation

When activated, C code is generated from the function and optimization works on compiled C code. This only works if the function does not use external models. This speeds up the optimization when the **timeDelay** method is used in the function, in other cases it's usually slower.

Iterations

Enter the maximum number of iterations to be performed during optimization.

Multistart

Enter the number of times to run the optimizer with different initial values.

If set to a number greater than one, the optimizer will run multiple times. The first optimization is started with the current working parameter set, while subsequent optimizations start with random parameter values. This can be used to find a global optimum.

Save every nth result

Save only every n^{th} iteration as a temporary parameterset. Typical values are in the range [1, 100], default is 1.

Tolerance

The optimization stop criterion. Typical values range from 1e-9 to 1e-16.

The value is used to stop the optimization in the following cases:

- A parameter change by the optimization algorithm would be smaller than this value.
- The change in the cost function is smaller than this value. The optimization has converged.

Finite Difference Factor

The gradients of external models (FMU, ...) are calculated by finite gradients.

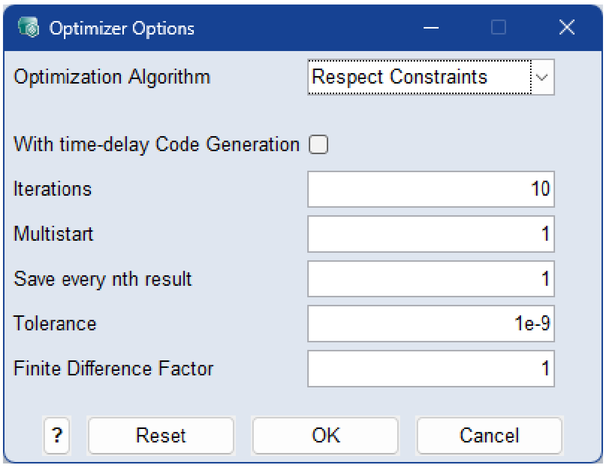
Gradient = **(f(x+e) - f(x)) / e**

with **e = Finite Difference Factor * Normalized Parameter Range * 1.5e-8**

If the output of a model does not change by such a small value, the optimizer stops at the first iteration. In this case, the **Finite Difference Factor** must be increased. Typical values are 10, 100, 1000, 10,000,...

1.5e-8 is the square root of the smallest number that can be shown with a double precision floating point number. If the model uses single precision floating point numbers, set the factor to 10,000 to consider the loss of precision.

Respect Constraints



With time-delay Code Generation

When activated, C code is generated from the function and optimization works on compiled C code. This only works if the function does not use external models. This speeds up the optimization when the **timeDelay** method is used in the function, in other cases it's usually slower.

Iterations

Enter the maximum number of iterations to be performed during optimization.

Multistart

Enter the number of times to run the optimizer with different initial values.

If set to a number greater than one, the optimizer will run multiple times. The first optimization is started with the current working parameter set, while subsequent optimizations start with random parameter values. This can be used to find a global optimum.

Save every nth result

Save only every n^{th} iteration as a temporary parameterset. Typical values are in the range [1, 100], default is 1.

Tolerance

The optimization stop criterion. Typical values range from 1e-9 to 1e-16.

The value is used to stop the optimization in the following cases:

- A parameter change by the optimization algorithm would be smaller than this value.
- The change in the cost function is smaller than this value. The optimization has converged.

Finite Difference Factor

The gradients of external models (FMU, ...) are calculated by finite gradients.

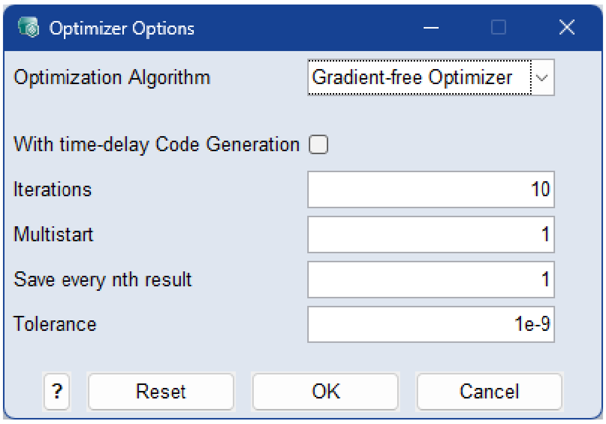
Gradient = **(f(x+e) - f(x)) / e**

with **e = Finite Difference Factor * Normalized Parameter Range * 1.5e-8**

If the output of a model does not change by such a small value, the optimizer stops at the first iteration. In this case, the **Finite Difference Factor** must be increased. Typical values are 10, 100, 1000, 10,000,...

1.5e-8 is the square root of the smallest number that can be shown with a double precision floating point number. If the model uses single precision floating point numbers, set the factor to 10,000 to consider the loss of precision.

Gradient-free Optimizer



With time-delay Code Generation

When activated, C code is generated from the function and optimization works on compiled C code. This only works if the function does not use external models. This speeds up the optimization when the **timeDelay** method is used in the function, in other cases it's usually slower.

Iterations

Enter the maximum number of iterations to be performed during optimization.

Multistart

Enter the number of times to run the optimizer with different initial values.

If set to a number greater than one, the optimizer will run multiple times. The first optimization is started with the current working parameter set, while subsequent optimizations start with random parameter values. This can be used to find a global optimum.

Save every nth result

Save only every n^{th} iteration as a temporary parameterset. Typical values are in the range [1, 100], default is 1.

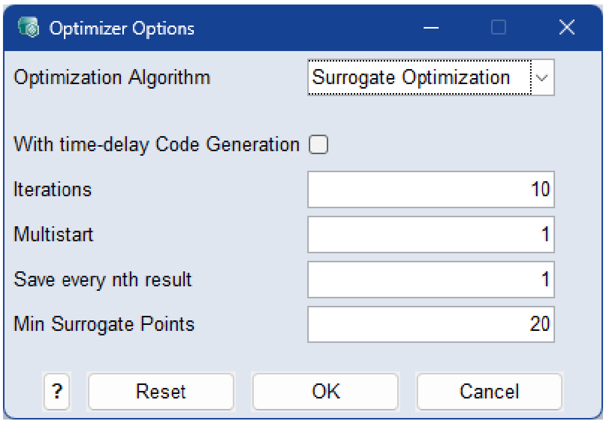
Tolerance

The optimization stop criterion. Typical values range from 1e-9 to 1e-16.

The value is used to stop the optimization in the following cases:

- A parameter change by the optimization algorithm would be smaller than this value.
- The change in the cost function is smaller than this value. The optimization has converged.

Surrogate Optimization



With time-delay Code Generation

When activated, C code is generated from the function and optimization works on compiled C code. This only works if the function does not use external models. This speeds up the optimization when the **timeDelay** method is used in the function, in other cases it's usually slower.

Iterations

Enter the maximum number of iterations to be performed during optimization.

Multistart

Enter the number of times to run the optimizer with different initial values.

If set to a number greater than one, the optimizer will run multiple times. The first optimization is started with the current working parameter set, while subsequent optimizations start with random parameter values. This can be used to find a global optimum.

Save every nth result

Save only every n^{th} iteration as a temporary parameterset. Typical values are in the range [1, 100], default is 1.

Tolerance

The optimization stop criterion. Typical values range from 1e-9 to 1e-16.

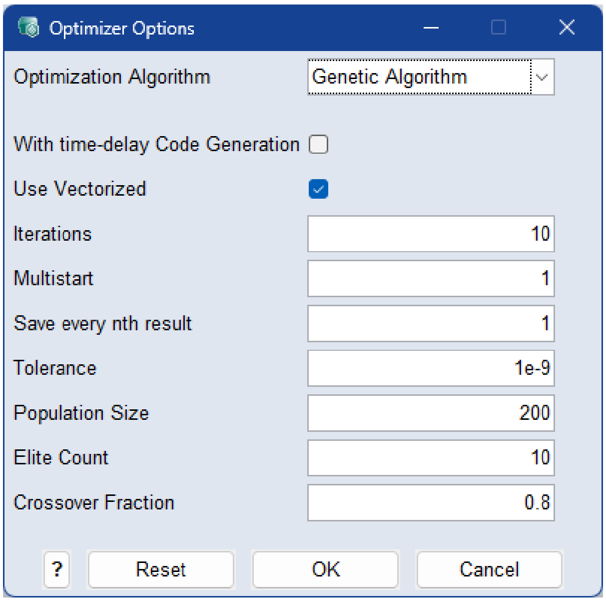
The value is used to stop the optimization in the following cases:

- A parameter change by the optimization algorithm would be smaller than this value.
- The change in the cost function is smaller than this value. The optimization has converged.

Min. Surrogate Points

The surrogate model is a radial basis function model, and the minimum number of basis functions used in the model is defined here. The initial model is assessed at the location of these basis functions.

Genetic Algorithm



With time-delay Code Generation

When activated, C code is generated from the function and optimization works on compiled C code. This only works if the function does not use external models. This speeds up the optimization when the **timeDelay** method is used in the function, in other cases it's usually slower.

Use Vectorized

If activated, the MOCA function is evaluated with all candidate solutions in a single vectorized call. The function's memory usage increases by a factor proportional to the number of individuals in the population during the evaluation process.

Iterations

Enter the maximum number of iterations to be performed during optimization.

Multistart

Enter the number of times to run the optimizer with different initial values.

If set to a number greater than one, the optimizer will run multiple times. The first optimization is started with the current working parameter set, while subsequent optimizations start with random parameter values. This can be used to find a global optimum.

Tolerance

The optimization stop criterion. Typical values range from 1e-9 to 1e-16.

The value is used to stop the optimization in the following cases:

- A parameter change by the optimization algorithm would be smaller than this value.
- The change in the cost function is smaller than this value. The optimization has converged.

Population Size

The number of candidate solutions available during the optimization process. Increasing the population size may lead to better solutions, but also results in longer processing times and higher memory usage. The usual range for this value is between 100 and 1000.

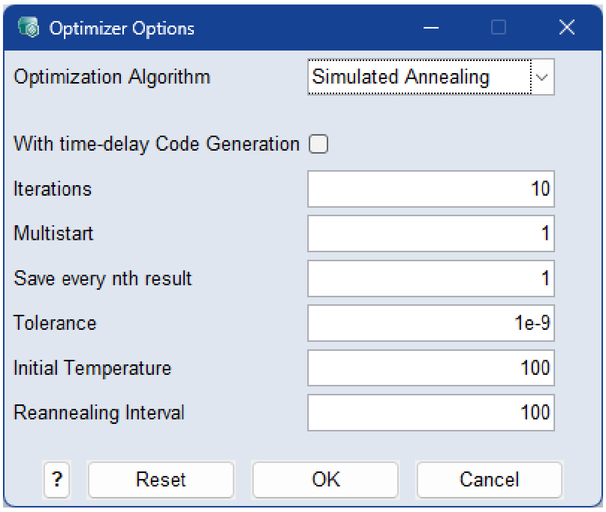
Elite Count

The number of good candidate solutions that are carried over unchanged to the next iteration. Using a smaller number causes the algorithm to converge at a slower pace. Common values range from 5 to 20% of the overall population size.

Crossover Fraction

Crossover candidate solutions are produced by combining two candidates. Mutated candidate solutions are created by changing a candidate at random. The crossover fraction controls the amount of crossover and mutation, with a range of 0 to 1. A value of 0.8 means that 20% of candidates undergo mutation, while 80% are produced by crossover. Typically, values range from 0.5 to 0.9.

Simulated Annealing



With time-delay Code Generation

When activated, C code is generated from the function and optimization works on compiled C code. This only works if the function does not use external models. This speeds up the optimization when the **timeDelay** method is used in the function, in other cases it's usually slower.

Iterations

Enter the maximum number of iterations to be performed during optimization.

Multistart

Enter the number of times to run the optimizer with different initial values.

If set to a number greater than one, the optimizer will run multiple times. The first optimization is started with the current working parameter set, while subsequent optimizations start with random parameter values. This can be used to find a global optimum.

Save every nth result

Save only every n^{th} iteration as a temporary parameterset. Typical values are in the range [1, 100], default is 1.

Tolerance

The optimization stop criterion. Typical values range from 1e-9 to 1e-16.

The value is used to stop the optimization in the following cases:

- A parameter change by the optimization algorithm would be smaller than this value.
- The change in the cost function is smaller than this value. The optimization has converged.

Initial Temperature

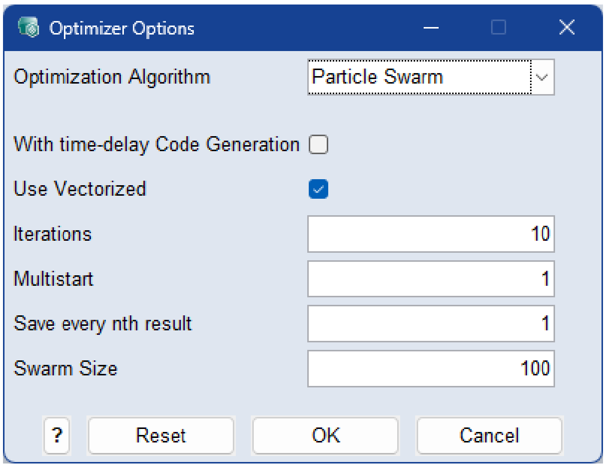
This value controls the probability of accepting worse solutions during the optimization process. Typical values range from 50 to 200.

The initial temperature decreases by **Initial Temperature * 0.95**$^{\text{Iterations}}$ per iteration. As the temperature decreases, the probability of accepting worse solutions decreases.

Reannealing Interval

After this number of iterations, the temperature is increased. Typical values are 50 to 200.

Particle Swarm



With time-delay Code Generation

When activated, C code is generated from the function and optimization works on compiled C code. This only works if the function does not use external models. This speeds up the optimization when the **timeDelay** method is used in the function, in other cases it's usually slower.

Iterations

Enter the maximum number of iterations to be performed during optimization.

Multistart

Enter the number of times to run the optimizer with different initial values.

If set to a number greater than one, the optimizer will run multiple times. The first optimization is started with the current working parameter set, while subsequent optimizations start with random parameter values. This can be used to find a global optimum.

Save every nth result

Save only every n^{th} iteration as a temporary parameterset. Typical values are in the range [1, 100], default is 1.

Swarm Size

The number of candidate solutions available during the optimization process. Enlarging the swarm size leads to longer optimization duration and increased memory usage, but may lead to better solutions. Typical values range from 50 to 200.

For an explanation of the optimization algorithms, see "Optimization Algorithms" on page 52

4.7.4 Consideration of the Roughness

Roughness of a Curve

The roughness *r* describes the change in the slope at the support points of the curve *c*. If the curve is given by an expression *c(x)*, the roughness is given as the sum of the second derivatives at the support points x_i, i=1..k.

For a curve, this means:

$$r_{curve} = \sum_{i=1}^{k} \left(\left. \frac{d^2 c}{dx^2} \right|_{x_i} \right)^2$$

Equ. 4-6: Roughness r of a curve

Roughness of a Map

The Roughness of a map m = m(x_1, x_2) has to consider the second input variable and therefore is defined as:

$$r_{map} = \sum_{i=1}^{k} \left(\left. \frac{d^2 m}{dx_1^2} \right|_{x_{1i}} + \left. \frac{d^2 m}{dx_2^2} \right|_{x_{2i}} \right)^2$$

Equ. 4-7: Roughness of a map

where *K* is the number of the support points (x_{11}, x_{21}), ... , (x_{1K}, x_{2K}) of the map.

The roughness is shown in the "Parameter Optimization Properties" Window (**Optimization Step › Optimization Criteria** button).

4.7.5 Optimization Criterion

To optimize one or more outputs, there is the target criterion **Smoothness** that limits the variation of the stiffness (see "Consideration of the Roughness" on the previous page) of a map or a curve. This factor is a weighted penalty term,

$$\underset{p}{\operatorname{argmin}}\left(\sum_{i=1}^{N}\left(Y_{i,predicted}-Y_{i,measured}\right)^{2}+\sum_{i=1}^{M} Si * r_{curve/map}\right)$$

Equ. 4-8: Smoothness factor S_i

where *S* is the Smoothness factor and *M* the number of support points of the map or curve.

Different Smoothing Factors in X/Y Direction

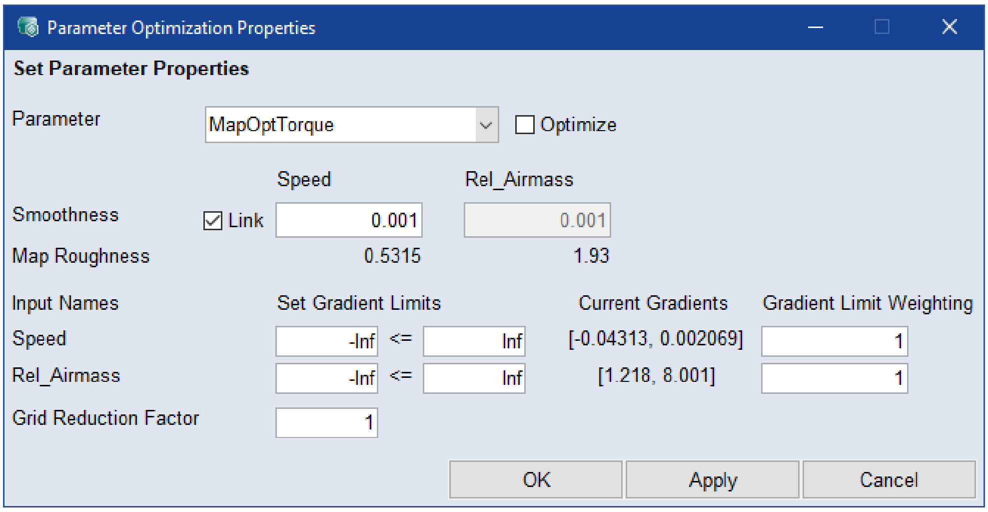
For maps, Cube-3D and Cube-4D, the smoothing factor S is used per input direction. If only one value is given, the factor works for all directions. If a vector is given, each element corresponds to one input direction. In case of a map, e.g., one may set smoothness in X direction to 0.1 and in Y direction to 0.001 by specifying a vector `[0.1 0.001]`.

The smoothing factor has to be a real number `≥ 0` and can be defined in the Optimization Step either via the **Optimization Criteria** button or by directly editing the column "Smoothness" in the parameter table.

Optimization Criteria Selection

For curves, maps, Cube-3D and Cube-4D, the gradients in each respective input direction can be constrained by defining a limit for the maximal and/or the minimal gradient. The gradient constraints can be defined in the Optimization Step via the **Optimization Criteria** button. All gradient constraints are handled as weak constraint by the optimizer.

You can also assign a weight to each set of gradient limits. This weight sets the priority of the gradient limits in relation to the primary optimization criteria. A higher weight makes the gradient limits more important.



A step-by-step instruction how to set an optimization criterion is given in the online help.

4.7.6 Optimization Without Sequence

Unless your project must fulfill special requirements, all steps for optimization are performed in the "Optimize" tab of the Optimization pane:

- preparing the optimization
  - specifying optimization options
  - specifying optimization criteria
  - specifying local constraints
- running the optimization
- performing optional activities
  - showing data
  - dealing with reference parameters

See the online help for instructions how to do these steps.

4.7.7 Optimization With a Sequence

If your project must fulfill special requirements, you can define a custom sequence of optimization steps in the **Sequence** tab of the **Optimization** step.

For example:

- First, calibrate one map using a subset of the available data.
- Then, retain the result of this first calibration and continue by calibrating the remaining parameters.

Once your sequence is complete, you can run the optimization.

For detailed instructions on how to create and execute an optimization sequence, refer to the online help.

4.7.8 Parameter Correlation

You can use the **Analysis › Parameter Correlation** menu option to check if the parameters are correlated. A strong correlation (+1 or -1) means that two parameters do not independently affect the function node. To determine the correlation, the following happens.

ASCMO-MOCA calculates the gradient matrix G regarding all parameters:

$$G = \frac{\partial F(x,p)}{\partial p}$$

with

- **F** - the optimization function to be minimized
- **x** - training data
- **p** - parameter

ASCMO-MOCA then calculates the covariance matrix C:

$$C = ((G^T * G) + I))^{-1}$$

with

- G^T - transpose of G
- I - identity matrix

Then the correlation coefficients c between parameters a and b are calculated. C_{ab}, C_{aa}, and C_{bb} are elements of the covariance matrix.

$$c_{ab} = \frac{C_{ab}}{\sqrt{C_{aa}} * \sqrt{C_{bb}}}$$

The results are displayed in the "Parameter Correlation" window.

4.7.9 Parameter Sensitivity

You can use the **Analysis › Normalized Parameter Sensitivity** menu option to check the influence of parameters on function nodes.

ASCMO-MOCA calculates the gradients G of a node regarding the parameters for all parameters p_j for all training data points x_i:

$$G = \frac{\partial F(x,p)}{\partial p}$$

with

- **F** - the optimization function to be minimized
- **x** - training data
- **p** - parameter

The gradient is normalized to the range of the parameter:

$$g_{ij} = \frac{\partial F(x_i,p_j)}{\partial pj} * (u_j - l_j)^{-1}$$

with

- u_j - upper limit of parameter p_j
- l_j - lower limit of parameter p_j

The results are displayed in the "Normalized Parameter Sensitivity" window as follows:

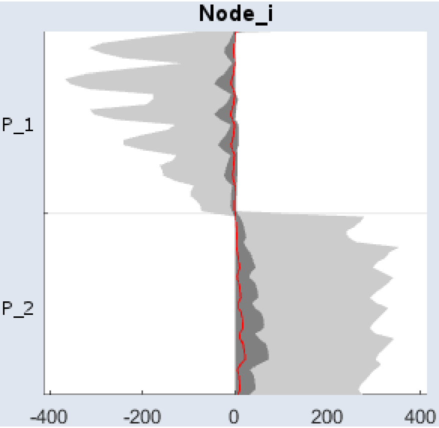


Fig. 4-7: Normalized parameter sensitivity

- dark gray area: maximum gradient regarding one parameter p_j over all training data points
- red line: mean gradient regarding one parameter p_j over all training data points
- light gray area: mean gradient ± 1 σ regarding one parameter p_j over all training data points

Note

Smaller values indicate less influence of the parameters on a node.

4.8 Symbolic Regression

Symbolic regression is a type of regression analysis on a symbolic level. Transferred to an application in the context of ASCMO-MOCA, this means to automatically find an equation-based or hybrid (mixing equation- and data-based approaches) model with the following properties:

- Model corresponds to a dataset well in a statistical sense.
- Model is as compact as desired.
- Model is human-interpretable.

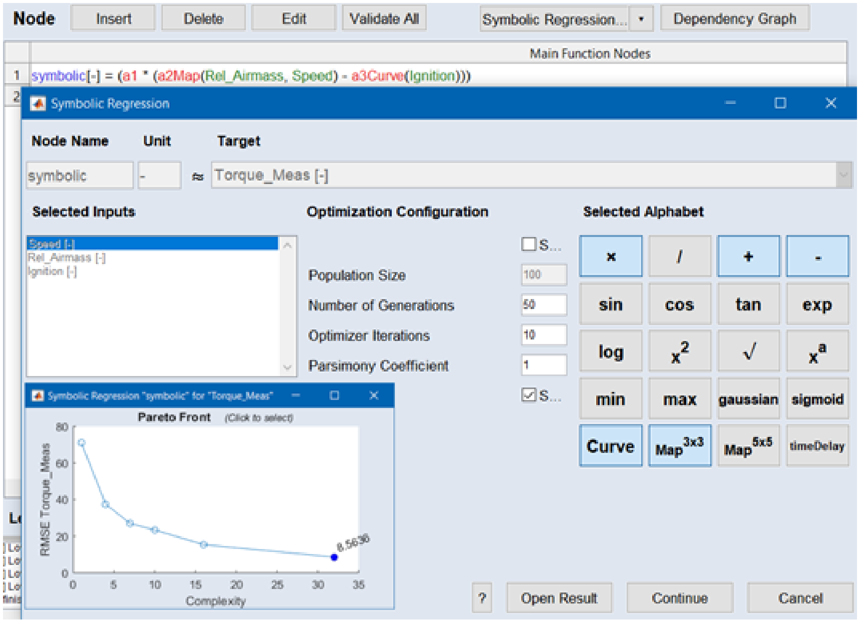
The symbolic regression plugin of ASCMO-MOCA provides a solution to this task by carrying out optimizations on the structural level of equations and local optimizations to fit identified models to data. In terms of embedded software function engineering these two steps correspond to function engineering and calibration, respectively.

ASCMO-MOCA supports engineers carrying out this steps more efficiently and effectively using artificial-intelligence. From an alternative perspective, the method can also be viewed as an automated way of system identification.

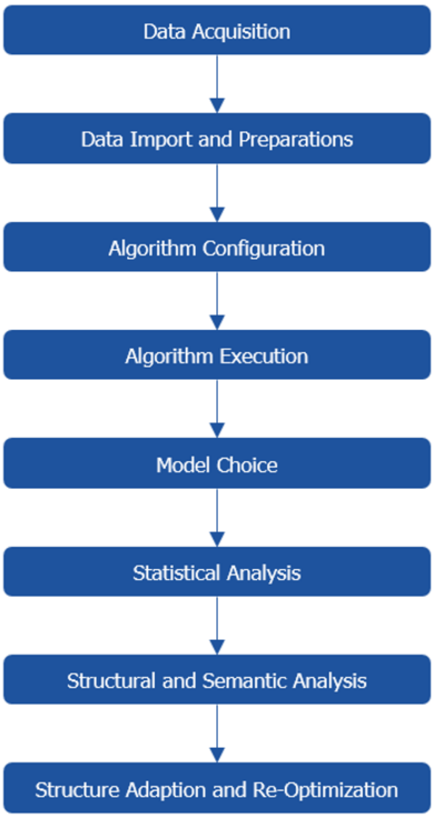
The Symbolic Regression Feature is located in the Function Step.

4.8.1 Symbolic Regression Workflow

Symbolic Regression supports you to derive automatically a formula from the data with a genetic algorithm. You can also use Symbolic Regression for (time-dependent) dynamic functions with time-delay.



The workflow to carry out Symbolic Regression is comprised of several steps.



- **Data Acquisition**:
  Acquire data which describes the system well. The dataset size should allow for a meaningful split into training, validation, and test data.
- **Preparations and Data Import**:
  Requirements for Data
  - Clean: all values are meaningful, no NaNs given, no errors from defect measurement devices, etc.
  - Labeled: For all values the measured value (label) is defined along with its units.
  - Splittable: The dataset can be split into training, validation and test data. All being representative in a statistical sense.
    After data preparation, start import.
- **Algorithm Configuration**:
  - Start the algorithm configuration with **Start Symbolic Regression** button in the Function Step.
  - Define the regression problem by setting the target quantity in the field **Target** and choose the input quantities by shifting them from **Available Inputs** to **Selected Inputs**.
  - Configure the algorithmic details in the fields below **Optimization**

Configuration. See Algorithmic Details.

- Choose the function/operation types you want to use by clicking on the elements below **Selected Alphabet**.

— **Execution**:
Start the algorithmic execution by clicking on **OK** in the "Symbolic Regression" window and stop the execution at any time by clicking on **Stop** in the status bar beneath log window (main window). In the log window on the command line you will see the value of the selected **Fitness Method** for the best model which was found at the current iteration step.

— **Model Choice**:
Once the algorithm is finished, ASCMO-MOCA will open a window showing the pareto-front. The latter is made of the models contained in the pareto-set. The pareto-set is defined in the space spanned by Fitness Method (y-axis) and Complexity (x-axis). Click on the bubbles to select a model. You will see the value of the currently selected Fitness Method right at the bubble. Additionally ASCMO-MOCA provides you with the selected model in the Function Step.

— **Statistical Analysis**: Evaluate the performance of the selected model in a statistical sense by choosing **Analysis › Residual Analysis**. See also Residual Analysis.

— **Structural and Semantic Analysis**:
The Function and Parameters Step allow you to analyze and interpret the model on a structural and semantic level, respectively. The Function Step gives you an inside into the model itself.
The Parameters Step allows a detailed inspection of all parameters and maps, which are used by the model of your choice.

— **Structure Adaption and Re-Optimization**:
ASCMO-MOCA seamlessly allows you to adapt the model structure as described in Functions. Once done, you can carry out a re-optimization of this structure. See also "Optimization" on page 51.

4.8.2 Algorithmic Details of Symbolic Regression

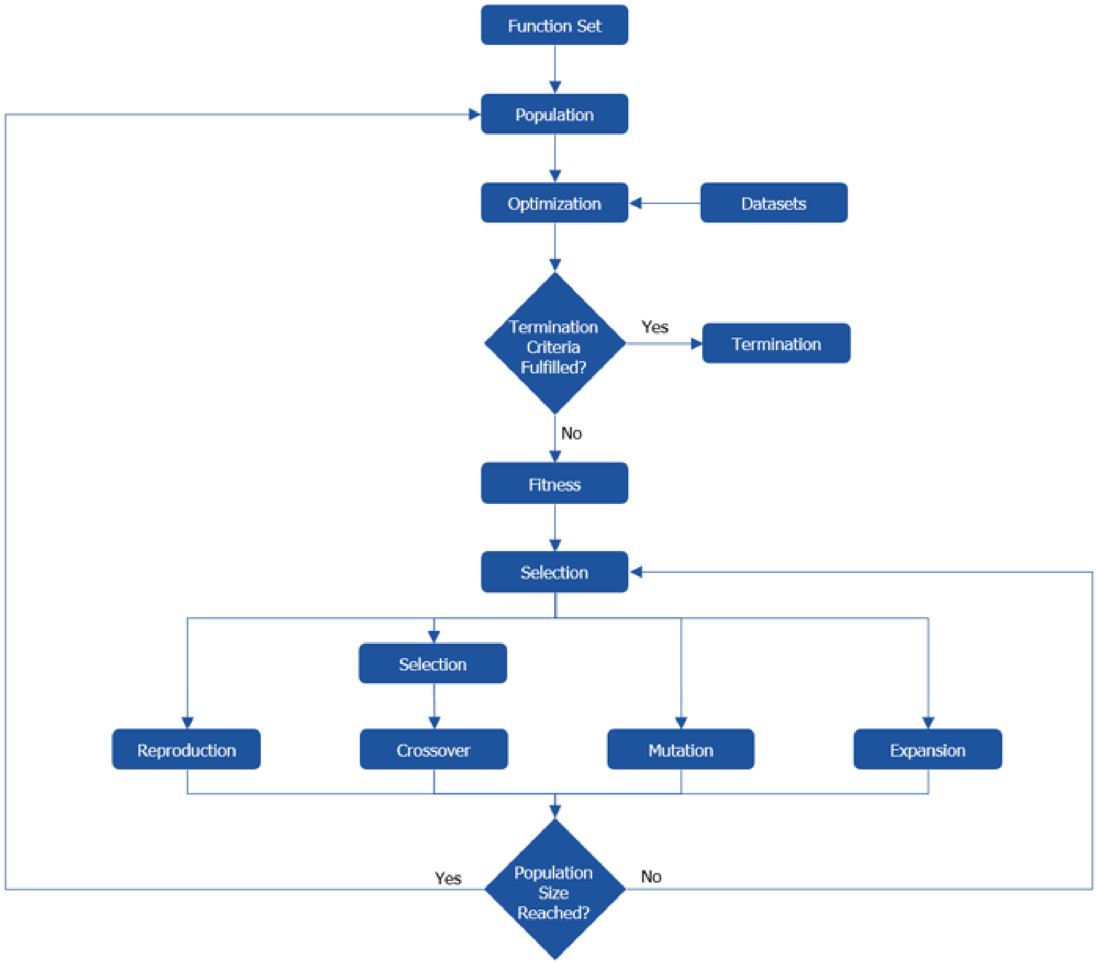


Fig. 4-8: Algorithmic Details of Symbolic Regression

To carry out symbolic regression, ASCMO-MOCA uses a modified version of genetic programming. The algorithm is specifically suited to fulfill the required task. The major algorithmic steps are described below and in the image.

Function Set

A set of functional expressions is defined which are used during the course of the algorithm. In ASCMO-MOCA this step is covered by choosing the function/operation types you want to use by clicking on the elements in the "Selected Alphabet" area.

Population

After the function set is defined, a population/set of models is created on a stochastic basis. Starting with this step the models are represented as graphs. The creation process is constrained w.r.t. the population size and the graphs' dimensions. During the course of the algorithm the population evolves from the evolution operations. Each iteration-step in the main loop of the algorithm corresponds to one generation.

Corresponding Configuration Parameters (**Optimization Configuration** & **Initial Configuration**) in the "Symbolic Regression" window

- **Random Seed**: Integer used as the seed for the random number generation, which effects all stochastic steps within the algorithm. Modify this value to improve the performance in critical cases.
- **Population Size**: Number of models contained in each population. For hard problems population sizes of several thousands have performed well.
- **Population Creation Method** (**Initial Configuration**): The initial population is created with one of the three different methods. For unknown problems the half-and-half methods is a good choice.
  - **half and half**: One half of the population is created with full method and the other half of the population is created with the grow method. The used maximum depth value is determined from a uniform distribution within **Population Init Depth Min** and **Population Init Depth Max**, for each graph individually.
  - **full**: Each graph is created such that it has a depth of **Population Init Depth Max** in all branches.
  - **grow**: Each graph is created such that at least one branch has a depth of **Population Init Depth Max**.
- **Population Init Depth Min**: Minimum initial depth.
- **Population Init Depth Max**: Maximum initial depth. Graphs with a depth of ten are already huge and very complex. Numbers around four are more appropriate.
- **Max Program Complexity** (**Optimization Configuration**): No graphs are created (initially and during evolution) having a larger complexity. A graphs' complexity is an integer given by the sum of complexities of all nodes the graph is made of. Node complexities are defined ASCMO-MOCA internally based on experience. Using this option offers you to fasten the algorithm's execution if you have a clear expectation and experience on the complexity for your symbolic regression problem.
- **Max Program Depth**: No graphs are created (initially and during evolution) having a larger depth.
- **Max Program Size**: No graphs are created (initially and during evolution) having a larger size (number of nodes).

Datasets

Naturally, symbolic regression requires one or several datasets to be carried out.

Optimization

For each graph an individual and local optimization is performed. The goal of this optimization is to minimize the value obtained by using the method given by the Fitness Method.

Corresponding Configuration Parameters (**Optimization Configuration**)

- **Optimizer Method**
  - **Levenberg-Marquardt** method (LM)
  - **Trust-region-reflective** method
  - **Dogbox** algorithm
- **Optimizer Iterations**: Maximum number of iterations the local optimization should be carried out.

Termination criteria fulfilled Request

The algorithm stops if one of the following criteria holds:

- It holds for one of the models in the population that the value obtained by using the method given by the Fitness Method is smaller than the termination threshold.
- The maximum number of generations is reached.

Corresponding Configuration Parameters (**Optimization Configuration**)

- **Termination Threshold**: Threshold used for termination. The threshold is a positive double.
- **Number of Generations**: Maximum number of generations.

Fitness

For each model in the population it's fitness is computed. To do so the raw-fitness is computed as the sum of two parts:

1. The value obtained by using the method given by the fitness method, the cost-function.
2. A scalar value penalizing the complexity of the graph.

The actual fitness itself is obtained from normalizing the raw-fitness to values within [0,1].

Corresponding Configuration Parameters (**Optimization Configuration**)

- **Fitness Method**: Various cost-function types are available.
  - **rmse** Root mean squared error.
  - **mse** Mean squared error.
  - **abs** Normalized absolute difference (mean of L^1-norm).
- **Parsimony Coefficient**: Factor to weight the impact of the complexity penalty. For unknown problems, a value of 1 is a reasonable starting point. For smaller values of the Parsimony Coefficient the formula gets more complex.

Program Selection

The structural optimization of the graphs starts with selecting graphs from the population. The selection is done on the basis of the fitness obtained in the previous step, with the methods described below.

Corresponding Configuration Parameters (**Optimization Configuration**)

- **Program Selection Method**:
  - **tournament**: From the given population **Tournament Size** graphs are randomly drawn from the population and the one with the best fitness is selected.
    › **Tournament Size**: Number of programs to select from tournament. Choosing the minimum of 2 individuals will lead to highly volatile progress of the algorithm. If the number converges to **Population Size**, a dominance of the fittest programs is enforced. Choosing about 10% of the Population Size is a good starting point.
  - **fitness-based**: A probability distribution is derived from the fitness of all graphs in the current population. This distribution is sampled to select graphs. The fitter a graph is, the more likely it will be selected.
  - **greedy overselection**:The population is divided into a high-fitness group and a low-fitness group. For both groups fitness-based selection is used. If the high-fitness group is used it is determined with **Probability Top**. Otherwise the low-fitness group is chosen.
    › **Fraction Top**: The fraction of the population defines the size of the high-fitness group. The group is filled with the graphs having the highest fitness.
    › **Probability Top**: Probability with which the high-fitness group is used for fitness-based selection.
  - **multi tournament**: Similar to tournament. With a selectable probability, however, fitness/complexity is taken as a criterion instead of just fitness. This option states the a multi-criterial optimization possibilities, only given for a gradient-free technique like genetic programming.
    › **Tournament Size**: Number of programs to select from tournament. Choosing the minimum of 2 individuals will lead to highly volatile progress of the algorithm. If the number converges to **Population Size**, a dominance of the fittest programs is enforced. Choosing about 10% of the Population Size is a good starting point.
    › **Probability Fitness**: Probability to use fitness/complexity instead of fitness.

Evolution

The evolution step comprises the graph-modification operations of reproduction, expansion, mutation, and crossover. It is the actual structural optimization step and relies on the graph selected in the previous step. The probabilities for the different techniques determine how likely they are to be applied to this graph.

Corresponding Configuration Parameters (**Evolution Probabilities**)

For the four options, each probability defines how likely the according option is carried out. The sum of the probabilities has to equal 100.

- **Crossover** Probability: Setting it in the range of 80 - 90 is a meaningful choice.
- **Reproduction** Probability: This operation ensures that graphs with a good fitness are likely to be taken-over unchanged to the next generation.
- **Expansion** Probability
- **Mutation** Probability

Reproduction

Reproduction of a graph simply means to take it over as-is in the population of the next generation.

Expansion

The expansion of a graph is carried out by:

- randomly selection a terminal node of the graph,
- creating a new random graph with depth two,
- replacing the terminal node by the new graph.

If the expanded graph has a better fitness than the original one, it is taken over to the next generation. Else, the original graph is taken over as-is.

Mutation

A mutation of a graph is be carried out with three different operations, each of them being applied to a randomly selected node of the graph.

- New Tree: The selected node and all sub-nodes are replaced by a newly created graph with a maximum depth of three.
- Hoist: The selected node of the graph is removed, keeping the consistency of the graph. This method helps to prevent bloat of the graphs.
- Point: The selected node is replaced by a random node with the same number of inputs.

How likely which method is applied is determined by the individual probability. The sum of all three probabilities has to equal 100.

Corresponding Configuration Parameters (**Mutuation Probabilities**)

- **New Tree**: Probability to carry out the new tree operation.
- **Hoist**: Probability to carry out the hoist mutation operation.
- **Point**: Probability to carry out the point mutation operation.

Crossover

The crossover operation essentially is about recombining two graphs to find an even more fitter one. The operation is carried out by the following steps:

1. The graph selected in the previous step is defined as the target graph.
2. A second graph is selected – exactly as described in Program Selection – and defined to be the source graph.
3. In both graphs one node with all its sub-nodes is selected randomly as branches to-be-exchanged.
4. The branch in the target graph is replaced by the branch of the source graph.

The resulting graph is taken over to the population of the next generation.

Reached Population Size Request

The evolution operations are applied until the population of the next generation has **Population Size** number of members.

5 Working with ASCMO-MOCA

5.1 User Interface of ASCMO-MOCA

This section provides an overview of the user interface of ASCMO-MOCA.

A detailed description of the functions of the main menu and the various dialog windows associated with it is located in the context-sensitive online help (<F1> or **Help › Online Help**).

5.2 Elements of the ASCMO-MOCA User Interface

This chapter provides a brief overview of the user interface key elements. The following figure shows the main user interface of ASCMO-MOCA.

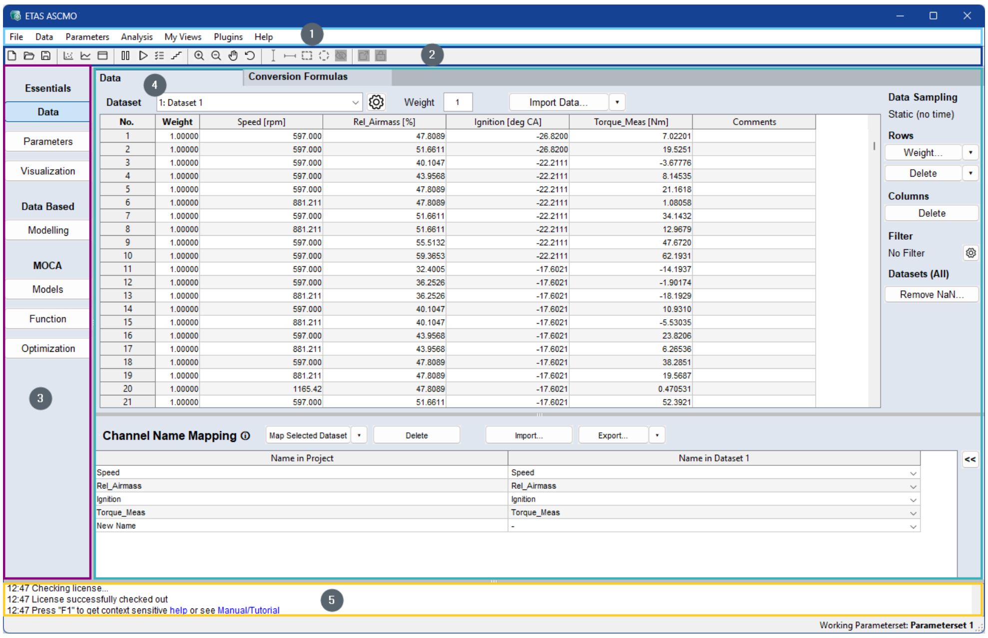


Fig. 5-1: Main user interface elements of ASCMO-MOCA

Most of the user activity will take place in the main working window. Moreover, there is a number of interactive options in the navigation and the main menu bar that are described below.

① Main menu

② Toolbar (see "Toolbar" below)

③ Navigation pane (see 82)

④ Main working window

⑤ Log window (see "Log Window" on page 84)

Status bar (footer) with current state information

5.2.1 Main Menu of ASCMO-MOCA

For details of the functions of the main menu, refer to the online help (‹F1› or **Help › Online Help**).

5.2.2 Toolbar

The toolbar contains a number of buttons that will run the following functions.

	New project	Opens a new instance of ASCMO-MOCA.
	Open project	Opens a data selection dialog where you can open available projects (`*.moca`).
	Save	Saves the current project.
	Scatter plot for training data	Opens the **Data and Nodes - Training Data** window. See also "Graphical Analysis of Data and Function Nodes" on page 21.
	Scope view of residuals, function evaluation and training data	Opens the **Scope View - Training Data** window.
	Open visualization in separate window	
	Pause evaluation	Pauses the evaluation of functions and external models unless you click **Optimize**. When activated, only NaNs are returned.
	Recalculate once, even if pause is active, update RMSE display	
	Select active datasets	Opens the Active Datasets window.
	Automatically quantize all parameters and limits of all parametersets on parameter change	Automatic quantization automatically applies the A2L conversion formula after a parameter change. Quantization is not active during optimization, but is applied when optimization is finished.
	Zoom in	By clicking in the plot, the visualization becomes larger.
	Zoom out	By clicking in the plot, the visualization becomes smaller.
	Pan	This button allows you to move the plot within the window.
	Rotate 3D	This button allows you to rotate a plot in all dimensions.

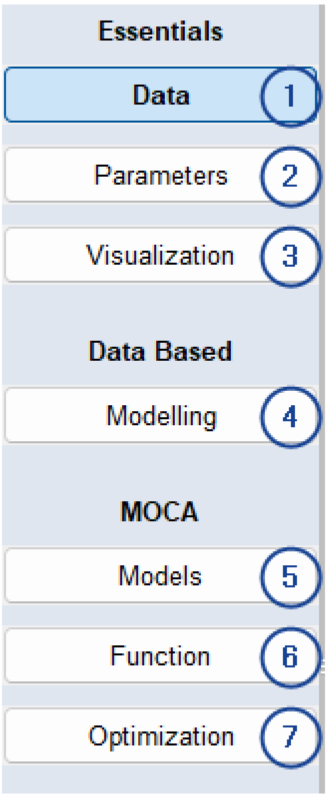
Only available for the **Visualization** step

	Add x axis cursor in plot	
	Add y axis cursor in plot	
	Mouse selection in plot with rectangle	

	**Mouse selection in plot with lasso**	
	**Hide all selections**	
	**Undock current visualization in separate window**	Undocks current visualization tab into a separate window.
	**Lock visualization against changes**	Locks the visualization of all tabs against changes unless you click **Optimize**.

5.2.3 Navigation Pane of ASCMO-MOCA

The navigation pane at the left side of the window leads you through the process steps from the import of the measuring data up to the export of the optimized parameters.



1 – **Data**: In the **Data** step, you can import a measurement file, edit the measurement data and export the data to a measurement file. You can also map the data channel from the measurement file to the corresponding function variable (Channel Name Mapping).

For more information, see "Data" on page 19, the tutorial (see "Step 1: Data Import" on page 88), and the online help

2 – **Parameters**: In the **Parameters** step, you can manage and modify maps, curves, scalars, cube-3D, cube-4D, compressed model and matrix for use in the **Function** step.

For more information, see "Parameters " on page 27, the tutorial (see "Step 3: Parameters" on page 104), and the online help.

3 – **Visualization**: In the **Visualization** step, you can visualize your data.

For more information, see " Visualization " on page 33, and the online help.

4 – **Modelling**: In the **Modelling** step, you can create and train data-based models (e.g., machine learning or black box models).

Note

To use this step you need an ASCMO-DYNAMIC license.

5 – **Models**: In the **Models** step, you can import models and link the parameters, inputs, and outputs to the parameters available in ASCMO-MOCA.

Note

To use a Simulink® model in ASCMO-MOCA, a Simulink® installation with a valid license is required.

For more information, see "Models" on page 40, the tutorial (see "Step 4: Models" on page 105), and the online help.

6 – **Function**: In the **Function** step allows you to construct a function from the stepwise creation of the nodes. The main point here is to link the parameters and the data.

For more information, see "Function" on page 43, the tutorial (see "Step 5: Build Up the Function" on page 115), and the online help.

7 – **Optimization**: In the Optimization step, you can perform the following main tasks:

- Define the parameter-based optimization criterion (e.g., smoothness, gradient constraints).
- Define parameters as a reference for comparison with subsequent optimization results.
- Starting the optimization.
- Export optimized parameters.

For more information, see "Optimization" on page 51, the tutorial (see "Step 6: Optimization" on page 122), and the online help.

5.2.4 Log Window

The bottom part of the main window displays information about the current program sequence, e.g. information about the optimization.

Blue underlined words in the log window are links that open, e.g., the online help or the user guide (a and b in the figure) or give access to sample projects (c - e in the figure). In addition, the log files can be saved for analysis and error handling reasons.

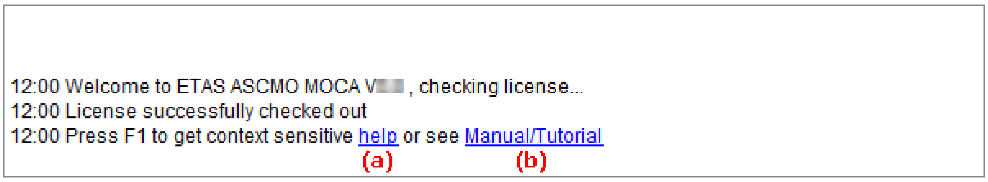


Fig. 5-2: Information in the log window (example; a: link to the online help, b: link to the User Guide PDF)

Saving the logfile

1. Right-click in the log window and select **Save Log to File** from the context menu.

   The "Save Log file As" window opens.

2. Insert a file name and click **Save**.

   The log file is saved.

6 Tutorial: Working with ASCMO-MOCA

This chapter will help you with an example to familiarize yourself with the basic functions of ETAS ASCMO-MOCA.

6.1 About this Tutorial

In this section you can find information about the structure of the tutorial and about the requirements on the measurement data that are used for the parameter optimization.

6.1.1 Challenge in this Tutorial

An ECU often contains models for the calculation of signals, as the sensor-based data logging is either too difficult or too expensive. A common use case is, for example, the calculation of the engine torque. With ASCMO-MOCA you can set up and calibrate a function and optimize the function's parameters based on the measured sensor data. The goal of the optimizer is to minimize the root mean square error "RMSE (Root Mean Squared Error)" on page 25 of the function's parameter. That means that the deviation between the function prediction and the measured sensor data will be minimized.

The structure of the torque related function, that will be modeled step by step during the tutorial, is displayed in "Step 5: Build Up the Function" on page 115.

6.1.2 Structure of the Tutorial

The subsequent tutorial is structured with the following working steps:

- "Start ASCMO-MOCA" on page 87

  This part of the tutorial describes how to start ASCMO-MOCA on your system.
- "Step 1: Data Import" on page 88

  In this first step, the measurement data will first of all be loaded and the channels will be associated with a function node.
- "Step 2: Data Analysis" on page 100

  For clearing up and evaluating the measuring data, at any time, you have the possibility to visualize it after the import graphically for anytime.
- "Step 4: Models" on page 105

  In this step, you are able to link an existing Simulink model with and prepare the mapping of the parameters, the inputs and outputs.
- "Step 5: Build Up the Function" on page 115

  After reading the measuring data and check the plausibility, you can start to set up the function for the torque sensor that will be modeled during the tutorial.

– "Step 3: Parameters" on page 104

  This step allows you to check and possibly adapt the parameters. Only the parameters will be visualized, which you have defined as reference after an optimization "Step 6: Optimization" on page 122.

– "Step 6: Optimization" on page 122

  Before starting with the optimization you have to insert different settings, which influence the optimization. After you have inserted these settings, you can finally start the optimization.

– "Step 7: Export" on page 125

  In this step you will export the created and optimized parameters. The parameters can be exported as DCM file (`*.dcm`) and the project can be saved for the runtime environment with limited functionality.

6.1.3 Requirements on Measurement Data

Basically, a simple rule needs to be considered for a successful parameter optimization in ASCMO-MOCA: The quality of the function's parameter optimization result always depends on the quality of the measurement data. Or in other words: If the parameters have been calibrated based on non space-filling or even wrong data, the function prediction is of little use.

Importing the measurement file in ASCMO-MOCA requires a file with the following properties:

– Data format:
  - Microsoft Excel (`*.xls` / `*.xlsx`)
  - MDA Export (`*.ascii`)
  - Comma Separated Values (`*.csv` / `*.txt`)
  - Measurement Data Format (`*.dat` / `*.mf4` / `*.mdf` / `*.mdf3`)

– Outputs in columns

– Names (and perhaps the units) have to be inserted in the first row (or in the first and second row).

Note

The data used for parameterization do not necessarily have to be derived from a physical experiment (e.g. test bench). They can also be for example a result of a computer simulation.

6.1.4 Data for Modeling

The data used for the parameter optimization in this tutorial can be found in the **`Torque_Data.xlsx`** Excel sheet in the ***`<installation>`*`\Example`** directory.

`<installation>` is the installation directory. By default, *`<installation>`* = `C:\Program Files\ETAS\ASCMO x.x.`

The measurement data from this file meets the already mentioned requirements for a successful parameter optimization in ASCMO-MOCA:

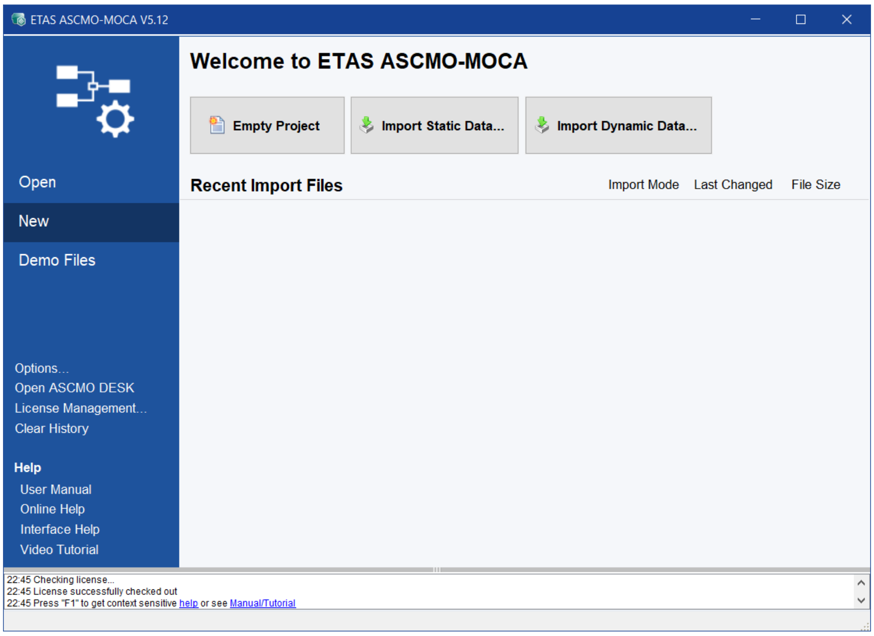
- The experimental design for logging the sensor data (e.g. at a test bench) corresponds to the DoE method, i. e. the measurements have been varied independently and are space-filling.
- The measured sensor data from the measurement file does not include any absurd values (e.g. values ≤ 0 for torque).

6.2 Start ASCMO-MOCA

This part of the tutorial describes how to start ETAS ASCMO-MOCA on your system.

<u>Starting ASCMO-MOCA</u>

1. Do one of the following:
   i. In the **ASCMO-DESK** window, click the **Model Calibration** tile.
   The start window of ASCMO-MOCA opens.



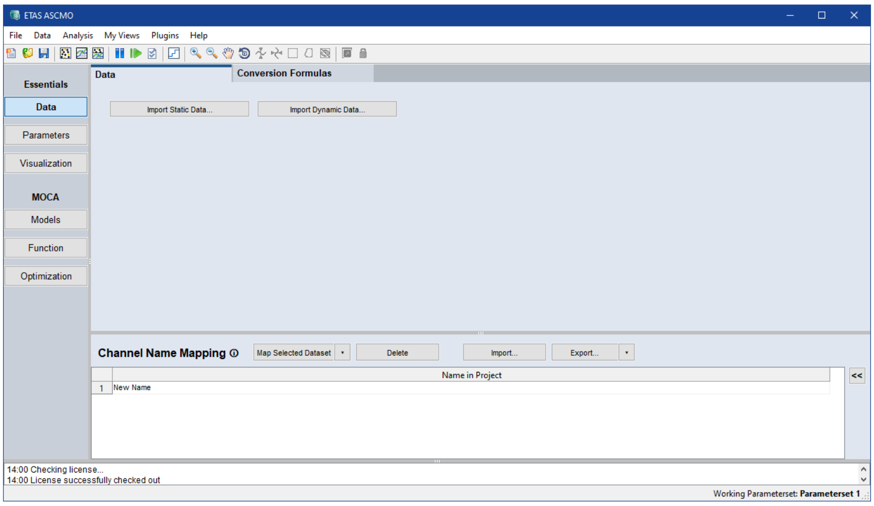
2. Click **New** in the menu panel on the left.
3. Click 

, 

, or 

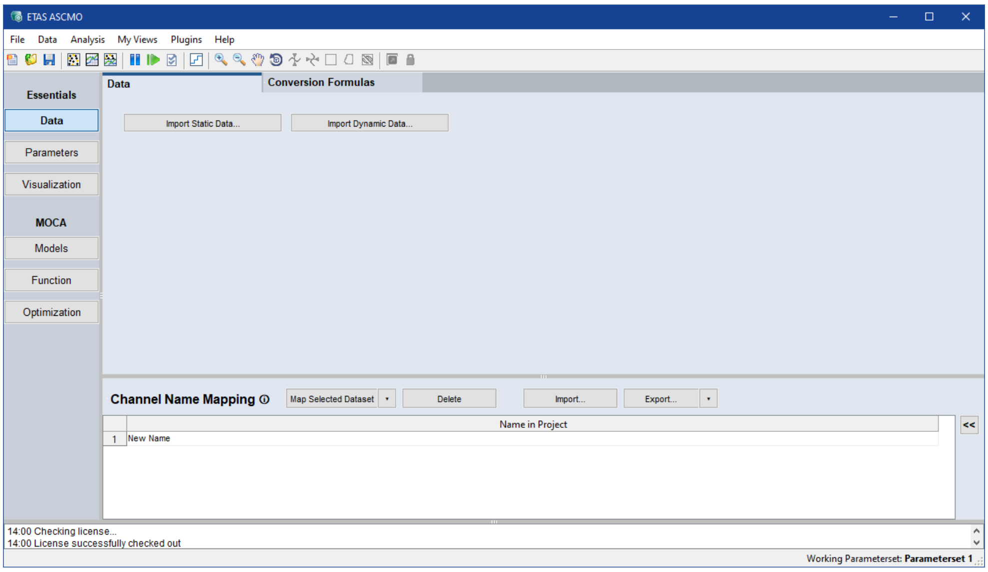
to open an empty project or to directly import static or dynamic data.

If you clicked **Empty Project**, the empty main window of ASCMO-MOCA opens. Now you can start with the measurement data import; see section "Step 1: Data Import" below.



6.3 Step 1: Data Import

In this first step, you will load the measurement data. When you import several files, you can assign the channel names to project specific names, e.g. if the channel names are not identical.



Loading the measurement file

If you want to start a new project, you first of all have to load the required measurement file for parametrization and optimization.

1. In the main working area, click the button **Import Static Data**.
2. In the file selection window, select the file `Torque_Data.xlsx` from the `<installation>\Example\Moca` directory.

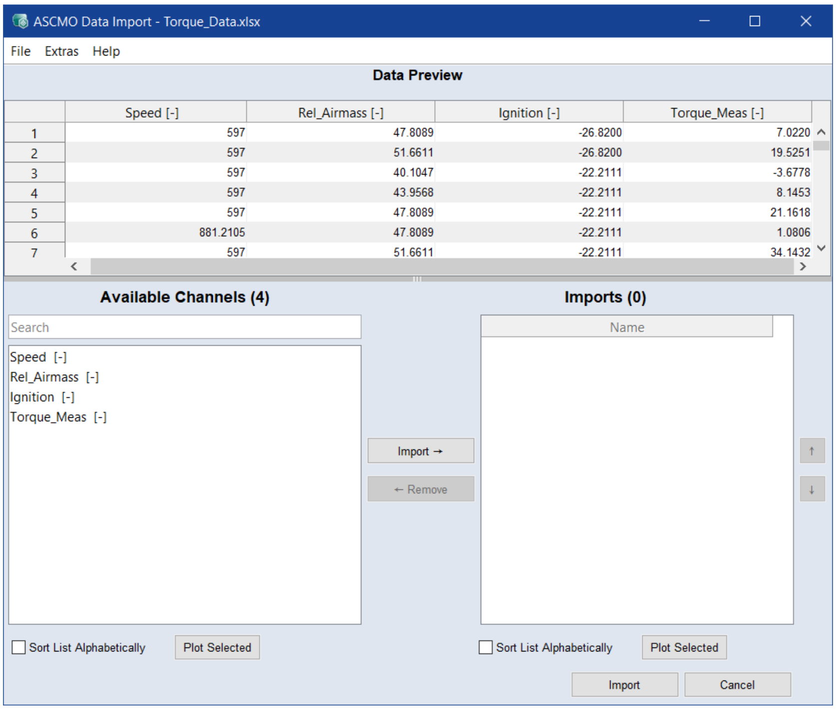
   By default, `<installation>` is `C:\Program Files\ETAS\ASCMO x.x`.

3. Click **Open**.

   If the import file contains several worksheets, the **Sheets** window opens.

4. In the **Sheets** window, select the worksheet you want to import (for this tutorial: `Torque_Data`), then click **OK**.

   The **ASCMO Data Import** window (see) opens.



The **Data Preview** table shows all data in the table. In the **Available Channels** field, you can determine which channels you want to import.

6.3.1 Checking the Plausibility of the Measurement Data

To check the measurement data again prior to the data import, you can display the measurement data, and you can check the relevance of the inputs.

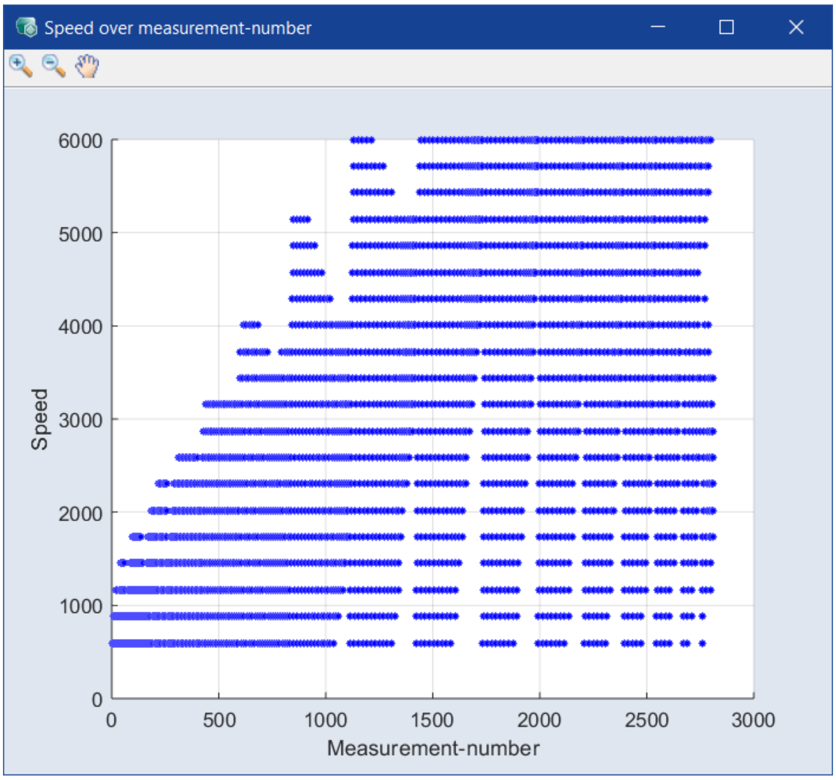
<u>To display measurement data prior to the import</u>

In the **Available Channels** field, select one or more measuring data channels.

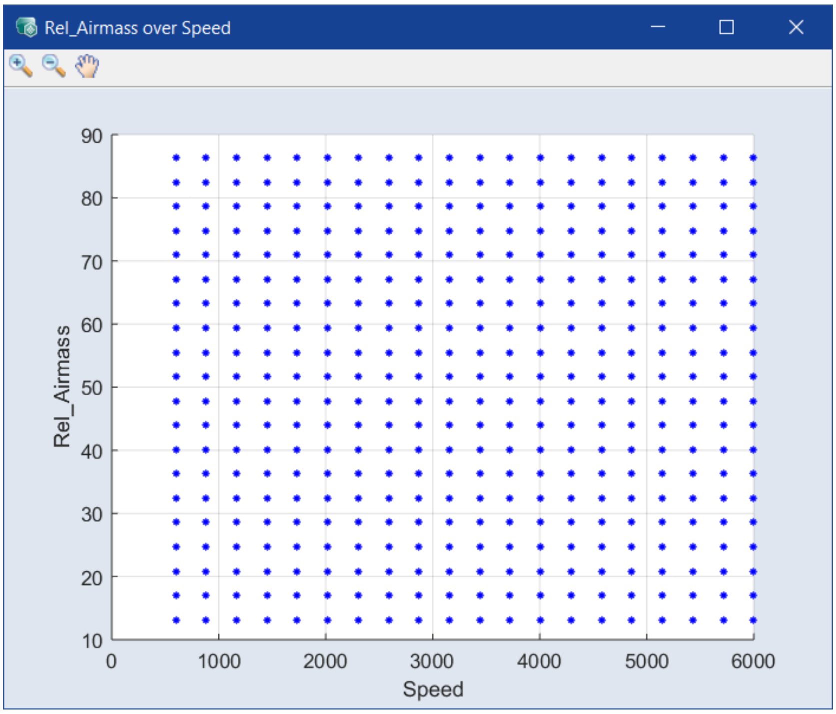
1. You can use the standard CTRL/SHIFT selection functions in the table, or click and hold LMB and drag the cursor over the cells/rows you want to select.
2. Do one of the following:
   - Click **Plot Selected**.
   - Select **Extras › Plot Selected**.

   A window opens that displays one of the plots listed below, depending on the number of selected channels.
3. Check the plots for outliers or other unusual/implausible data.

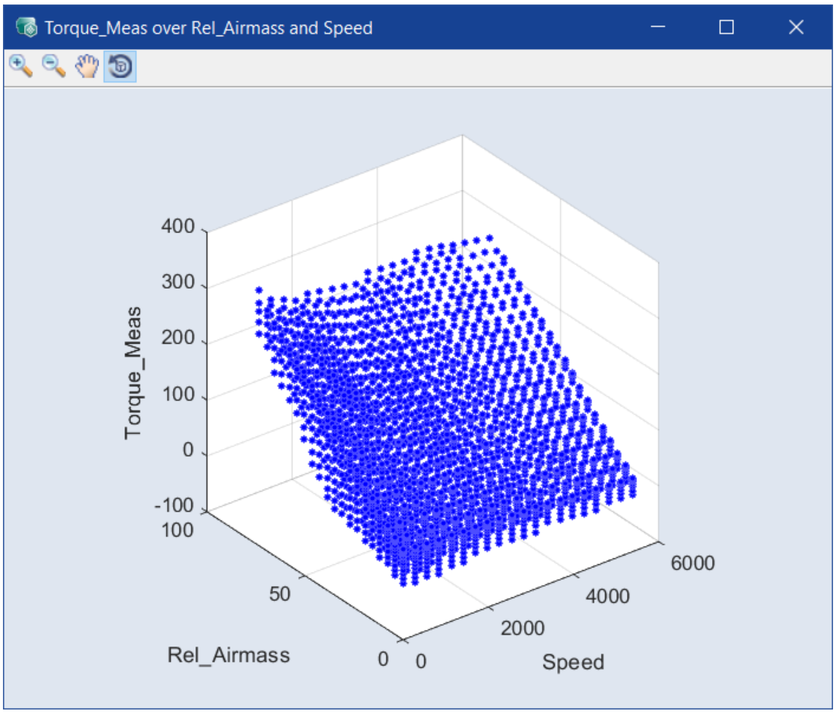
1 channel: measured data against number of measurement - e.g., `Speed` against measurement number.



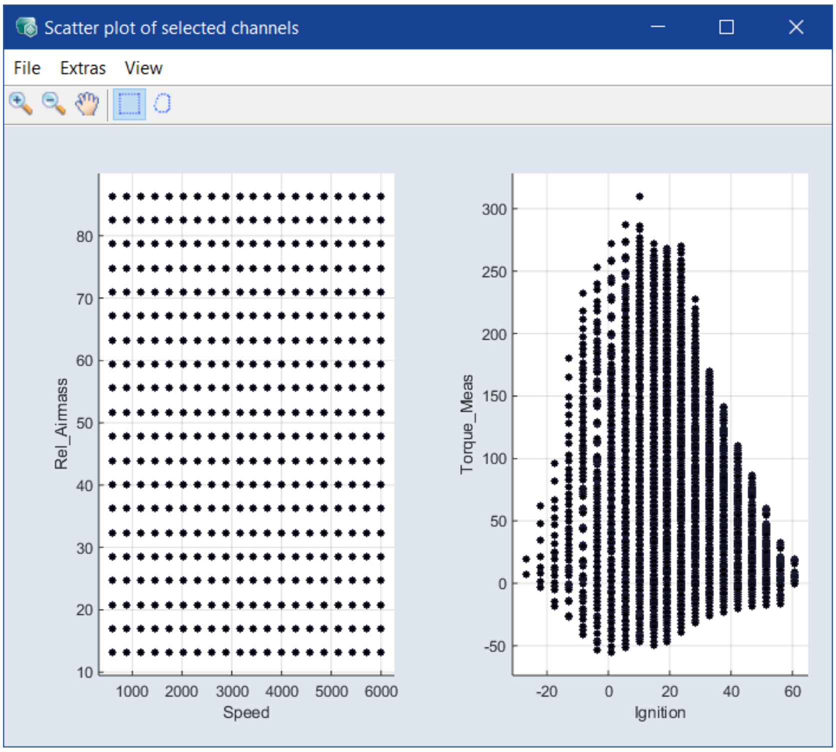
2 channels: data of one column against data of the other - e.g., `Rel_airmass` against `Speed`.



3 channels: data of the third column against the plane set up by the other two - e.g., `Torque_Meas` against the `Speed-Rel_airmass` plane.



4 or more channels: a series of scatter plots.



To check the relevance of the inputs

See "Checking the Relevance of the Inputs" on page 20 for more information on how the relevance is determined.

1. In the **Available Channels** field of the **ASCMO Data Import** window, select the input measuring data channels.

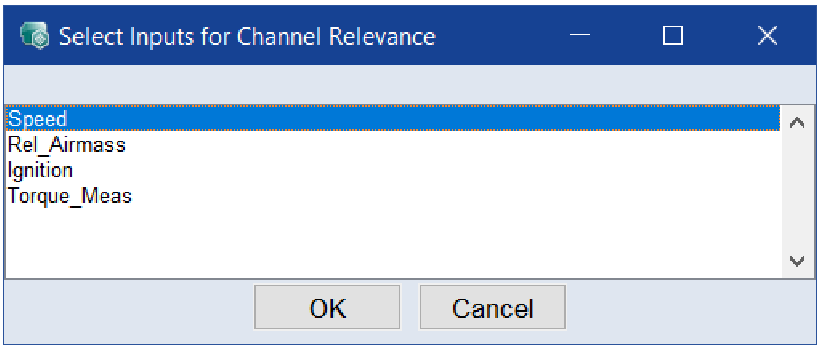
   > **Note**
   >
   > At least one channel must remain unselected. If you select all channels, you cannot check the relevance of the inputs.

2. Click **Import** or double-click a channel.

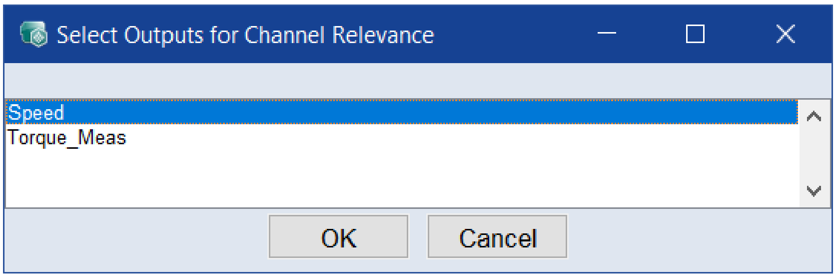
   The selected channels are added to the **Import** list.

3. Select **Extras › Find Relevant Channels**.

   The **Select Inputs for Channel Relevance** window opens.

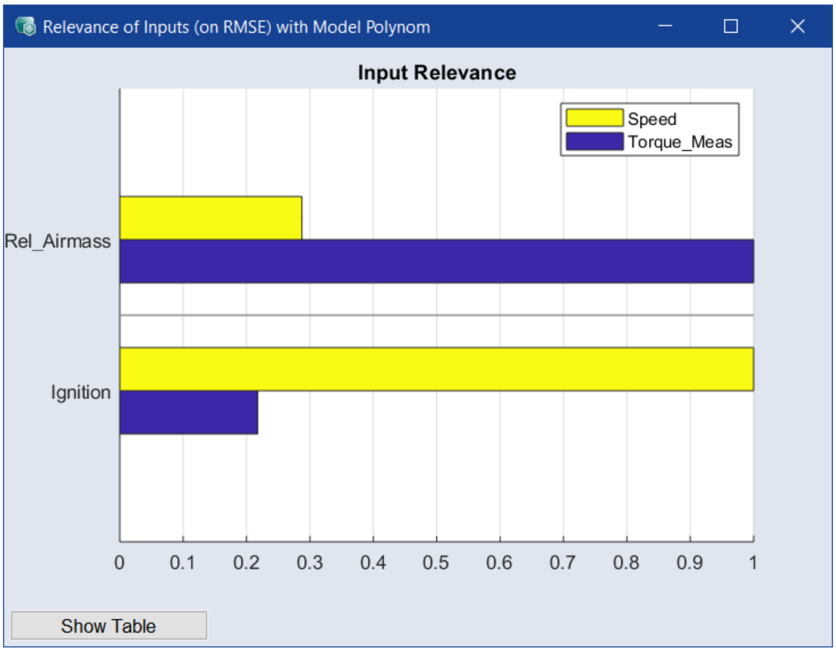
4. In that window, select at least two input channels and click **OK**.
5. The **Select Outputs for Relevance** window opens.

6. In that window, select one or more output channels and click **OK**.

   The "Relevance of Inputs" window opens. It visualizes the influence of the inputs on the outputs.

7. Click **Show Table** to display the results as a data table in the **Relevance of Inputs Table** window.
8. If desired, refine your import selections based on the results.



6.3.2 Saving and Loading a Configuration

A configuration file (`*.ini`) may contain a special assignment of individual measurement data columns to the function variables.

Saving and loading a configuration

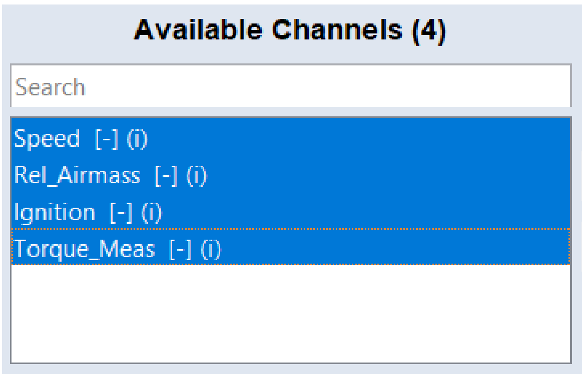
1. In the **ASCMO Data Import**window, select **File › Save Channel Config (*.ini)**.
2. In the file selection window, enter the name of the file under which the current configuration should be saved.
3. To load a previously saved configuration file, select **File › Load Channel Config (*.ini)**.

6.3.3 Importing Measurement Data

Now the data can be imported.

Importing the measurement data

1. In the **ASCMO Data Import** window, select all channels in the **Available Channels** field.

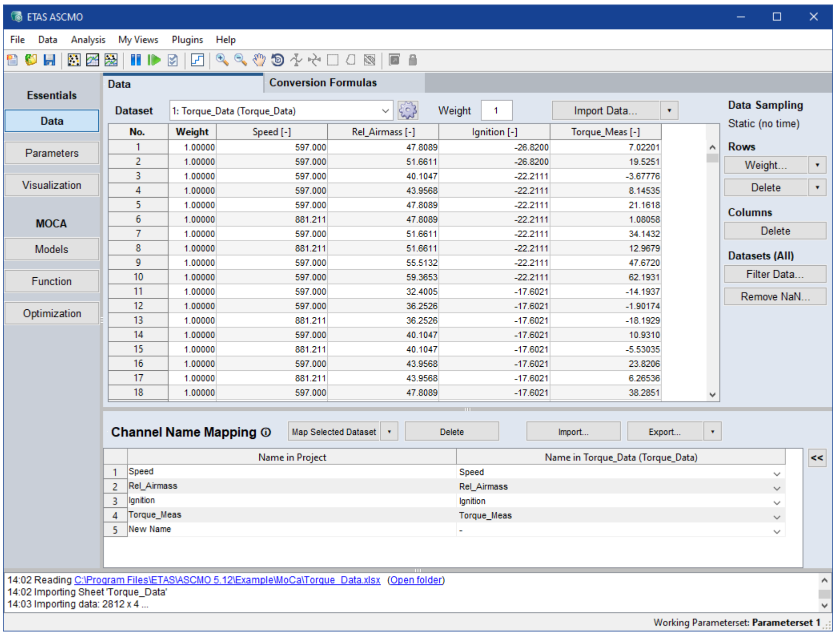


2. Click **Import →**.

   The channels are added to the **Import** list.

3. Click **Import** at the bottom right.

   The data is imported in the current ASCMO-MOCA project. The content of the imported file is displayed in the **Data** tab.



6.3.4 Mapping Measurement Channels to Variables

In the next step, the channels of the imported measurement file have to be assigned to a variable (node), which will be used in the functions later.

> **Note**
>
> If the measurement file's structure meets the requirements (see "Requirements on Measurement Data" on page 86) for the data import , every channel is automatically assigned to the corresponding variable.

Because ASCMO-MOCA automatically performs the assignment, you can proceed with the analysis of the imported measurement data (see "Step 2: Data Analysis" on page 100).

Changing the variable name

If you want to use a different variable name (**Name in Project** column) in your functions, you can change the name in the Data Step, **Channel Name Mapping** table. To do so, proceed as follows:

1. In the **Name in Project** column, click the variable whose name you want to change.
2. Enter the new variable name.
3. Press Enter.

⇨ The new variable name is applied.

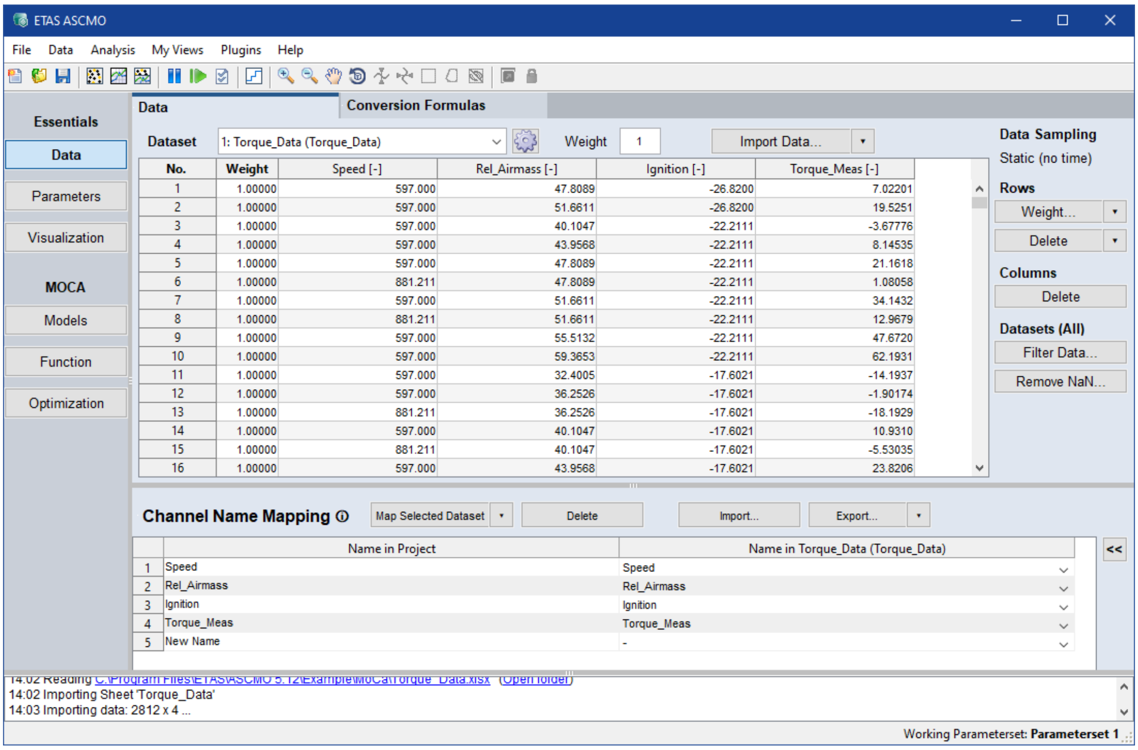
Deleting a mapping

If you do not require certain channels in your measuring data for the parameter, you can delete the mapping in the Data pane, **Channel Name Mapping** table.

1. In the **Name in Project** column, select the desired variable.
2. Click **Delete**.

⇨ The variable is deleted from the **Channel Name Mapping** table.

6.3.5 Working in the Data Step of ASCMO-MOCA



ASCMO-MOCA supports multiple datasets for training and test data, shown as multiple tabs in the data pane. All training datasets together are used for the optimization, while the test datasets are used for evaluation and prediction purposes.

A weight per dataset can be given, which controls the impact of this dataset on the optimization.

Different datasets can have different column names; name mapping is then used to correctly attach the different datasets.

The first import is always used as the first training dataset. When you import another file, you can choose to use this dataset as an additional training data-set or as test dataset or as a replacement. To get the RMSE for a test data set, **Analysis › Residual Analysis › Test Data › *** can be used.

Loading multiple data sets

If you want to load multiple data sets, proceed as follows:

1. Click **Import Data**.
2. In the file selection window, enter or select path and name of the file you want to import.
3. Click **Open**.

   The **ASCMO Data Import - *‹file name›*** window opens.
4. In that window, select the input channels as described in "Loading the measurement file" on page 88.
5. Click **Import**.

   The selected file is imported.

Renaming a data set

1. Go to the tab of the data set you want to rename.
2. Right-click the tab and select **Rename data set** from the context menu.
3. In the **Rename** window, enter the new name, then click **OK**.

Deleting a data set

1. Go to the tab of the data set you want to delete.
2. Click **Delete Dataset**.

   A confirmation window opens.
3. In the confirmation window, click **Delete**.

   The selected data set is deleted from the project.

6.3.5.1 Data Point Weights

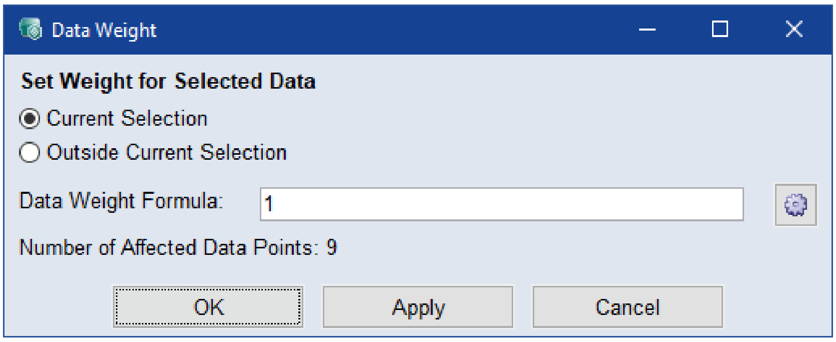
With the **Weight** column, the optimization weight for a data point can be set. Data rows can be set inactive by setting the value to zero. Inactive rows are ignored for the RMSE calculation and optimization. The default is one, and higher values show the optimizer that the respective points are more import-ant, i.e. a stronger emphasis to meet the optimization criterion for this data point. The weight influences the optimization but is not reflected in the dis-played RMSE.

Multiple rows can be selected with the ‹SHIFT› and ‹CTRL› key and left mouse button clicking. Then the weight for multiple rows can be set with the **Weight** button.

Another possibility to set the weight of data points is available in the scatter plot window, opened with **Analysis › Scatter Plot › ***. After marking some data points in a scatter plot, you can use the **Extras › Set Marked Points Weight** menu option to set the weight for the marked data.

Setting the weights of selected data points

1. To set the weights of data points via the **Weight** button, proceed as follows:
   i. Select one or more rows.
   ii. Click **Weight**.



   iii. In the **Data Weight Formula** window, enter the weight and select if you want to set it for the data points inside or outside the current selection.
   iv. Click **OK**.

   The weight is assigned.

2. To set the weight of an individual row, proceed as follows:
   i. In the **Weight** column, double-click in the row you want to edit.
   ii. Enter the number you want to assign.

Adjusting the weights of the entire data set

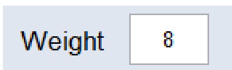
To adjust the weights of the entire data set, proceed as follows:

Note

The data set weight has no effect if there is just one data set.

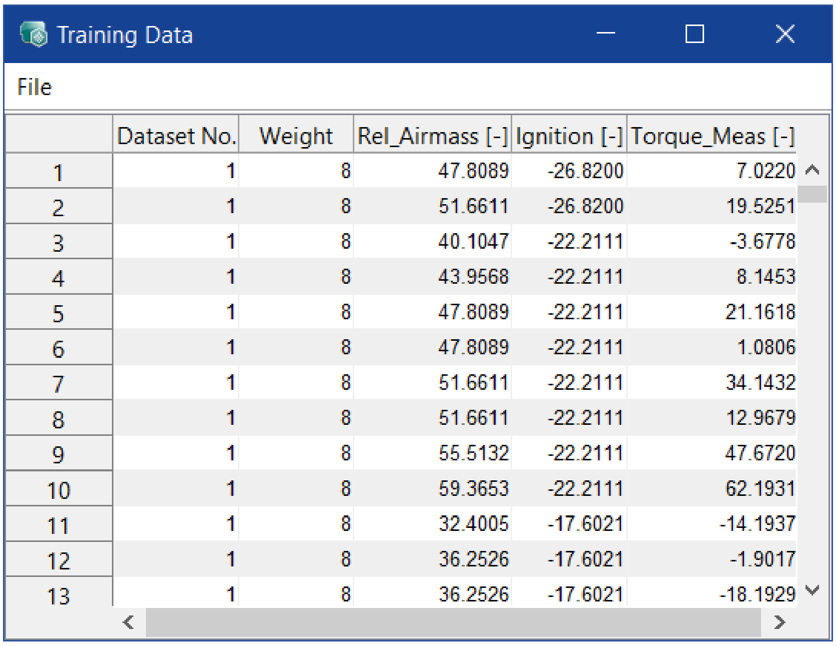
1. In the **Weight** field, enter a value.

   A value of 0 disables the data set. For this tutorial, enter the value 8.



2. To see the effect of the data set weight, select **Analysis › Data Table › Training Data** or **Test Data** or **Training and Test Data**.

   The following window opens.



	Dataset No.	Weight	Rel_Airmass [-]	Ignition [-]	Torque_Meas [-]
1	1	8	47.8089	-26.8200	7.0220
2	1	8	51.6611	-26.8200	19.5251
3	1	8	40.1047	-22.2111	-3.6778
4	1	8	43.9568	-22.2111	8.1453
5	1	8	47.8089	-22.2111	21.1618
6	1	8	47.8089	-22.2111	1.0806
7	1	8	51.6611	-22.2111	34.1432
8	1	8	51.6611	-22.2111	12.9679
9	1	8	55.5132	-22.2111	47.6720
10	1	8	59.3653	-22.2111	62.1931
11	1	8	32.4005	-17.6021	-14.1937
12	1	8	36.2526	-17.6021	-1.9017
13	1	8	36.2526	-17.6021	-18.1929

All values in the **Weight** column have been multiplied with the entered value in the **Weight of *<Dataset_name>*** field. A row weight of 5 and a data set weight of 8 mean that this row has an absolute weight 40.

6.3.5.2 Managing Data in a Dataset

ASCMO-MOCA offers various possibilities to edit, filter and sort the data in a dataset:

Editing data points

To set a value in a particular column and row, proceed as follows:

1. In the column you want to edit, click in the row you want to edit.
2. Enter the number you want to assign.

Removing NaN values

If your imported data contains non-numeric values, you can automatically delete the affected rows in all data sets. Proceed as follows:

1. To delete the rows with NaN values, click **Remove NaN**.

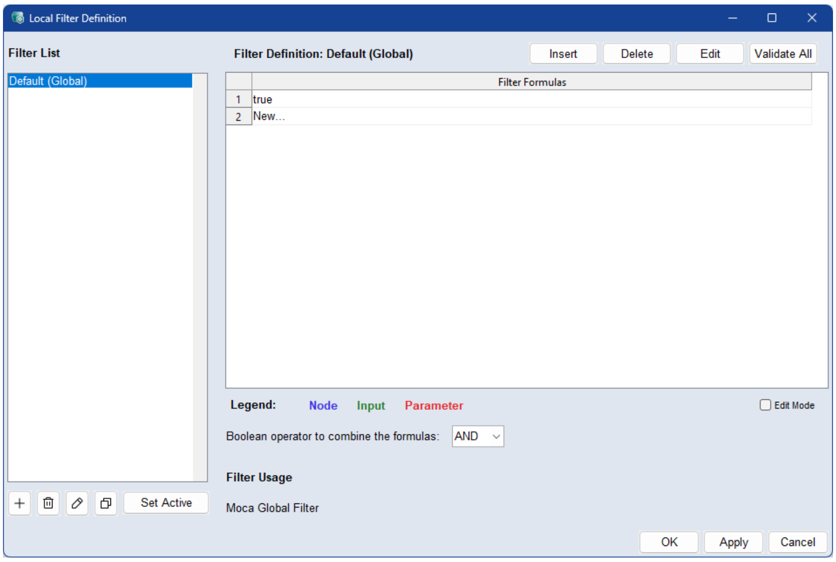
Filtering data

Note

The filter affects all data sets and only the views, not the optimization. To ignore data for the optimization, set the data weight to 0. For more information, see the MOCA Help.

1. Select **Analysis > Filter Data**.

   The Filter Data window opens.

2. To add a new filter, click + below the filter list.
3. In the pop-up window, type a name for the filter and confirm with **OK**.
4. To add a formula to the filter, click the **New** row.
5. The **Add Filter** window opens, where you can create a formula using the Data/Parameters/Nodes lists and the calculation buttons.

   Data points are shown only if the expression is true for a data point, e.g. %speed% > 2000 shows only points greater than 2000.
6. To validate the formula, click **Validate**.
7. If the formula is valid, click **OK**.
8. Select **AND** or **OR** from the drop-down list as the boolean operator to link the formulas.
9. To activate the filter as global or local filter, select it from the list and click **Set Active**.
10. Click **OK**.

Deleting a data point

1. Select a value in one or more rows.
2. Click the **Delete** button in the **Rows** area.

   A confirmation window opens.
3. In the confirmation window, click **Delete**.

   The selected rows are deleted.

Deleting an input column

1. Select a value in one or more columns.
2. Click the **Delete** button in the **Columns** area.

   A confirmation window opens.

3. In the confirmation window, click **Delete**.

   The selected columns are deleted.

6.4 Step 2: Data Analysis

For clearing up and evaluating the measuring data, you have the possibility to visualize it after the import graphically for anytime.

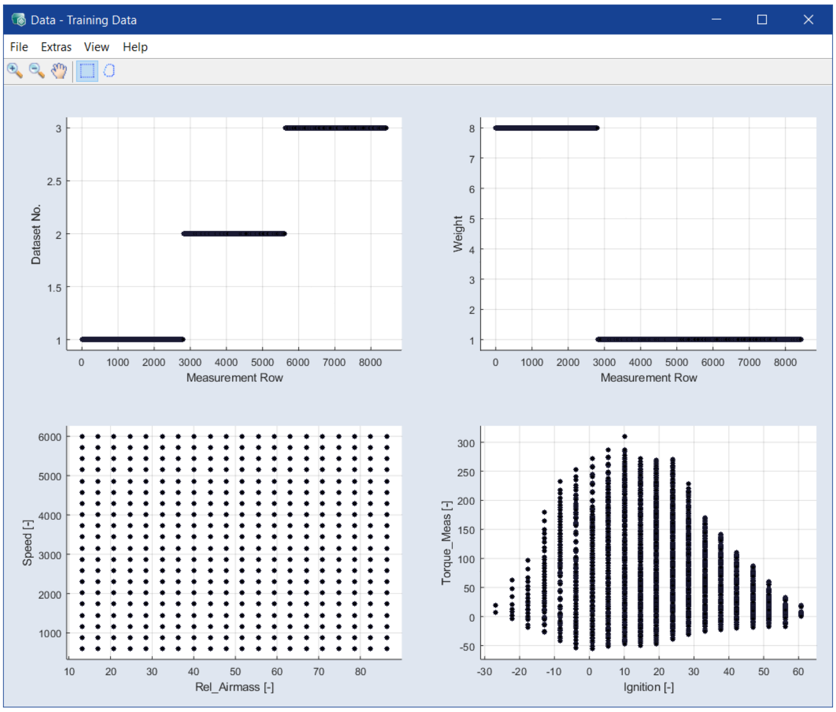
During the analysis, particular the following points should be considered.

- Have all parameters been varied according to the experiment plan and did the measured system remain in the operating mode intended for this purpose?
- Do the output variables fall in physically meaningful ranges?
- Are there any outliers included, which have to be removed, if appropriate?

Visualizing measurement data in a scatter plot

1. Select **Analysis › Scatter Plot › Training Data/Test Data/Training and Test Data › Data/Function Nodes**.

   The **Data** and - if your project contains function nodes – **Function Nodes** windows open. Only the **Data** window is used for the current task.



   The bottom-left plot (`Speed` over `Rel_Airmass`) shows the space-filling variation of the data in the experimental plan.

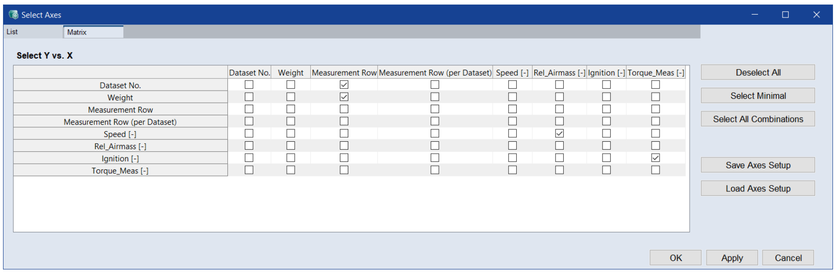
Changing the axis pairs

Since the current view does not show the dependence of the relevant measurement data, you must adjust the axis pairs. The selection of the displayed axes takes place directly in the plot window.

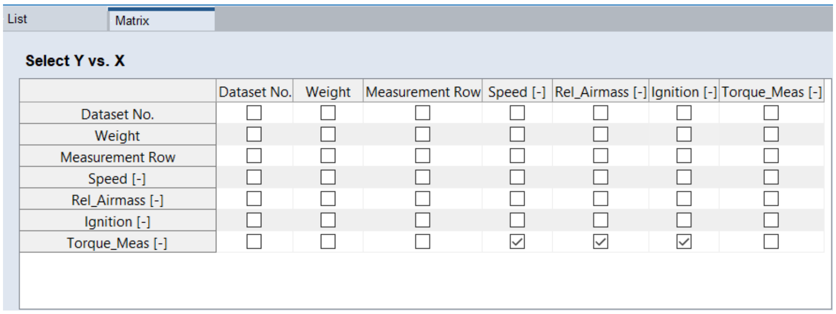
1. In the **Data** window of Scatter Plot, select **View › Select Axes**.

   The "Select Axes" window opens.

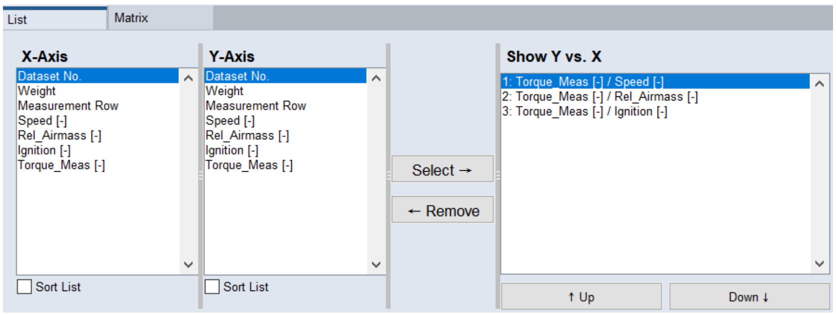
2. Go to the **Matrix** tab.



3. In the **Matrix** tab, select the axis pairs shown in the following figure.

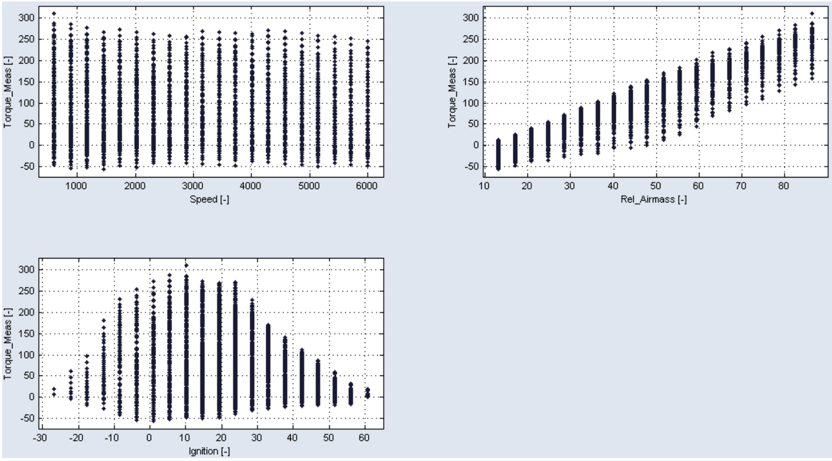


   In the **List** tab, you have an alternate representation of the selection of the axes pairs that are to be visualized.



4. Do one of the following:
   - Click **Apply**.
   - Click **OK**.

⇨ The visualized axes in the scatter plot will be adjusted. The **Select Axes** window remains open.



Deleting an outlier

You can now use the scatter plots to remove outliers.

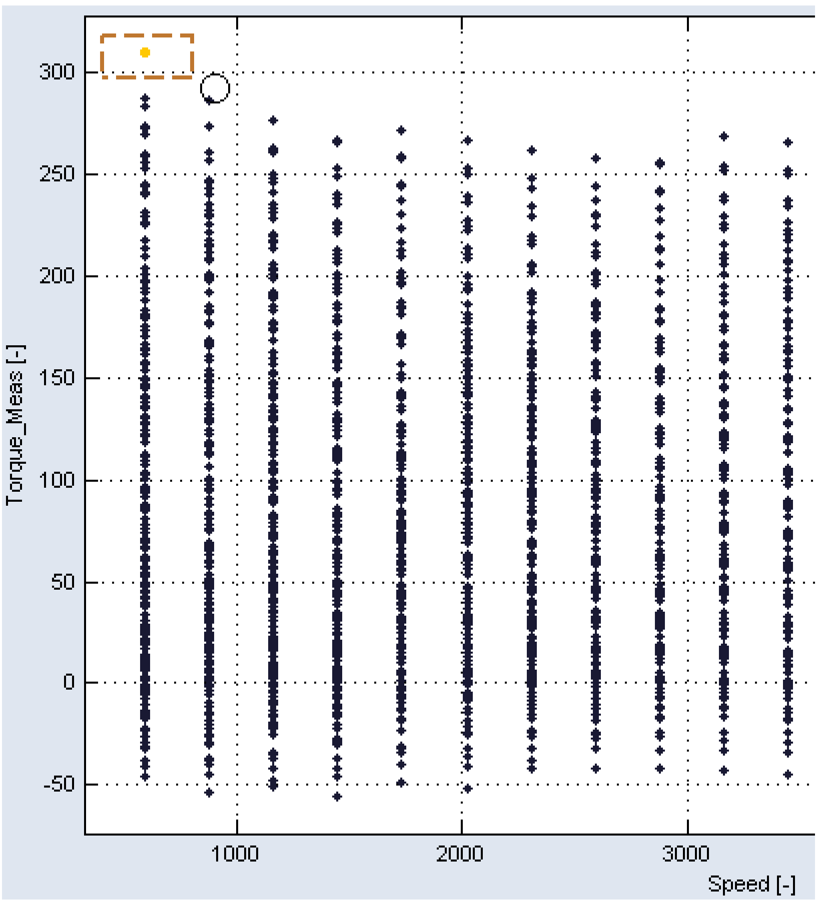
1. Search the plots for outliers.

   In this tutorial, use the top-left point in the **Torque_Meas over Speed** plot, with `Torque_Meas > 300`.

2. Click the **Mouse selection in Plot with Rectangle** button.

3. Click in the plot and draw a rectangle around the outlier.

   The selected points are colored in all scatter plots.



4. Right-click the frame of the rectangle and select **Mark Point** from the context menu.

   The marked points are highlighted with a red circle.

5. Select **Extras › Delete Marked Points**.

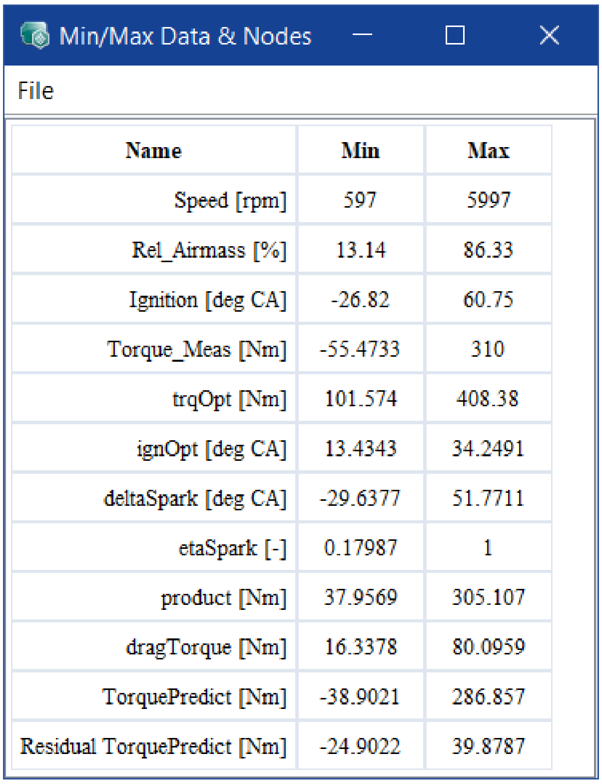
➪ The outlier is deleted from the measurement data.

Displaying the measurement data range

After deleting the outlier from the measurement data, you can check whether the values of the measurement file are within a plausible range.

1. In the main window, select **Analysis › Show Data Min/Max**.

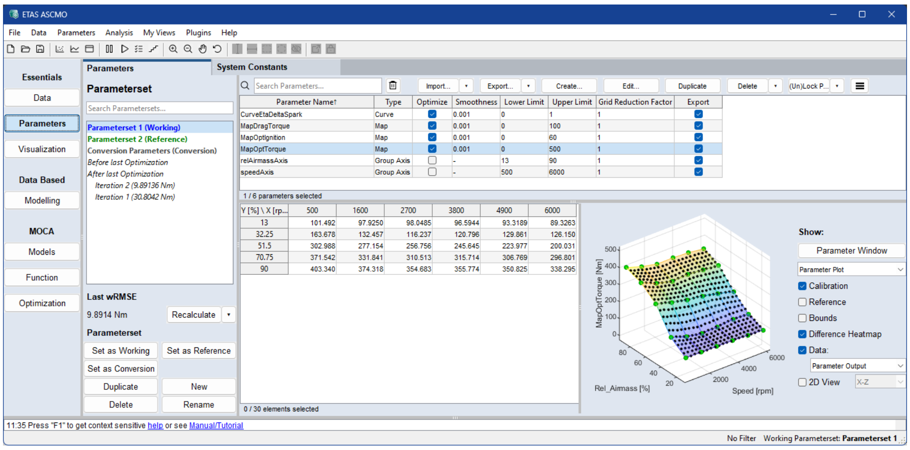
   The **Min/Max Data & Nodes** window opens.



Name	Min	Max
Speed [rpm]	597	5997
Rel_Airmass [%]	13.14	86.33
Ignition [deg CA]	-26.82	60.75
Torque_Meas [Nm]	-55.4733	310
trqOpt [Nm]	101.574	408.38
ignOpt [deg CA]	13.4343	34.2491
deltaSpark [deg CA]	-29.6377	51.7711
etaSpark [-]	0.17987	1
product [Nm]	37.9569	305.107
dragTorque [Nm]	16.3378	80.0959
TorquePredict [Nm]	-38.9021	286.857
Residual TorquePredict [Nm]	-24.9022	39.8787

After importing and reviewing the measurement data, you can start to add nodes to the function. To do so, select the **Function** working step in the navigation (see "Elements of the ASCMO-MOCA User Interface " on page 80). The respective elements in the main working window will appear, where you can save, delete and edit the function "Step 5: Build Up the Function" on page 115).

6.5 Step 3: Parameters



This step allows you to check and possibly adapt the parameters. Only the reference parameters will be visualized, but not the optimized parameters. You can set the optimized parameters as (new) reference in the Optimization step.

In addition, you can display the reference and the current map during the optimization and visualize the data points in maps and curves.

Also you have the possibility to fix individual grid points and to lock them for the optimization.

Further information about how to create and edit a parameter is given in "Step 5: Build Up the Function" on page 115.

Note

In this tutorial, the parameters are already defined, so you can skip this step and continue with the optimization (see "Step 6: Optimization" on page 122).

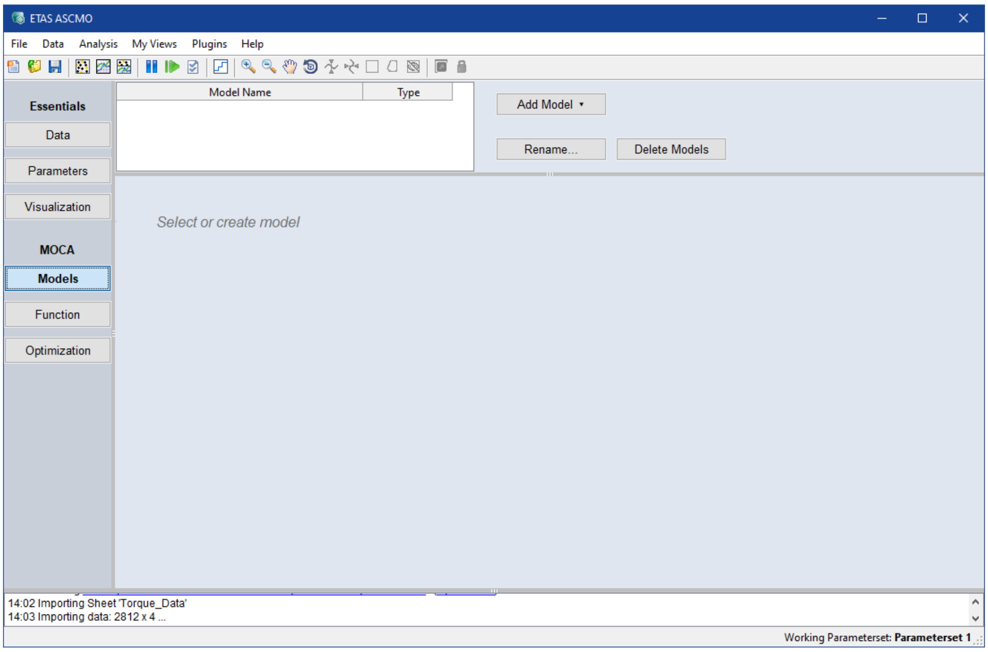
NOTICE

If you extend the data range and limits of the function parameters beyond the valid range of your system (e.g., a test bench), the system can become overloaded and damaged, when using the exported parameters in the system.

Always ensure that the limits and ranges in ASCMO-MOCA match the limits and ranges of your system before exporting the parameters.

If you want to perform a specific calibration and optimization task, these values are required knowledge.

6.6 Step 4: Models



In this step, you can link an existing Simulink® model with ASCMO-MOCA and prepare the mapping of the parameters, the inputs and outputs.

Note

You do not have to embed a model to ASCMO-MOCA as a part of this tutorial. Therefore, you can skip this step in the navigation and start to build the function "Step 5: Build Up the Function" on page 115.

This step requires a Simulink® installation. By default, ASCMO-MOCA will use the MATLAB® and Simulink® version most recently installed on your system.

Selecting a Simulink® version

1. In the main window, select **File › Options**.

   The **Options** window opens.
2. In the **Simulink Version** dropdown, select the version you want to use.
3. Click **OK** to apply your settings.

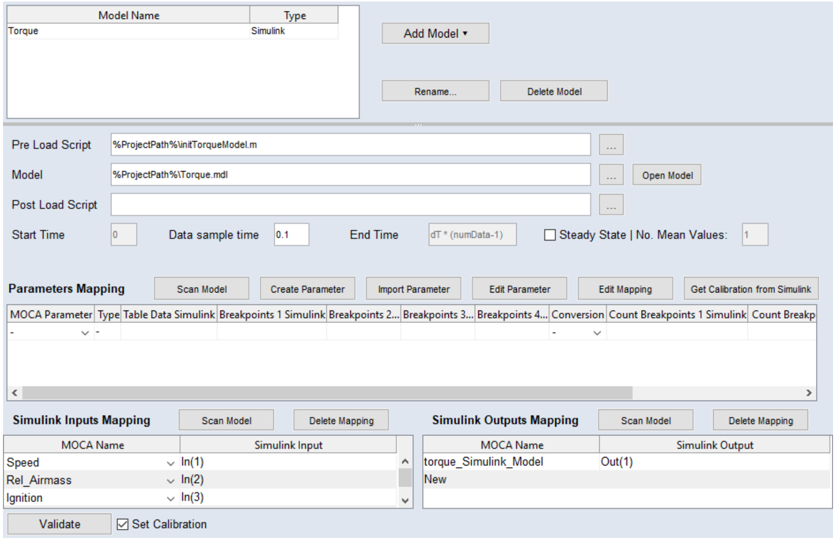
6.6.1 Adding A Simulink® Model and Scripts

You can add a Simulink model, which can be selected as node in "Step 5: Build Up the Function" on page 115. To add a Simulink model, proceed as follows:

Adding a Simulink® model and scripts

1. In the main working area, click **Add Model** and select **Connect to Simulink Model**.

   A new line is added to the model list; additional options are displayed in the lower part of the Model Step.



2. In the **Simulink Model** field, enter or select (via the ... button) path and name of the Simulink model to be optimized.

   This can be an `*.mdl` (before R2012a) or `*.slx` (from R2012a) Simulink model.

3. Press <ENTER> or click in another field.

   A warning opens if the Simulink model does not exist. Proceed as follows.

   i. Confirm the warning with **OK**.

   ii. Correct path and/or file name of the Simulink model.

4. If desired, enter or select (via the ... button) path and name of an executable M-script in the **Pre Load Script** field.

   This Init script is optional and may contain a pre-calibration for the Simulink model. The pre-calibration is expected as a MATLAB workspace variable that will be assigned in the **Parameters Mapping** table.

5. If desired, enter or select (via the ... button) path and name of another executable M-script in the **Post Load Script** field.

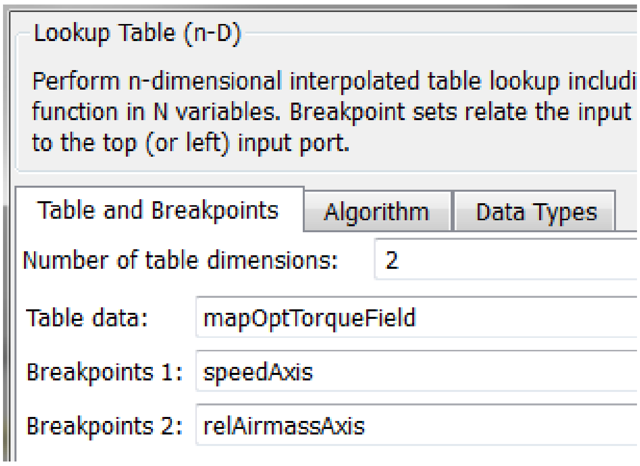
   This script is optional.

> **Note**
>
> You can use `%ProjectPath%` as the location. This is then automatically replaced by the current location of the ASCMO-MOCA project.

6.6.2 Mapping Simulink® Parameters

In the **Parameters Mapping** area, project-related maps/curves and scalars can be mapped to the parameters from the Simulink model.

ASCMO-MOCA expects the parameters to be calibrated as MATLAB® workspace variables. This could for example be the following Simulink map:



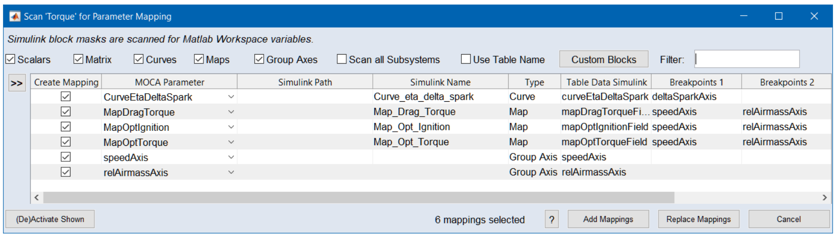
You can map project-related parameters and Simulink parameters either automatically (see "Scanning the Simulink® model and mapping parameters" below) or manually (see "Mapping parameters manually" on the next page).

Scanning the Simulink® model and mapping parameters

To automatically scan the Simulink model for possible parameters, proceed as follows.

1. In the main window, **Parameters Mapping** area, click **Scan Model**.

   The block masks of lookup tables and other blocks are scanned for variable names. The results are then presented in the **Scan Model ‹model_name› for Parameter Mapping** window.

2. Activate/deactivate the **Scalars**, **Matrix**, **Curves**, **Maps** and/or **Group Axes** checkboxes to show/hide parameters of the respective types.
3. In the **Filter** field, enter the string by which you want to filter the list, then press ‹ENTER›.

   > **Note**
   >
   > The filter is not caps-sensitive. You cannot use wildcards.

   Only parameters whose Simulink names or paths contain the search string are displayed.
4. In the **Create Mapping** column, activate the checkboxes in all rows you want to map.

5. Click **Add Mappings** to add the selected mappings to the **Parameters Mapping** list.

   With **Add Mappings**, existing content in the **Parameters Mapping** list is kept. If an existing row is selected again, this selection is ignored and a message is issued in the log window.

   Replace Mappings removes existing content in the **Parameters Mapping** list.

<u>Mapping parameters manually</u>

1. In the **Parameters Mapping** area, **MOCA Parameter** column, select a parameter from the dropdown list.

   You can also create a new parameter with the **Create Parameter** button or import a DCM file with the **Import Parameter** button.

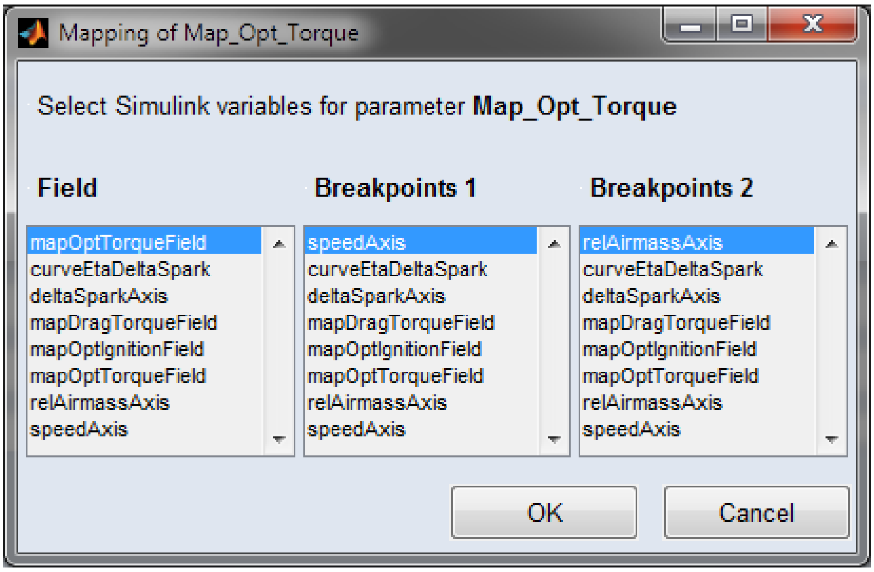
2. In the **Table Data Simulink** and **Breakpoints *‹n›* Simulink** columns, enter the variable names from the block mask.

Or - as an alternative -

3. Proceed as follows.

   i. Click **Edit Mapping**.

      The **Mapping of *‹parameter›*** window opens. The init script is executed first and the model is loaded. The existing MATLAB workspace variables are displayed. ASCMO-MOCA automatically performs a name search.



   ii. In the **Field** column, select the MATLAB workspace variable that describes the parameter value or table data.

   iii. In the **Breakpoint *‹n›*** columns, select MATLAB workspace variables used for the table axes.

   iv. Click **OK** to accept your settings and close the **Mapping of *‹parameter›*** window.

The **Table Data Simulink** and **Breakpoint ‹n› Simulink** columns in the **Parameters Mapping** area are updated according to your selections.

6.6.3 Mapping Simulink® Inputs

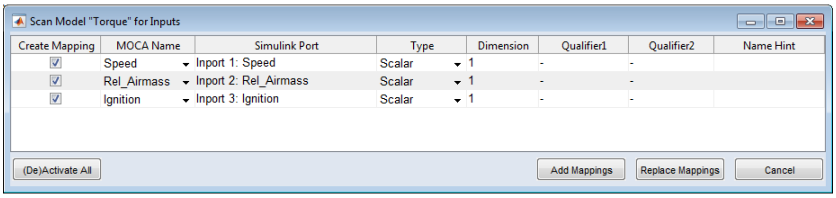
In the **Simulink Inputs Mapping** area, imported data columns or nodes from the ASCMO-MOCA project can be mapped to the Simulink model inputs.

<u>Scanning the Simulink® model and mapping inputs</u>

To automatically scan the Simulink model for inputs, proceed as follows:

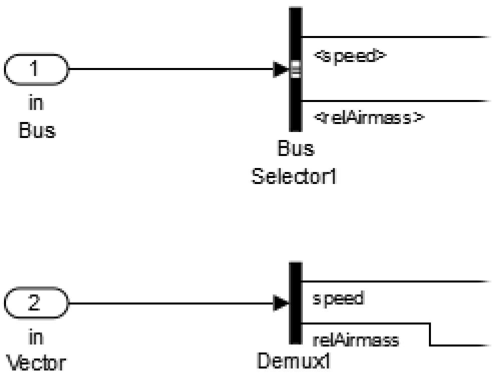
1. In the **Simulink Inputs Mapping** area, click **Scan Model**.

   The model is scanned for Inport and From Workspace blocks. The results are then presented in the **Scan Model ‹model_name› for Inputs** window.

2. If necessary, enter the dimension and qualifiers manually.

   Consider, for example, the following input ports, which expect a bus and a vector.

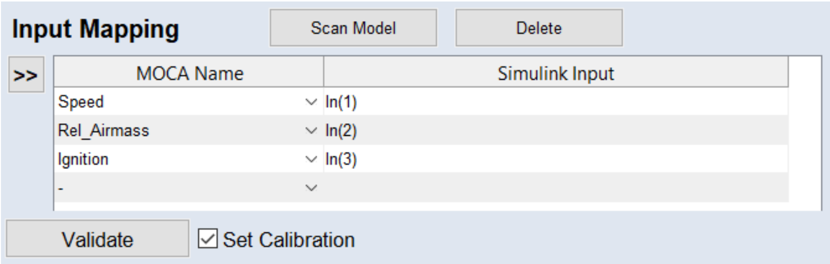
   ASCMO-MOCA automatically tries to identify such a bus or vector signal, by following the signal flow in Simulink. If this fails, you have to manually enter the type, dimension and qualifier.

3. In the **Create Mapping** column, activate the checkboxes in all rows you want to map.
4. Click **Add Mappings** to add the selected mappings to the **Simulink Inputs Mapping** list.

   With **Add Mappings**, existing content in the **Simulink Inputs Mapping** list is kept. If an existing row is selected again, this selection is ignored and a message is issued in the log window.

Replace Mappings removes existing content in the **Simulink Inputs Mapping** list.

After clicking **Add/Replace Mappings**, the **Simulink Inputs Mapping** table is filled.



The notation `Speed | In(1)` means that the first **Simulink Input data** column is passed as **Speed**.

A list of possible notations is given in the online help.

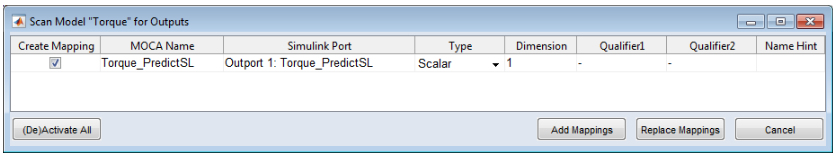
6.6.4 Mapping Simulink® Outputs

In the Simulink Outputs Mapping area, the simulation outputs are made available to ASCMO-MOCA. Outport and ToWorkspace blocks are supported.

<u>Scanning the Simulink® model and mapping outputs</u>

1. In the **Simulink Outputs Mapping** area, click **Scan Model**.

   The model is scanned for Outport and ToWorkspace blocks. The results are then presented in the **Scan Model ‹model_name› for Outputs** window.

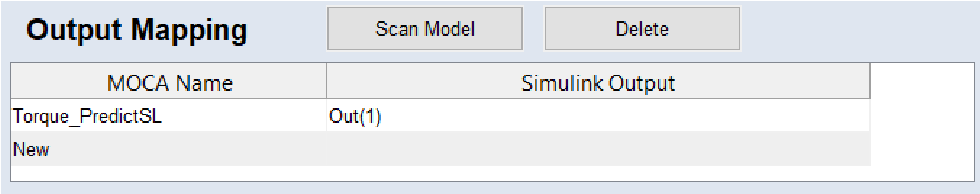
2. If necessary, enter the dimension and qualifiers manually.
3. In the **Create Mapping** column, activate the checkboxes in all rows you want to map.
4. Click **Add Mappings** to add the selected mappings to the **Simulink Outputs Mapping** list.

   With **Add Mappings**, existing content in the **Simulink Outputs Mapping** list is kept. If an existing row is selected again, this selection is ignored and a message is issued in the log window.

   Replace Mappings removes existing content in the **Simulink Outputs Mapping** list.

Example

`Torque_PredictSL | Out(1)` makes the Simulink output available for optimization under the name `Torque_PredictSL`.



A list of possible notations is given in the online help.

6.6.5 Validating and Using the Simulink®Model

> **Note**
>
> Running a Simulink® model is only possible if a suitable Simulink® version with corresponding license is available on your system.

After mapping parameters, inputs and outputs, perform the following steps:

6.6.5.1 Validating a Simulink®Model

Validating a Simulink® Model

1. In the lower section of the main working area, click **Validate**.

   The validation is performed.

During validation, the following steps are executed:

- Start the (optional) init script.
- Add the model path to the MATLAB search path.
- Open the Simulink model.
- Start the (optional) post-load script.
- Check if all parameters are available as workspace variables.
- Replace the in/out ports in accordance to the In/Out mapping.
- Start a simulation with a subset size of the data.
- Read the output values.

Possible errors are reported. If the test is successful, a success message is displayed. The Simulink model is now ready for optimization.

Before you can use a model for the optimization, you need to make it available in the function.

Using a Simulink® model

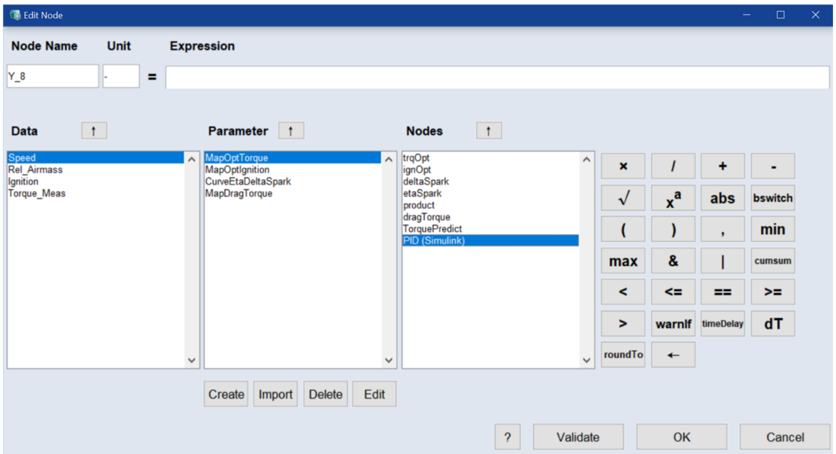
Before you can use a model for the optimization, you need to make it available in the function. Proceed as follows:

1. In the navigation pane, click **Function**.

   The Function pane opens.

2. Add a new node (see the online help for details).

The Simulink models are available in the **Nodes** area of the **Edit Node** window.



3. Insert the model in the expression.

   The expression is set to `%Torque%(Speed, Rel_Airmass, Ignition)`. In addition, the name `Torque_mdl` is entered in the **Node Name** field.

4. If desired, validate the node.

5. Click **OK**.

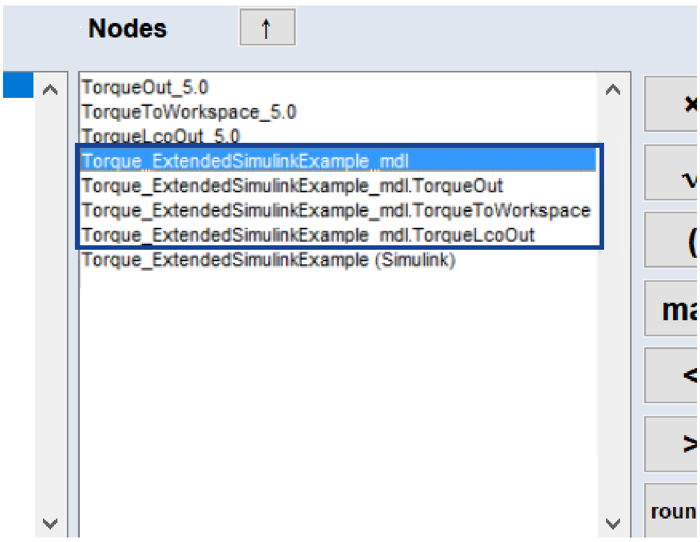
   The node for the model is added to the **Main Function Nodes** table.

   For the model output, another node named `Torque_mdl.torque_Simulink_Model` is created.

6. If desired, create more function nodes.

   See "Step 5: Build Up the Function" on page 115 for more information.

If you are using a Simulink model with multiple outputs, one function node is created for each model output. These nodes can be used in optimization criteria and expressions.



The default notation used in the **Main Function Nodes** table is marked in Fig. 6-1. The notation used in previous versions of ASCMO-MOCA (nodes `Torque*_5.0` in Fig. 6-1) remains valid.

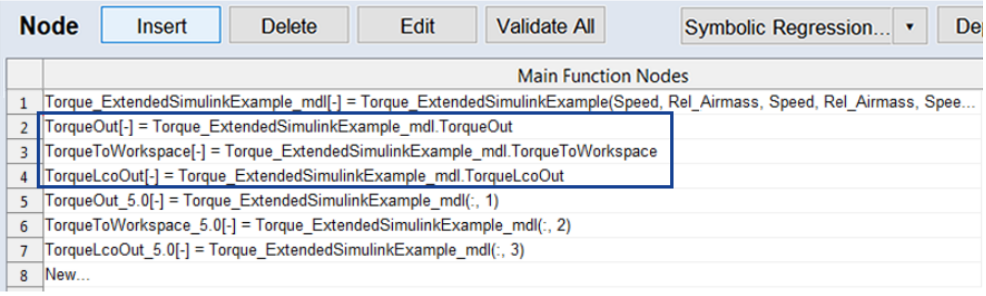


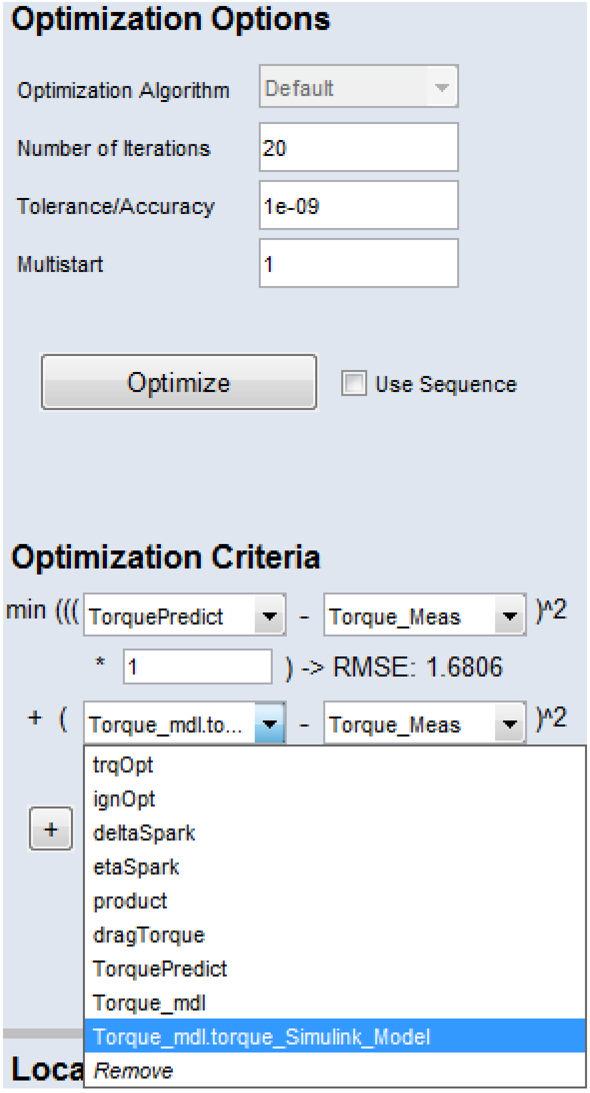
Fig. 6-1: Notation for a Simulink model with multiple outputs

Note

An implementation of ASCMO-MOCA using a Simulink® model with several outputs can be found in the example project `Torque_ExtendedSimulinkExample.moca` in *<installation>*`\Example\Moca` directory. By default, *<installation>* `= C:\Program Files\ETAS\ASCMO x.x`.

Optimizing a Simulink® model

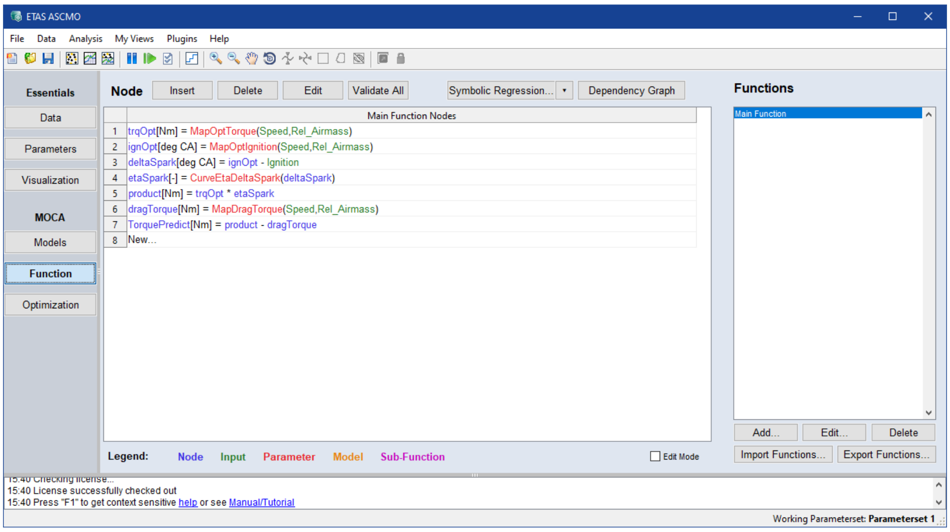
1. In the navigation pane, click **Optimization**.
2. Under **Optimization Criteria**, select the optimization criteria that are based on the outputs.



3. Click the **Optimize** button.

   The optimization of the Simulink model is started. Information about the optimization can be found in the Log window (for example on iterations, RMSE). The resulting RMSE is shown below each optimization criterion.

6.7 Step 5: Build Up the Function



After reading the measuring data and checking the plausibility, you can start to set up the function for the torque sensor that will be modeled during the tutorial. The available operators are described in section " Mathematical Operators for Function Nodes" on page 43.

> ***NOTICE***
>
> If you extend the data range and limits of the function parameters beyond the valid range of your system (e.g., a test bench), the system can become overloaded and damaged, when using the exported parameters in the system.
>
> Always ensure that the limits and ranges in ASCMO-MOCA match the limits and ranges of your system before exporting the parameters.
>
> If you want to perform a specific calibration and optimization task, these values are required knowledge.

6.7.1 Modeling the Function

In the tutorial you will build the following function of the physical **engine torque** model.

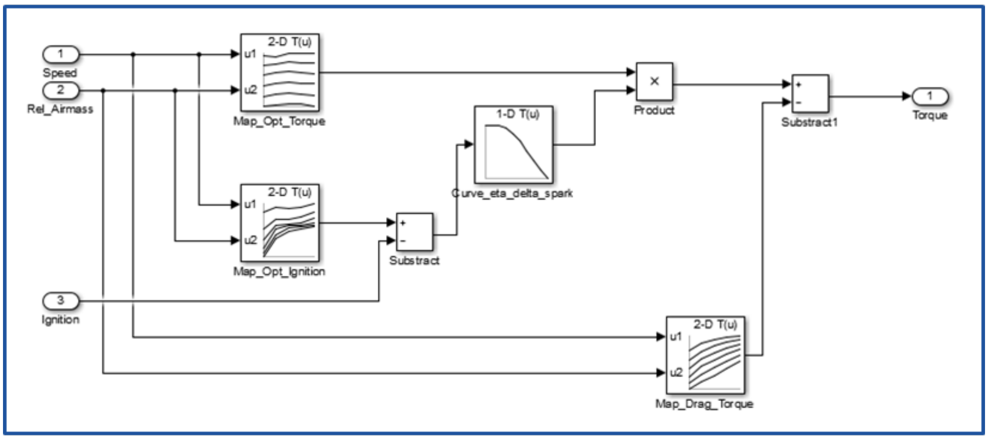


Fig. 6-2: Structure of the function to be modeled

> **Note**
>
> Functions are always set up from left to right.

The model function shown in "Structure of the function to be modeled" above contains the following inputs:

- **1** - `Speed`
- **2** - `Rel_Airmass`
- **3** - `Ignition`

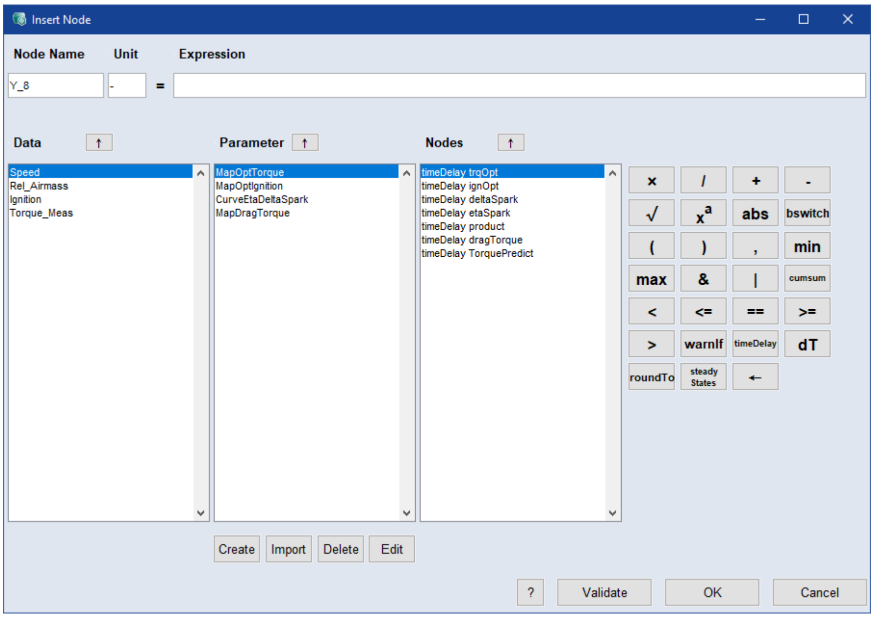
In addition to the inputs, you have imported the measured model output `Torque_Meas` in "Step 1: Data Import" on page 88. These values will be used as reference for the optimization, for minimizing the deviation between the measured values and the function prediction `TorquePredict`.

Adding the first node

To insert the first node `trqOpt`, proceed as follows.

1. Do one of the following:
   - In the **Main Function Nodes** table, click the `New` entry.
   - Click **Insert**.

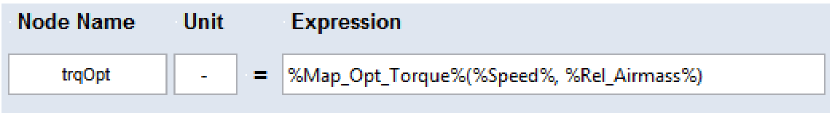
   The **Insert Node** window opens. All data channels you imported are listed in the **Data** area.



2. In the **Node Name** field, enter the name `trqOpt`.
3. If desired, enter a unit.

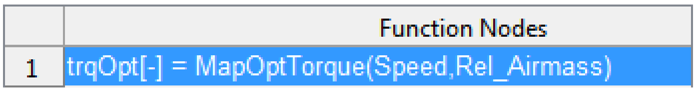
   The unit has no influence on the calibration of the parameter and is only visualized for support.
4. Create a parameter `MapOptTorque` as described in "Creating a new parameter" below.
5. Specify the expression for the function node.
   i. In the **Parameter** area, select the parameter `MapOptTorque`.
   ii. Click the  button.

      The parameter is added to the **Expression** field.

   iii. Click **Validate** to check the validity of the new node.
6. Click **OK** to add the node and close the **Edit Node** window.

➪ The node is displayed in the first row of the **Function Nodes** table.

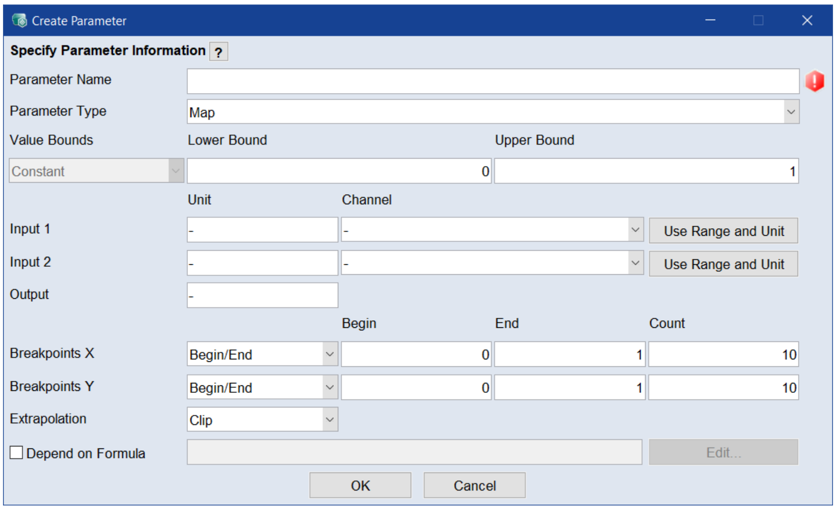


Creating a new parameter

To create the `MapOptTorque` parameter, proceed as follows.

1. In the **Edit Note** window, click the **Create** button below the **Parameter** area.

   The **Create Parameter** window opens.



2. Enter the parameter information.

   For `MapOptTorque`, use the following values:

Parameter Name:	`MapOptTorque`
Parameter Type:	`Map`
Value Bounds:	`0 - 500`
Input 1:	`Speed`
Input 2:	`Rel_Airmass`
Breakpoints X:	`Begin/End`; Begin = `500`; End = `6000`; Count = 6
Breakpoints Y:	`Begin/End`; Begin = `10`; End = `90`; Count = 6
Extrapolation:	`Clip`

> **Note**
>
> If you click **Use Range**, the values range of the X and Y axes are automatically set to the minimal and maximal value of the channel.

3. Click **OK**.

   The parameter is created. It appears in the **Parameter** area.

Next, you create and set up the node `ignOpt`.

Adding and editing the node **`ignOpt`**

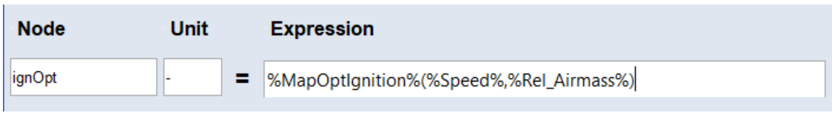
To add and edit the second node `ignOpt`, proceed as follows:

1. Open the **Edit Node** window.
2. Enter the node name `ignOpt`.

   For this node you need a parameter `MapOptIgnition`.
3. Create the parameter `MapOptIgnition` (see "Creating a new parameter" on the previous page) with the following values:

Parameter Name:	`MapOptIgnition`
Parameter Type:	`Map`
Value Bounds:	`0 - 60`
Input 1:	`Speed`
Input 2:	`Rel_Airmass`
Breakpoints X:	`Begin/End`; Begin = `500`; End = `6000`; Count = 6
Breakpoints Y:	`Begin/End`; Begin = `10`; End = `90`; Count = 6
Extrapolation:	`Clip`

4. Specify the following expression for the function node:



5. Check the validity of the new node.
6. Click **OK** to add the node and close the **Edit Node** window.

Next, you create and set up the node `deltaSpark`.

Adding and editing the node **`deltaSpark`**

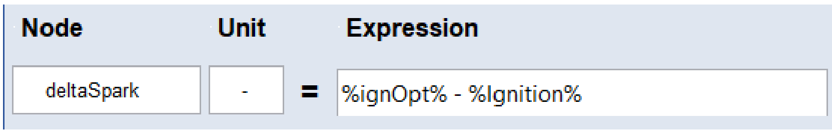
To add and edit the third node `deltaSpark`, proceed as follows:

1. Open the **Edit Node** window.
2. Enter the node name `deltaSpark`.

   For this node you need the node `ignOpt` and the input `Ignition`.
3. In the **Nodes** area, select `ignOpt` and click ↑.

   The `ignOpt` node is added to the **Expression** field.
4. Click **-** to add a subtraction operator.
5. In the **Data** area, select `Ignition` and click ↑.

   The `Ignition` channel is added to the **Expression** field.
6. Make sure that the expression looks as follows:



7. Check the validity of the new node.
8. Click **OK** to add the node and close the **Edit Node** window.

Next, you create and set up the node `etaSpark`.

Adding and editing the node **`etaSpark`**

To add and edit the fourth node `etaSpark`, proceed as follows:

1. Open the **Edit Node** window.
2. Enter the node name `etaSpark`.

   For this node you need a parameter `CurveEtaDeltaSpark`.
3. Create the parameter `CurveEtaDeltaSpark` (see "Creating a new parameter" on page 117) with the following values:

Parameter Name:	`CurveEtaDeltaSpark`
Parameter Type:	`Curve`
Value Bounds:	`0 - 1`
Input 1:	`deltaSpark`
Breakpoints X:	`Begin/End`; Begin = `-10`; End = `55`; Count = `10`
Extrapolation:	`Clip`

4. Specify the following expression for the function node:

Node	Unit		Expression
etaSpark	-	=	%CurveEtaDeltaSpark%(%deltaSpark%)

5. Check the validity of the new node.
6. Click **OK** to add the node and close the **Edit Node** window.

Next, you create and set up the node `product`.

<u>Adding and editing the node **`product`**</u>

To add and edit the fifth node `product`, proceed as follows:

1. Open the **Edit Node** window.
2. Enter the node name `product`.

   For this node you need the nodes `trqOpt` and `etaSpark`.
3. In the **Node** area, select `trqOpt` and click ↑.

   The `trqOpt` node is added to the **Expression** field.
4. Click × to add a multiplication operator.
5. Add the `etaSpark` node to the expression.
6. Make sure that the expression looks as follows:

Node	Unit		Expression
product	-	=	%trqOpt% .* %etaSpark%

7. Check the validity of the new node.
8. Click **OK** to add the node and close the **Edit Node** window.

Next, you create and set up the node `dragTorque`.

Adding and editing the node **`dragTorque`**

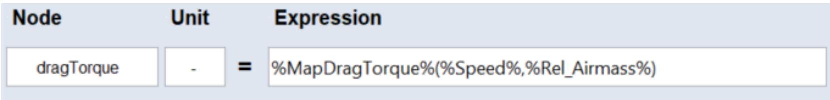
To add and edit the sixth node `dragTorque`, proceed as follows:

1. Open the **Edit Node** window.
2. Enter the node name `dragTorque`.

   For this node you need a parameter `MapDragTorque`.
3. Create the parameter `MapDragTorque` (see "Creating a new parameter" on page 117) with the following values:

Parameter Name:	`MapDragTorque`
Parameter Type:	`Map`
Value Bounds:	`0 - 100`
Input 1:	`Speed`
Input 2:	`Rel_Airmass`
Breakpoints X:	`Begin/End`; Begin = `500`; End = `6000`; Count = 6
Breakpoints Y:	`Begin/End`; Begin = `10`; End = `90`; Count = 6
Extrapolation:	`Clip`

4. Specify the following expression for the function node:



5. Check the validity of the new node.
6. Click **OK** to add the node and close the **Edit Node** window.

Next, you create and set up the node `TorquePredict`.

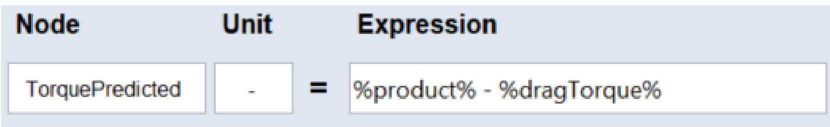
Adding and editing the node **`TorquePredict`**

To add and edit the last node `TorquePredict`, proceed as follows:

1. Open the **Edit Node** window.
2. Enter the node name `TorquePredict`.

   For this node you need the nodes `product` and `dragTorque`.
3. In the **Node** area, select `product` and click .

   The `product` node is added to the **Expression** field.
4. Add a subtraction operator.
5. Add the `dragTorque` node to the expression.
6. Make sure that the expression looks as follows:



7. Check the validity of the new node.

8. Click **OK** to add the node and close the **Edit Node** window.

   After adding the node `TorquePredict`, the creation of the function to be optimized as a representation of the physical torque model is finished. The **Function Nodes** table should look like this:

	Function Nodes
1	trqOpt[-] = Map_Opt_Torque(Speed, Rel_Airmass)
2	ignOpt[-] = MapOptIgnition(Speed, Rel_Airmass)
3	deltaSpark[-] = ignOpt - Ignition
4	etaSpark[-] = CurveEtaDeltaSpark(deltaSpark)
5	product[-] = trqOpt * etaSpark
6	dragTorque[-] = MapDragTorque(Speed, Rel_Airmass)
7	TorquePredict[-] = product - dragTorque
8	New...

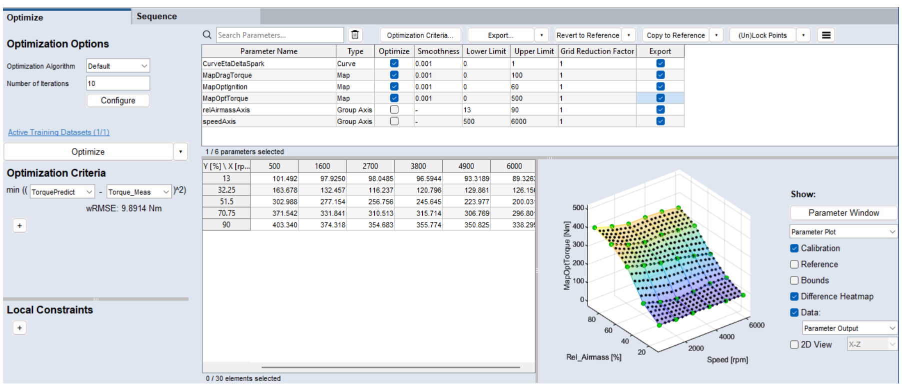
Note

If you activate the **Edit Mode** option in the main working window, you can change the elements of the function directly in the **Function Nodes** table. The names of data, parameters and nodes are marked with %.

	Function Nodes
1	trqOpt[-] = %Map_Opt_Torque%(%Speed%, %Rel_Airmass%)

In the next step "Step 3: Parameters" on page 104 you have the possibility to check and edit the created parameters, if appropriate.

6.8 Step 6: Optimization



Before you start optimizing, you need to choose an optimization algorithm. To help you choose the best algorithm for your purpose, see "Optimization Algorithms" on page 52.

After you choose an algorithm, click **Optimizer Options** **Configure** to customize the options. For a description of the options available for each algorithm, see "Optimizer Options" on page 56.

> **Note**
>
> The above listed criteria and limits have to be adapted for the specific problem, such that a satisfactory minimal deviation (see "Variables RMSE and R2" on page 25) between the measured data and the function prediction can be reached by optimization.

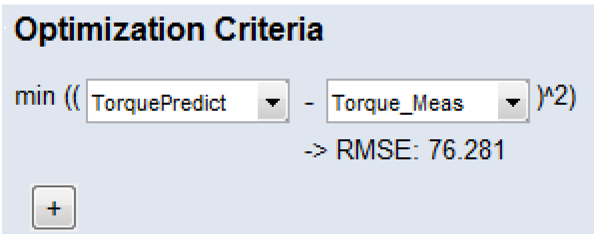
In the **Optimization Options** area, you specify several parameters, see 6.8 "Step 6: Optimization" on the previous page.

In the **Optimization Criteria** area, you specify the optimization target. Select a function node in the first dropdown and a data channel (or '0') in the second dropdown. The optimizer then tries to find a set of parameter values that minimizes the quadratic deviation of these two quantities.

The **Local Constraints** area is not used in this tutorial to keep the example in the tutorial simple. Depending on the optimization problem, these constraints can be used to guide the optimization in a specific direction.

Preparing the optimization

1. In the **Optimization Options** area, do the following:
   i. In the **Number of Iterations** field, enter a number of 40 iterations.
   ii. Increase the number of **Multistart** (**Optimizer Options** **Configure**) to prevent the optimizer from getting stuck in a local minimum.
2. In the **Optimization Criteria** area, do the following:

   i. From the first drop-down list, select the function output **TorquePredict**.
   ii. From the second drop-down list, select the imported data channel **Torque_Meas**.

      The flat parameters results in a high RMSE (see "Variables RMSE and R2" on page 25). After the first optimization, the RMSE will be significantly reduced.

   You have the possibility to define a sum of such optimization criteria using the + **Add a new Optimization criterion** button.

Performing the optimization

NOTICE

Damage due to wrong calibration data

Wrong usage of calibrations derived from ASCMO-MOCA model can lead to engine or test bench damage.

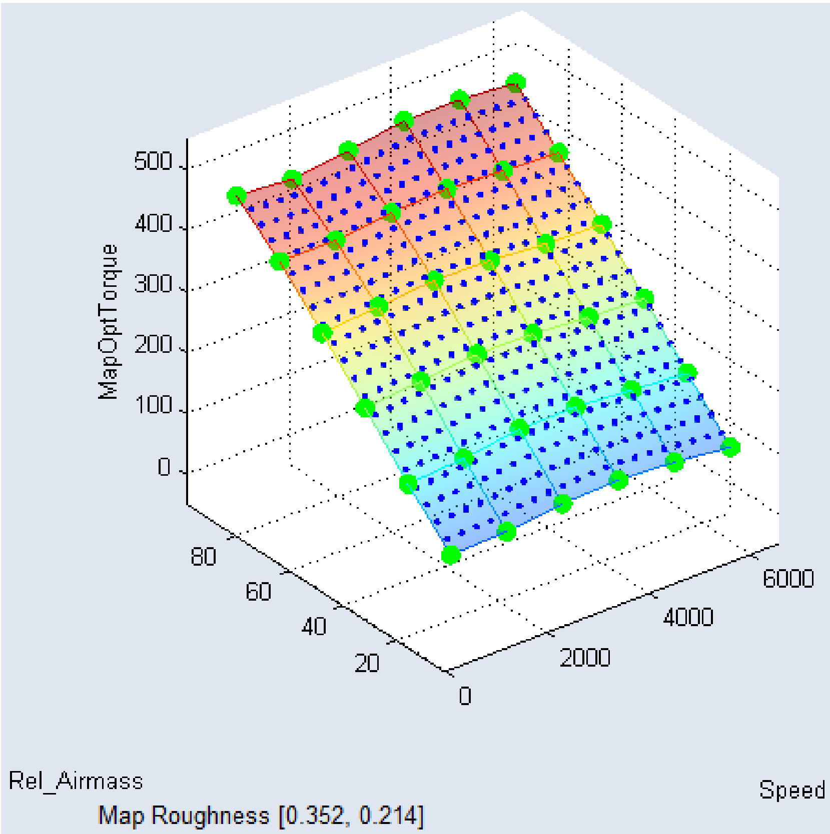
Compare measured data and model created data with Residual Analysis feature after the optimization or before exporting at the latest. Feature is accessible via Analysis > Residual Analysis > Training and Test Data > Absolute Error Analysis.

See "Performing the optimization" above, export options in **Parameters** Step or **Optimization** Step, and 6.9 "Step 7: Export" on the next page.

Once you have finished the preparations, start the optimization.

1. Click **Optimize**.

   The optimizer starts optimizing the parameters. Information about the optimization can be found in the log window (e.g., iterations, RMSE). The resulting RMSE is displayed below the optimization criterion. The visualization of the maps is adjusted accordingly.

   The optimization in this tutorial is now completed. Some optional activities in the **Optimization** step are described in the online help, section "Instructions (Optimization Step)".

In the following step (see "Step 7: Export" below), the optimized parameters will be exported for further processing.

6.9 Step 7: Export

> ***NOTICE***
>
> **Damage due to wrong calibration data**
>
> Wrong usage of calibrations derived from ASCMO-MOCA model can lead to engine or test bench damage.
>
> Compare measured data and model created data with Residual Analysis feature after the optimization or before exporting at the latest. Feature is accessible via Analysis > Residual Analysis > Training and Test Data > Absolute Error Analysis.

See "Performing the optimization" on the previous page, export options in **Parameters** Step or **Optimization** Step, and 6.9 "Step 7: Export" above.

In this step, you will export the created and optimized parameters. The parameters can be exported in several file formats, and the project can be saved for the runtime environment ASCMO-MOCA Runtime with limited functionality.

Exporting the parameters

1. In the **Optimization** step, click the **Export** button.

   The **Export Parameters** window opens.
2. In that window, enter or select the file path and file name for the export.
3. From the **Save as type** drop-down list, select the export format.
4. Click **Save**.

⇨ The parameters are exported to a file in the selected format.

Exporting selected parameters

1. In the **Optimization** or **Parameters** step, activate the checkboxes in the **Export** column for the parameters you want to export.

   You can also select multiple parameters and use the context menu to set or clear the export flags.

   Corresponding group axes related to selected parameters will automatically be exported.
2. Click **Export**.

   The **Export Parameters** window opens.
3. In that window, enter or select the file path and file name for the export.
4. In the **Save as type** drop-down list, select the export format.
5. Click **Save**.

⇨ The selected parameters are exported to a file in the selected format.

Exporting the project for ASCMO-MOCA Runtime

When you export a project to ASCMO-MOCA Runtime, the project gets encrypted, and the function in the project cannot be seen or edited.

1. In the main menu, select **File › Export to MOCA-Runtime**.

   The **Export to MOCA-Runtime** window opens.

2. Activate **Show Sequence** if you want to show the optimization sequence in the exported project.

   With that, you can see and edit the sequence in ASCMO-MOCA Runtime. The sequence is hidden if **Show Sequence** remains deactivated.

3. Activate **Allow opening in MATLAB (MOCA-Runtime p-Code)** to allow the exported project to be opened in the p-Code version of ASCMO-MOCA Runtime.

   If you activate **Allow opening in MATLAB (MOCA Runtime p-Code)**, the exported project gets encrypted with another key. ASCMO-MOCA Runtime p-Code cannot open exported projects without this setting. Such projects can only be opened with the standalone version of ASCMO-MOCA Runtime.

4. Click **Export** to continue.

   The **Export MOCA project to MOCA-Runtime** window opens. The `*.moca_runtime` format is preselected; it is mandatory.

5. Enter the file name and the file directory.
6. Click **Save**.

   The project is saved for ASCMO-MOCA Runtime.

Glossary

F

Function

A function is the set of all elements required to represent the physical model.

P

Project

Creation and optimization of functions and parameters occur in the context of a project. This project can be saved and loaded. One project at a time can be opned and edited in one instance of ASCMO-MOCA.

R

Residual

The residual is the discrepancy between measured data and function calculation. ASCMO-MOCA distinguishes between relative, absolute, and studentized error.

Residuum

The residual is the discrepancy between measured data and function calculation. ASCMO-MOCA distinguishes between relative, absolute, and studentized error.

RMSE

The root mean square error (RMSE) is a measure of the deviation of the predictions of a model from the actual values of the modeled object. The single deviation is called residuum.

Root Mean Square Error

The root mean square error (RMSE) is a measure of the deviation of the predictions of a model from the actual values of the modeled object. The single deviation is called residuum.

Roughness

Roughness describes the change in slope from one grid point of a curve, pap, or cube to the next.

S

Scalar

A scalar is a 0-dimentional calibration parameter.

System Constant

A system constant is a container for an element that cannot be changed. The counterpart o a system constant is a variable.

T

Tolerance

Generally, a tolerance is a threshold which, if crossed, stops the iterations of a solver.

Figures

Equations

Index

R

S

T

U

V

W

8 Contact Information

Technical Support

For details of your local sales office as well as your local technical support team and product hotlines, take a look at the ETAS website:

www.etas.com/hotlines

ETAS offers trainings for its products:

www.etas.com/academy

ETAS Headquarters

ETAS GmbH

Borsigstraße 24	Phone:	+49 711 3423-0
70469 Stuttgart	Fax:	+49 711 3423-2106
Germany	Internet:	www.etas.com