



Pointing Error Engineering For Telecommunication Missions

ESA Contract No. 4000123466/18/UK/ND

Technical Officer: Bénédicte Girouart (TEC-SAA)

Software User Manual

| | | | |
|-----------------|---------------------|--|--------------|
| Document | ASTOS-P4COM-SUM-001 | | Date: |
| Issue: | 1.3 | | 2022-02-04 |

| | Name/Function | Organization | Signature | |
|----------------------------|----------------------|---------------------|------------------|------------|
| Prepared by: | Marc Hirth | Astos Solutions | | 2022-02-04 |
| | | | | |
| | | | | |
| Sent to: | Bénédicte Girouart | ESA/ESTEC | | |
| | Nicolas Deslaef | ESA/ESTEC | | |
| Product Assurance: | | | | |
| Project Management: | Marc Hirth | Astos Solutions | | 2022-02-04 |

Document Change Record

| Issue | Date | Affected Chapter/Section/Page | Reason for Change Brief Description of Change |
|-------------|------------|--------------------------------|---|
| 1.0 Draft | 2019-11-25 | all | Preliminary draft update (no updated screenshots and figures) |
| 1.0 Draft 2 | 2020-02-20 | all | Typos and phrasing corrected, adding missing temporal truncated Gaussian description, updated analysis features |
| 1.0 Draft 3 | 2020-04-28 | all | Updated screenshots and fixed several typos; additional keyboard shortcuts |
| 1.0 Draft 4 | 2020-07-07 | 7.10, 10.9 | Updated sections for GPE block |
| 1.0 Draft 5 | 2020-08-19 | 9.2, 9.7 | Updates according to TR1 RIDs |
| 1.0 | 2020-11-11 | 9.2 | Update after revised TR1 RIDs |
| 1.1 | 2021-01-28 | 5, 6.4.2 | Added information about PES connections |
| 1.2 | 2021-10-25 | 1.2, 6.4.2.2.1, 7.10, 9.6, 9.7 | Update after 2 nd revision TR1 RIDs |
| 1.3 | 2022-02-04 | 4.3, 6.4.2.2.1, 8.8, 9.7, 11.3 | Added note on remote desktop restriction of license Mark metrics introduced in [RD16] Finalize temporal SI guidelines Updated PointingSat & references |

Table of Contents

| | | |
|-------|---|----|
| 1 | Applicable and Reference Documents..... | 7 |
| 1.1 | Applicable Documents | 7 |
| 1.2 | Reference Documents | 7 |
| 2 | Terms, Definitions and Abbreviated Terms | 9 |
| 2.1 | Acronyms | 9 |
| 2.2 | Definitions | 10 |
| 2.3 | Keywords | 10 |
| 3 | Introduction..... | 11 |
| 4 | Installation and Start-Up | 12 |
| 4.1 | Prerequisites | 12 |
| 4.2 | Installation | 12 |
| 4.3 | License Management..... | 13 |
| 4.3.1 | Node-Locked License | 13 |
| 4.3.2 | Floating License..... | 14 |
| 4.4 | Running PEET | 16 |
| 5 | Workflow | 18 |
| 6 | Graphical User Interface..... | 20 |
| 6.1 | Input Generation and Input Formats..... | 20 |
| 6.1.1 | Input fields and Table Data..... | 20 |
| 6.1.2 | Import of External Data..... | 20 |
| 6.1.3 | Units..... | 22 |
| 6.2 | Toolbar Icon and Keyboard Shortcut Overview | 24 |
| 6.3 | System Editor Window | 25 |
| 6.3.1 | Editor Panel | 26 |
| 6.3.2 | Execution Log | 28 |
| 6.4 | Menus | 28 |
| 6.4.1 | File Menu | 28 |
| 6.4.2 | Setup Menu | 32 |
| 6.4.3 | Database Menu | 46 |
| 6.4.4 | Analysis Features Menu | 47 |
| 6.4.5 | Windows Menu | 71 |

| | | |
|-------|---|-----|
| 6.4.6 | Info Menu | 71 |
| 6.5 | Database Browser | 71 |
| 6.5.1 | Pointing Error Source Blocks..... | 71 |
| 6.5.2 | Basic Blocks | 73 |
| 6.5.3 | Static System Blocks | 74 |
| 6.5.4 | Dynamic System Blocks | 74 |
| 6.5.5 | Evaluation Blocks | 75 |
| 6.6 | Budget Tree View | 75 |
| 6.7 | Breakdown Tree View | 78 |
| 6.8 | Plot Window | 81 |
| 7 | Script-Based Execution | 83 |
| 7.1 | Common Function Parameters and Input Default Values | 83 |
| 7.2 | Selecting a Scenario | 84 |
| 7.3 | Initialising a Scenario | 84 |
| 7.4 | Inspecting Scenario Settings | 85 |
| 7.5 | Modifying Scenario Settings | 90 |
| 7.6 | Evaluating a Scenario | 93 |
| 7.7 | Evaluating Specific Analysis Features | 94 |
| 7.8 | Analysing and Inspecting Scenario Results..... | 94 |
| 7.8.1 | Functions | 94 |
| 7.8.2 | “Browsing” the Analysis Object..... | 99 |
| 7.9 | Exporting Scenario Results..... | 100 |
| 7.10 | Support Functions..... | 102 |
| 8 | Example Scenarios..... | 105 |
| 8.1 | Example 1: Statistical Budgets | 105 |
| 8.1.1 | Description | 105 |
| 8.1.2 | Setup in PEET | 105 |
| 8.2 | Example 2: Spectral Budgets & Requirement | 110 |
| 8.2.1 | Description | 110 |
| 8.2.2 | Setup in PEET | 110 |
| 8.3 | Example 3: User-defined Domains | 113 |
| 8.3.1 | Description | 113 |
| 8.3.2 | Setup in PEET | 113 |
| 8.4 | Example 4: Correlation | 116 |

| | | |
|-------|---|-----|
| 8.4.1 | Description | 117 |
| 8.4.2 | Setup in PEET | 117 |
| 8.5 | Example 5: Frequency Bandwidth and Refinement | 120 |
| 8.5.1 | Description | 120 |
| 8.5.2 | Setup in PEET | 121 |
| 8.6 | Example 6: Script-Based Execution | 123 |
| 8.6.1 | Description | 124 |
| 8.6.2 | Setup in PEET | 124 |
| 8.7 | Example 7: User-Defined Post-Processing | 127 |
| 8.7.1 | Description | 128 |
| 8.7.2 | Setup in PEET | 128 |
| 8.8 | PointingSat Scenario | 129 |
| 9 | Hints and Guidelines | 131 |
| 9.1 | Software Limitations..... | 131 |
| 9.2 | Software Accuracy | 131 |
| 9.3 | Basic Sensor/Actuator Models..... | 133 |
| 9.4 | Periodic Signals and Evaluation Time Frame..... | 133 |
| 9.5 | Ensemble Domains..... | 134 |
| 9.6 | Worst-Case Values of Distributions and Worst-Case Distributions | 136 |
| 9.7 | Remark on Temporal Statistical Interpretation | 139 |
| 9.8 | Sensitivity Analysis | 141 |
| 10 | Database Blocks..... | 143 |
| 10.1 | Common Block Settings..... | 143 |
| 10.2 | Accelerometer Noise..... | 143 |
| 10.3 | Camera Range Noise | 144 |
| 10.4 | Container..... | 145 |
| 10.5 | Coordinate Transformation | 146 |
| 10.6 | Dynamic System | 147 |
| 10.7 | Feedback System | 148 |
| 10.8 | Flexible Plant | 149 |
| 10.9 | General Periodic Error | 151 |
| 10.10 | GPS..... | 153 |
| 10.11 | Gyro Rate Noise | 154 |
| 10.12 | Gyro-Stellar Estimator..... | 155 |

| | | |
|---------|-------------------------------|-----|
| 10.13 | Input PEC..... | 155 |
| 10.14 | Mapping | 156 |
| 10.15 | PEC..... | 157 |
| 10.16 | PES (Time-Constant)..... | 157 |
| 10.17 | PES (Time-Random)..... | 162 |
| 10.17.1 | Random Variable | 162 |
| 10.17.2 | Random Process | 167 |
| 10.18 | PID Controller | 172 |
| 10.19 | Reaction Wheel..... | 173 |
| 10.19.1 | Reaction Wheel (Force)..... | 173 |
| 10.19.2 | Reaction Wheel (Torque) | 176 |
| 10.20 | Rigid Plant..... | 179 |
| 10.21 | Star Tracker Noise | 180 |
| 10.22 | Static System | 182 |
| 10.23 | Summation | 182 |
| 10.24 | Total Error | 183 |
| 10.25 | Total Error (Position)..... | 184 |
| 10.26 | Block Overview | 185 |
| 11 | Troubleshooting..... | 187 |
| 11.1 | Incompatibilities | 187 |
| 11.2 | License Activation | 187 |
| 11.3 | Starting PEET in MATLAB | 189 |
| 11.4 | Other Support | 191 |

1 Applicable and Reference Documents

1.1 Applicable Documents

- [AD1] ECSS-E-ST-60-10C; Control Performance Standard ECSS-E-ST-60-10C, ESA-ESTEC Requirements & Standards Division, 2008.
- [AD2] ESSB-HB-E-003; ESA Pointing Error Engineering Handbook, Issue 1, 19 July 2011

1.2 Reference Documents

- [RD1] IEEE Standard Specification Format Guide and Test Procedure for Linear Single-Axis, Nongyroscopic Accelerometers; IEEE Std 1253-1998 (R2008); IEEE: New York, NY, USA, 2008
- [RD2] IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros; IEEE Std 952-1997; IEEE: New York, NY, USA, 1998
- [RD3] Control Performance Guidelines, ESSB-E-HB-60-10A; ESA-ESTEC Requirements & Standards Division, 2008
- [RD4] Random Data – Analysis and Measurement Procedures, Bendat J.S., Piersol A.G., 3rd edition, Wiley, 2000
- [RD5] PEET2 - Systems engineering support to the development of the Pointing Error Engineering SW Framework, GNC_F.TCN.779118.ASTR, Iss.1.4, Project Documentation, 2015
- [RD6] Error Budgets for Formation Flying Missions, NPD/5022/TD/TR/001 v1.r1.m0, ESA Documentation, Harwood A., March 2008
- [RD7] PEET Mathematical Algorithm Description, ASTOS-P4COM-TN-003, Project Documentation, 2022 (included in PEET electronic manual)
- [RD8] Reaction Wheel Microvibration Model , PFF-MEMO-MC-001; Memo, Casasco M., April 2013
- [RD9] Development and Validation of Empirical and Analytical Reaction Wheel Disturbance Models , Masterson R.A., Master Thesis, Massachusetts Institute of Technology, June 1999.P3-EST-TN-7001
- [RD10] Development of Empirical and Analytical Reaction Wheel Disturbance Models, Masterson R.A., Miller D.W., Grogan R.L., AIAA 99-1204, AIAA Structural Dynamics and Materials Conference, St. Louis, USA, 1999.
- [RD11] Star-Tracker Noise Model, Project Documentation, Memo, Casasco M., April 2012
- [RD12] PointingSat Case Study, GNC_F.TCN.788536-AIRB, Project Documentation (available with software release), 2022
- [RD13] End-To-End Testing Document, GNC_F.TCN.788544.AIRB, Project Documentation, 2016
- [RD14] “How to use POI in Matlab 2018a+”, The Mathworks Matlab Central, website: <https://www.mathworks.com/matlabcentral/answers/422950-how-to-use-poi-in-matlab-2018a>, last visited Nov 28., 2019
- [RD15] https://www.lisamission.org/ltpda/usermanual/ug/sigproc_tfe.html, last visited Aug 18., 2021

[RD16] ESSB-HB-E-003; ESA Pointing Error Engineering Handbook, Issue 1.1 or 2, to be released (ongoing update)

2 Terms, Definitions and Abbreviated Terms

2.1 Acronyms

The following abbreviations are used throughout this document.

| Acronyms | |
|-------------|---|
| AD | Applicable Document |
| ASM | Advanced Statistical Method |
| AST | Analysis Step (in [AD2]) |
| BLWN | Band-Limited White Noise |
| CDF | Cumulative Distribution Function |
| CESS | Coarse Earth and Sun Sensor |
| CRV | (Time-) Constant Random Variable |
| D | Drift (Signal) |
| ECSS | European Cooperation for Space Standardization |
| ESA | European Space Agency |
| GRN | Gyro Rate Noise |
| GUI | Graphical User Interface |
| IEEE | Institute of Electrical and Electronics Engineers |
| I/O | Input/Output |
| JMI | Java MATLAB Interface |
| LoC | Level of Confidence |
| LTI | Linear Time-Invariant |
| P | Periodic (Signal) |
| PDF | Probability Density Function |
| PEEH | Pointing Error Engineering Handbook |
| PEET | Pointing Error Engineering Tool |
| PEC | Pointing Error Contributor |
| PES | Pointing Error Source |
| PSD | Power Spectral Density |
| RV | (Time-) Random Variable |
| RP | Random Process |
| S.I. | Statistical Interpretation |
| SSM | Simplified Statistical Method |
| WC | Worst-Case |

2.2 Definitions

The following definitions are used throughout this document.

| Definitions | |
|---------------------------------|---|
| Block mask | The input dialog provided by the blocks for parameter input. |
| Domain | The generic term used to assign error sources to a group according to their time- or ensemble-random properties. |
| Error Contribution Block | All evaluation blocks where a signal is evaluated w.r.t. a requirement, i.e. <i>PEC</i> and <i>Total Error</i> blocks; Pointing error contribution ("PEC", without the keyword style) block is used interchangeably throughout the document |
| Signal | The error signal information which is exchanged between adjacent blocks. |

2.3 Keywords

The following keyword styles are used throughout this document to simplify the retrieval of elements in the graphical user interface.

| Equivalence | |
|----------------------|--|
| <i>Keyword Style</i> | Block present in the database |
| <i>Keyword Style</i> | Selectable element in a drop-down list |
| <i>Keyword Style</i> | Label of a panel (bold) or input field |
| <i>Keyword Style</i> | Menu element |
| <i>Keyword Style</i> | GUI window |

3 Introduction

The ECSS Control Performance Standard E-ST-60-10C [AD1], published in November 2008, provides solid and exact mathematical elements to build up a performance error budget. However, an additional document that provides guidelines and summation rules based on the top-level clauses gathered in this ECSS-E-ST-60-10C standard was considered necessary by ESA to provide ESA projects with a clear pointing error engineering methodology. This methodology is the basis for a step-by-step process with guidelines, recommendations and examples consistent with and complementing the ECSS standard. The answer to this necessity is the ESA Pointing Error Engineering (PEE) Handbook [AD2] that was published in 2011 as ESA applicable document with the reference ESSB-HB-E-003.

According to the methodology defined in the Control Performance Standard [AD1] and in the Pointing Error Engineering Handbook [AD2], a software tool called PEET (**P**ointing **E**rror **E**ngineering **T**ool) was developed to support system engineers and control engineers working in CDF and in Phase A studies in performing preliminary pointing error engineering tasks.

PEET was designed as an extension to MATLAB. It uses the computational power provided by MATLAB and provides the user a graphical user interface suitable for building pointing systems and analysing pointing errors. The graphical user interface is implemented in Java. The core of the PEET software, containing the mathematical algorithms for error computation, is implemented using MATLAB classes. PEET runs completely inside the MATLAB environment.

The PEET prototype released in the end of 2012 (with several updates until 2015) was restricted to the simplified statistical method described in [AD1] and [AD2], i.e. results are based on the assumption that the central limit theorem is applicable and the total error follows a (nearly) Gaussian distribution. This assumption may lead to significant deviations from the actual result, in case dominant non-Gaussian error contribution exist.

The key update of the prototype to this release version supersedes the above assumption by taking into account the probability density function information of random variable error contributions rather than only fundamental statistical moments. Furthermore, it extends the options for defining error sources and their evaluation and provides a direct support for requirement breakdown activities in a revised and updated graphical user interface.

This document serves as user manual for the software and guides the user in setting up scenarios (pointing systems & requirements) and evaluating pointing budgets.

Chapter 4 describes the installation of the tool and how to start it from the MATLAB command line. Chapter 5 describes a typical workflow with the tool and links these steps to the analysis steps in [AD2]. Chapter 6 introduces the graphical user interface of PEET with all its windows and menu options. Furthermore, it shows how scenario is defined with all its details.

Chapter 7 describes the (MATLAB) script-based execution of the tool without using the GUI once a system has been set up. Chapter 8 gives some simplified step-by-step examples for defining and analysing pointing systems and chapter 9 adds some general hints and guidelines for that purpose. Finally, chapter 10 gives a detailed description of all the building blocks in PEET with all related settings and information on the background models.

4 Installation and Start-Up

4.1 Prerequisites

The following items are mandatory to install PEET.

- Standard desktop PC or laptop (16GB RAM or more recommended)
- Windows 10 / Linux (Ubuntu 20)
- MATLAB 2011b (MATLAB 2016b or higher recommended), 64 bit only
- MATLAB Control System Toolbox

The following **incompatibilities** exist for PEET (see also chapter 11.1):

- MATLAB Text Analytics Toolbox (since MATLAB 2017b)

4.2 Installation

Note: The installation procedure needs to be performed by a user with administrator rights.

Note: The PEET software has to be installed either on a computer where a node-locked license will be activated or on a client computer when PEET is foreseen to run using a floating license. In the latter case there is no need to install the PEET software on the computer where the *FLEXlm* license server with the PEET license is running.

PEET is installed by executing (e.g. double-clicking) an installation file named `Setup_PEET_[version].exe`, which can be downloaded from the Astos Solutions website after registration.

In case the same software version (first and second level of version numbering are identical) is already installed on the system, the existing version can be uninstalled (recommended) before the installation process continues.

Please follow the instructions of the installation program. After few steps, a serial number is requested which should have been received via email from Astos Solutions GmbH (if not, please contact service@astos.de). Then a destination directory for the PEET installation needs to be specified which is referred to as `#peetroot#` in the following.

Later in the process, you are requested to choose between three different installation types:

- **Typical** – A full installation is performed
- **Minimal** – Only the required parts are installed
- **Custom** – The parts to be installed can be chosen

Once the installation process has finished, any MATLAB installation that fulfils the requirements stated in chapter 4.1 can be used to start PEET.

First, start the desired MATLAB installation and browse to the `#peetroot#` folder. In this folder, execute the script `installPEET.m` and follow the instructions displayed in the MATLAB command window. In case a previous PEET version is detected by the script, it

is automatically uninstalled to avoid issues due to shadowing classes. Dependent on user rights and preferences, both admin and local installations are supported.

At the end of the installation process, a restart of MATLAB is required to make all changes take effect.

Note: If PEET shall be used with multiple MATLAB versions on one machine, the installation step using the `installPEET.m` script needs to be repeated once for each MATLAB version.

After restarting MATLAB, PEET can simply be started by typing “`peet`” in the MATLAB command window.

At the first start-up, an error dialog with message “No feature PEET version 1.X found.” May show up. This message indicates a lack of an active or valid license and can safely be ignored at this point (*Ok* button). Setting and activating a license is finally realized as described in the next chapter.

4.3 License Management

The activation of a node-locked license and of a floating license is described in the following two paragraphs. A node-locked license is a license connected to the hardware of one computer on which it is activated and is only valid for the PEET installation on this computer. Alternatively, several users can share concurrently a certain number of licenses in the same network using a floating license infrastructure.

The status of the license can be monitored in the *Info* → *License Manager...* menu from the *Features* tab.

Note: After a successful license activation, PEET needs to be restarted once to make the activation take effect and to unlock all menu and toolbar actions.

Note: In case of a stand-alone license based on trusted storage (no USB dongle), the PEET software has to be used directly on the computer where it is installed. Using PEET via Microsoft Windows Remote Desktop (RDP) connection is not supported. In this case, an error message "No feature PEET version X.Y found!" is shown.

4.3.1 Node-Locked License

In the main window of the software, use the menu item *Info* → *License Manager...* to access the *License Manager* dialog. This dialog contains two tabs: The first tab *Features* provides an overview of the available features. In order to activate a license, the tab *License Management* needs to be selected. It provides an overview of all locally activated licenses. To activate a new license, please click on the *Activate* button to open the *License Activation* wizard which will guide you through the activation process.

First, the connection type needs to be selected. Generally, the license shall be activated using *Internet connection*. Only in cases where an Internet connection is not available on the computer where the PEET software is installed, an *Offline* activation has to be performed. Please select the preferred activation method and click the *Next* button:

License activation using an Internet connection

The following input is required:

- **Entitlement ID:** has been received in an email containing the license data or can be retrieved from the web portal (licenses.astos.de). Note, the entitlement ID is not the user ID.
- **Access code:** has been provided in an email together with the user ID. It is also used to access the web portal.

A click on the **Next** button starts the activation process and shows the result of the license activation. In case the activation fails, the **Back** button can be used to check the provided data for the activation.

Offline license activation

For an offline activation please follow these steps:

1) Entitlement ID and Access code

The following input is required:

- **Entitlement ID:** has been received in an email containing the license data or can be retrieved from the web portal (licenses.astos.de). Note, the entitlement ID is not the user ID.
- **Access code:** has been provided in an email together with the user ID. It is also used to access the web portal.

Click on the **Next** button to continue.

2) Activation Request

The content of this dialog page needs to be copied to a text file (request file). This file is necessary in order to generate a response file on the web portal.

3) Computer with internet connection

The generated request file needs to be copied to a computer with an Internet connection. On this computer please log in to the web portal (licenses.astos.de). Click on the **Manage** button to open the next page and then click on the **Activate license** button. Please follow the instructions and upload the request file and download the response file. Copy the response file to the computer with no internet connection on which the PEET license activation is in progress in order to continue the process.

4) Activation Response

Click on the **Next** button on the Activation Request dialog page and a dialog is shown where the generated response file has to be provided in order to proceed with the license activation.

5) Activation Process

A click on the **Next** button starts the activation process and shows the result of the license activation. In case the activation fails, the **Back** button can be used to check the provided data.

4.3.2 Floating License

Note: This procedure needs to be performed by a user with administrator rights.

In case of a floating license infrastructure (also called concurrent license infrastructure) a license server has to be installed on one computer (server). All computers (clients) on which PEET shall be used have to have access to the server computer via a network. PEET must be installed on all client computers (see chapter 4.2) and does not need to be installed on the server computer. It is not required to activate any licenses on the client computers. Instead a license server package (FLEXlm) — downloadable from www.astos.de — has to be installed on the server and the floating PEET license has to be activated only there. This package contains detailed information about installation and administration of the license server. Please follow this reduced step-by-step guide to install and activate a server license:

- 1) Go to <http://www.astos.de/downloads> and log in with your user name and password.
- 2) Download `FLEXlmXXX_Windows.zip` (XXX identifies the program version).
- 3) Extract all the files from `FLEXlmXXX_Windows.zip` in a directory of your choice (e.g. `C:\Program Files (x86)\Astos Solutions\flexlm`). This folder will be referred to as `#flexlm_folder#` throughout this guide.
- 4) Under Windows, on the administrator *command prompt*, navigate to the `#flexlm_folder#\Anchor_Service_Installer` folder and run `installanchorservice.exe astoslic PEET`.

Under Linux, on the administrator *command prompt*, navigate to the `#flexlm_folder#\Anchor_Service_Installer` folder and run the `install_fnp.sh` script without parameters.

- 5) On the *command prompt* navigate to the `#flexlm_folder#\svractutil` folder and launch `svractutil.exe -a` to start the activation process. The process of activating a license on the server is identical to the activation process of a node-locked license described in the previous section (4.3.1). The only difference is that textual input on the command line is used instead of a wizard.

- 6) Navigate to the `#flexlm_folder#\ladmin` folder and run `ladmin.exe -allowRemoteStopServer yes`. This will launch the license server. The default port of the web front-end is 8090. A detailed explanation of the command line usage of *ladmin* is provided in the `#flexlm_folder#\fnp_LicAdmin.pdf` file.

Under Linux it should be taken care that *ladmin* is started as a service in order that it is started automatically after a restart of the operating system.

The use of *lmgrd* is not officially supported by PEET but it should also work if set up properly as described in the *FLEXlm* documentation.

- 7) Insert `http://#server_address#:8090` (e.g. `http://192.168.5.25:8090`) in the address field of the web browser and press **Enter** to open the **FlexNet Publisher** web front end. The web front end is used to configure the *FLEXlm* license server.

- 8) On the displayed web page click on the **Administration** link in the top right corner. It opens the administration web page.

- 9) Enter the user name and password and click on the **Submit** button to log in. Both the default user name and password are `admin`. When logging in for the first time the user is asked to change the password, which is advisable.

- 10) Click on the link **Vendor Daemon Configuration** in the lower left corner of the **Administration** page. The **Vendor Daemons** page is displayed.

11) If `astoslic` is available in the **Vendor Daemons** list and its status is **Up**, the license server is running and ready to be used. No further actions are required. If `astoslic` is not available in the list, click on the [Import License](#) button. Select the `astoslic.lic` license file, which is located in the `#flexlm_folder#\lmadmin\astoslic` folder. Also select the check box in order to override the license file on the license server.

Now `astoslic` should be available in the **Vendor Daemons** list. If the status is not **Up**, select `astoslic` from the list. On the new page, click on the [Start](#) button. After a few seconds the status of `astoslic` should change to **Up** and the license server is ready for use.

Now the license server is installed and the service is started. In case of problems please refer to the *FLEXlm* manual `fnp_LicAdmin.pdf` or contact service@astos.de.

Finally it is necessary to indicate the position of the server to the PEET installation on the client. In order to use licenses from a license server, the `license.dat` file located in the `.\license` folder of the PEET installation on each client needs to be edited. Replace the content of this file with:

```
SERVER #server_address# FLEXID=9-a6300155 #optional_server_port#  
USE_SERVER
```

and substitute `#server_address#` and `#optional_server_port#` with the real server data. Here is an example:

```
SERVER 192.168.5.25 FLEXID=9-a6300155  
USE_SERVER
```

The optional port is needed in case the license server is not using the default ports. The floating license installation and activation is now accomplished.

4.4 Running PEET

After a successful installation, the PEET GUI can be started from the MATLAB command window by entering:

- `"peet"`
Starts the PEET GUI with an empty scenario.
- `"peet([scenarioPath])"`
Starts the PEET GUI and loads a scenario by providing the respective absolute path stored in a string `[scenarioPath]`.
- `"peet last"`
Start the PEET GUI and loads the last scenario used in a previous PEET session.

Further, PEET can also be used for script-based operation in MATLAB without using the GUI. All possible commands are explained in section 7.

In case of any issue with the startup, please refer to the troubleshooting section (chapter 11).

5 Workflow

This chapter describes a typical step-by-step workflow for setting up and evaluating pointing scenarios with PEET.

Step 1: Build up the pointing system

The first step is to use the building blocks from the [Database Browser](#) to set up the pointing scenario in the [System Editor](#) from pointing error sources (via optional system transfers) to the total error. Optionally, *PEC* blocks can be used to indicate subsystem budget levels or to indicate where lower level requirements can be defined later on.

Step 2: Definition of ensemble domains (optional)

If desired, it is possible to define dedicated domains for the ensemble properties of error sources (see chapter 6.4.2.1) in the [Setup → Ensemble Domains](#) menu.

By default, one single “global” domain is implicitly used for all ensemble properties and no further user interaction is required.

Step 3: Specification of scenario evaluation settings and requirements

This step covers the specification of the required parameters for an unambiguous definition of pointing requirements (see [AD2],chapter 6):

- A) Selection of the pointing error index and the requirement type (spectral or statistical) using the [General](#) tab in the [Setup → Scenario Definition](#) menu.

Tip: Multiple requirements sets can be defined in one scenario.

- B) Optional: Specification of requirement values and IDs associated to all *PEC* and *Total Error* blocks using the [Requirement Specification](#) tab in the [Setup → Scenario Definition](#) menu. This allows comparison with budget values and tracking of violations after the computation
- C) Definition of the default frequency evaluation settings (only necessary if random process type error sources are used in the scenario), specification of the budget evaluation period and selection of the line-of-sight axis in the [Setup → Evaluation Settings](#) menu.

Step 4: Define error source dependencies (optional)

The dependencies between all (connected) pointing error sources and their axes can further be configured, in particular:

- A) Correlation (temporal and ensemble) in the [Setup → Set Correlations](#) menu. By default, all quantities are considered as uncorrelated.
- B) Coherence between random process error sources in the [Setup → Set Coherence](#) menu. By default, all random process sources have no coherence.
- C) Phase differences between periodic signals in the [Setup → Set Phase Relations](#) menu. By default, all periodic signals have zero phase difference.

Step 5: Refine frequency grid for random process evaluation (optional)

The default frequency evaluation grid and the evaluation bandwidth can be checked and adapted to the user needs in the [Setup → Check Frequency Bandwidth](#) and [Setup →](#)



Refine Frequency Grid menus. This is only necessary in case random process error sources are defined in the scenario.

Step 6: Define 'post-processing' features (optional)

If desired, select and/or define auxiliary evaluation algorithms using the **Analysis Features** menu (see 6.4.4).

This comprises specific routines to be applied to the "nominal" x/y/z/LoS data available at **PEC** and **Total Error** blocks (e.g. conversion to azimuth/elevation errors) as well as the definition of user-defined algorithms. In addition, the analysis of the contribution (in [%]) of each PES and/or PEC alone to any other **PEC/Total Error** block can be enabled/disabled in the evaluation of a scenario.

Step 7: Evaluate the scenario

Evaluation for of the pointing budget either for all requirement sets defined in the **Setup → Scenario Definition** menu) by pressing the -button in the toolbar. Alternatively, only a single set can be evaluation by pressing the -button after the desired set has been selected in the adjacent drop-down list.

After the evaluation is completed, the results can be inspected in the **Budget Tree Window** and/or compared to the requirements in the **Breakdown Tree View**.

Step 8: Export the results (optional)

The results of an evaluation can be exported to an MS Excel format. This report can be configured and generated using the **File → Create Report...** menu (see 6.4.1.1).

Note: Manually creating a report is only possible for the last evaluated requirement set of a scenario.

6 Graphical User Interface

This chapter provides an introduction to the graphical user interface of PEET, in particular the various window types and menus which are used to configure a pointing scenario and to analyse pointing budgets.

6.1 Input Generation and Input Formats

6.1.1 Input fields and Table Data

Any kind of input in menus or block dialogs needs to be provided either in input fields or tables.

The size of the input fields depends on the dimension of the quantity to be specified (see Figure 6-1). By default, only numerical inputs are supported (see chapter 6.1.2.2 on how to use MATLAB notation) and invalid entries are immediately highlighted while editing with a yellow coloured background of the input cell.

Tip: All input fields accept scientific notation, e.g. $1.53e-4$ for $1.53 \cdot 10^{-4}$.

A left-click on an input field enables the editing mode. In case of inputs for multiple dimensions, also the Tab key be used to continue to the next cell.

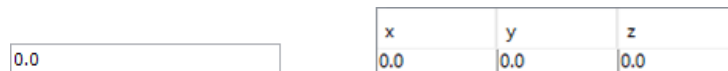
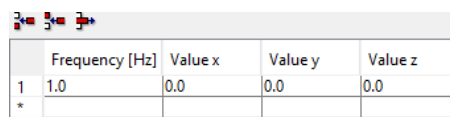


Figure 6-1: 1D and 3D input fields




In many cases, a single value (for each dimension) is not sufficient and a larger array of data is required (e.g. numerical values of a spectrum over frequency, amplitudes of periodic signals at multiple frequencies). In these cases, data tables serve as input interface (see Figure 6-2).



| | Frequency [Hz] | Value x | Value y | Value z |
|---|----------------|---------|---------|---------|
| 1 | 1.0 | 0.0 | 0.0 | 0.0 |
| * | | | | |

Figure 6-2: Data table (exemplary)


Editing a table works similarly as for the input fields. An input to the table row labelled with an asterisk (*) adds a new row to the table. In addition, three icons are available above the table to modify the table layout:

-  Add a row above the currently selected row.
-  Add a row below the currently selected row.
-  Delete the currently selected row.

6.1.2 Import of External Data

Apart from the manual input of numerical values, it is also possible to import data from Excel spreadsheets and MATLAB workspace variables.

6.1.2.1 Linking to Excel data

A link of values to Excel data is initiated with a right-click on the input field and selecting **Import from Excel**. This opens a new dialog as shown in Figure 6-3 where first the (relative) path of the *Excel file* (.xlsx format) needs to be specified. It is also possible to browse to the file location using the  -button.

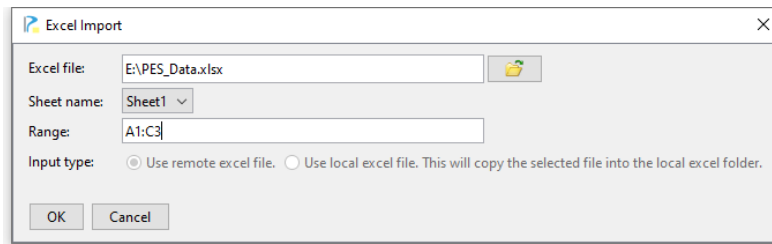


Figure 6-3: The Excel import dialog

The next step is to select the *Sheet name* within the file from the dropdown list. Finally, the data *Range* must be defined. In case of vector data, the data range can either be defined as a row or column vector dependent on the format of the input field to be linked. The link is then confirmed with the **OK** button and an automatic consistency check of cell data and dimension is applied.

Scalar values

Scalar values can be defined as follows: The column is defined by upper case letters, the row is defined by an index. The row index of the first row is 1.

Example: Setting the *Range* to A1 links the first cell in the first row of an Excel sheet.


Matrix and table data

Matrix and table data values can be defined similar to scalar values. In this case, the upper left and the lower right cell of the data range must be defined.

The upper left and the lower right cell are defined using upper case letters for the columns and indices for the rows. The row index of the first row is 1. The cells are separated by ':'

Example: Setting the *Range* to A1 : C3 links a 3x3 matrix starting at the first cell of the first row of an Excel sheet.

6.1.2.2 Using MATLAB workspace variables

This method is available for most menu or block panels. It is initiated by enabling the checkbox next to the  -icon.

When enabled, the affected input fields or tables reduce to a “scalar” input field no matter what the original dimension was. These input fields are then no longer restricted to numerical values, but accept any kind of MATLAB-compatible notation of a quantity with a matching size. This can be used to:

- Express vectors and matrices, e.g.
 - [1 4.5 -2.6] for a 1x3 vector
 - [1 4.5 -2.6]’ for a 3x1 vector

- `[1 2; 1 3; 1 4.3]` for a 3x2 matrix
- Use “functions” and operations on values, e.g.
 - `5/3, 3*pi` for operations on a value
 - `cos(1.2), log10(3)` to apply a function to a value
- Use workspace variables, e.g.
 - `myVector` to link a complete variable `myVector`
 - `myMatrix(2, :)` to link the second row of matrix `myMatrix`
 - `myStruct.vec1` to link the subset `vec1` of a structure `myStruct`
- Combinations of above

Tip: When sharing scenarios with other users, make sure not to use any toolbox functions in the input fields that might not be available to all users.

Using workspace variables is especially important for a script-based operation (see chapter 7) in a batch-mode sense, i.e. where values of a certain parameter or settings is modified repeatedly between evaluations without the need of using the GUI.

6.1.3 Units

PEET supports the association of a unit to a given (set of) input(s) for blocks and menus. There are two different purposes for using units which are described below.

Units for block parameters and scenario settings

Adjacent to most input fields and tables in block dialogs and menus, a drop-down list for the unit of the input is available to allow a more flexible definition of input parameters.

The numerical values of the input fields are not updated in the GUI when another unit is specified, i.e. the software only performs an internal conversion of the given values to the unit required by the various models.

Units of error signals and consistency checks

Apart from the unit of certain parameters, a unit can also be assigned to the entire output signal of a block. Different to the unit specifications described above, this assignment becomes visible to the user as it defines the unit of the displayed results in the [Budget Tree View](#) and [Breakdown Tree View](#) and in a report. Dependent on the block type (see chapter 10), sometimes it is also possible or necessary to specify a unit for the input signal.

In any case, the following consistency checks are automatically applied to the signals:

- When two or more signals are summed, their units must belong the same unit group (e.g. [Time]). They can however differ in the unit itself (e.g. [h] and [s]) and the software accounts for the proper conversion for the summation.
- The selected input unit of a block has to belong to the same unit group as the unit of the input signal.
- Some blocks are restricted to inputs/outputs of a certain unit group. In this case, only units from the supported unit group can be selected.

Any inconsistency detected when connecting blocks in the scenario leads to the pop-up of a dialog informing about the inconsistency.

Tip: It is also possible to disable these *Checks and conversions* for the signal units by settings the respective toggle-button to **disabled** in the **Setup → Evaluation Settings** menu. This might be useful to avoid numerous unit definitions for a quick-assessment with a data set where the unit consistency is already confirmed or known. "Expected" default I/O units of dedicated blocks are provided in chapter 10.26.

Definition of custom units

If the desired unit is not present in a unit selection drop-down list of an input field, it can be defined or composed manually by choosing the **Custom...** option. This opens a dialog window which allows building up a **User-defined unit** from scratch using the following setup options:

- **Unit group** specifies the category of a unit element, e.g. **Length**.
- **Prefix & unit** are drop-down lists to specify an available unit of the selected **Unit group** (e.g. **Meter**) which can optionally be preceded by a metric system prefix (e.g. **Centi-**).
- A panel with buttons to add or remove elements of a composed unit. This functionality is described below.

To **add an element** to a composed unit at the desired location (numerator or denominator), use the respective **Add** button above or below the fraction bar. Added elements then appear in the **Short unit string preview** panel.

To **remove an element**, select it with a left-click in the **Short unit string preview** panel and use the respective **Remove** button above or below the fraction bar.

To **increase/decrease an exponent of an existing element**, select it with a left-click and use the respective **Add/Remove** button again. The step size of the exponent is 0.5 to cope also with special units required for the definition of certain spectral models.

Tip: Selected elements of a composed unit are indicated with a red frame in **Short unit string preview** panel.

Note: A user-defined unit can be composed of elements from different unit categories (i.e. length, mass, etc.), but the result has to be compliant with the quantity to be described.

The **SI view** panel displays the conversion factor of the user-defined unit with respect to the related SI unit of the quantity. In case the resulting unit does not match the required unit for an output signal or block parameter, `[incompatible unit]` is displayed.

Example: Recreating [Newton] from fundamental SI units

Although **Newton** is already defined in the **Unit group Force**, it could also be "rebuilt" from its SI components (kgm/s²) as follows:

- Choose **Mass** from the **Unit Group** list, then **Kilogram** from the second **Prefix & unit** list. No prefix is required from the first list.
- Click the **Add** button above the fraction bar **twice** to add the element to the numerator with proper exponent.
- Choose **Length** from the **Unit Group** list, then **Meter** from the second **Prefix & unit** list. No prefix is required.

- Click the **Add** button above the fraction bar **twice** to add the element as factor to the numerator with proper exponent.
- Choose **Time and Frequency** from the **Unit Group** list, then **Second** from the second **Prefix & unit** list. Again, no prefix is required.
- Click the **Add** button below the fraction bar **four times** to add the squared element to the denominator.

Recently used units

Units created via the **Custom...** option are automatically made available in all unit drop-down lists for quick selection – in case they are compatible with the permitted unit of a block or parameter.











The maximum number and entries in this list can be reset/configured in the **Recent units list** panel of the **Preferences** menu (see 6.4.2.9).




















6.2 Toolbar Icon and Keyboard Shortcut Overview

The following table gives an overview of icons used in the toolbars of the different windows and (keyboard) shortcuts in menus and windows.

Tip: Mouse-over tooltips are also available for the toolbar icons.

**Table 1: Toolbar icons and keyboard shortcuts; ID in center column:
E – System editor, B – Budget Tree View, R- Breakdown Tree View, P – Plot window**

| Icon | Action | Available in | Shortcuts |
|---|-----------------------------------|--------------|--|
| Scenario | | | |
|  | Open a new scenario | E | Ctrl-N |
|  | Open an existing scenario | E | Ctrl-O |
|  | Save the current scenario | E | Ctrl-S |
| Editor/Menu | | | |
|  | Copy block/selection to clipboard | E | Ctrl-C, Right-click + Copy (blocks) |
|  | Cut block/selection | E | Ctrl -X, Right-click + Cut (blocks) |
|  | Delete block/selection | E | DEL Right-click + Delete (blocks) |
|  | Paste block/selection | E | Ctrl -V, Right-click + Paste (blocks) |
|  | Undo modification in editor | E | Ctrl -Z |
|  | Redo modification in editor | E | Ctrl -Y |
|  | Open online-help | E,B,R,P | F1, also mouse-over help in block dialogs and menus |

| Window | | | |
|---|--|---------|--|
|  | Open the Block Database | E | D |
|  | Open the Budget Tree View | E,R | B |
|  | Open the Breakdown Tree View | E,B | R |
|  | Open the System Editor | B,R | E |
| Tree View and Plots | | | |
|  | Reset zoom to default | E,B,R,P | Numpad 0 |
|  | Zoom in | E,B,R,P | Numpad +, Hold right-mouse button + move mouse right |
|  | Zoom out | E,B,R,P | Numpad - Hold right-mouse button + move mouse left, |
|  | Fit to view | E,B,R | Spacebar |
|  | Pick data point | P | |
|  | Save GUI plot | P | |
|  | Save plot as MATLAB figure | P | Only for last computed requirement set |
|  | Refresh plot | P | Only for last computed requirement set |
|  | Show all tables in panel | B,R | |
|  | Hide all tables in panel | B,R | |
|  | Show all plots in panel | B,R | |
|  | Hide all plots panel | B,R | |
| Initialization and Evaluation | | | |
|  | Evaluate currently selected requirement set in scenario | E | |
|  | Evaluate all requirement sets in the scenario | E | |
| Report | | | |
|  | Create report | E | Only for last computed requirement set |

6.3 System Editor Window

The system editor is the main window of PEET. It consists of a menu bar, a toolbar, an execution log and the editor panel.

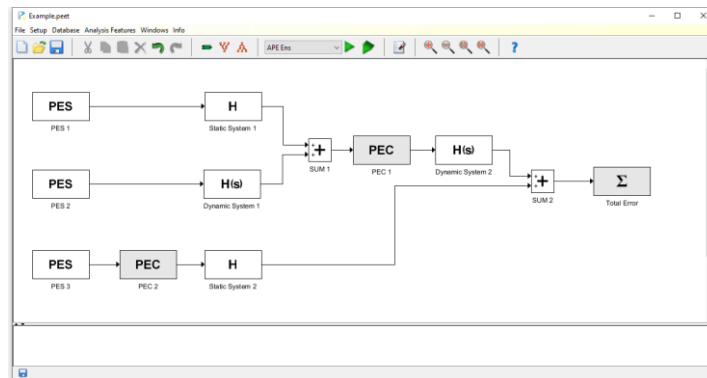


Figure 6-4: The System Editor window

6.3.1 Editor Panel

The editor panel is the main tool to design the architecture of a pointing system. It can be populated with a selection of model blocks from the [Database Browser](#) (chapter 6.5) which then need to be connected to represent the error signal flow. As a minimum, one pointing error source needs to be connected to a *Total Error* block to represent an evaluable scenario.

Note: Each scenario requires at least one error source block and accepts only one *Total Error* (or *Total Error (Position)*) block.

Tip: Populating the editor panel with blocks and connections should be the first step when setting up a scenario (i.e. before further defining the requirement sets) for a straightforward workflow.

Adding and moving blocks and groups

New blocks are added to a scenario by dragging (i.e. left-click and hold) the desired block from the [Database Browser](#) to the editor panel at the desired location. An invisible grid in the editor simplifies the horizontal and vertical alignment of blocks. Multiple instances of the same block can either be realized by repeated dragging or by copy & paste operations (using keyboard shortcuts, right-click menus or the toolbar, see overview in chapter 6.2). In the latter case, all block settings and parameters are copied as well.

Blocks can be moved in a similar way by dragging them to the desired location in the system editor. Groups of blocks (and connections) can be selected starting with a left-click on an empty part of the editor and drawing a frame around the desired group of elements while keeping the mouse button pressed. The selection is completed by releasing the mouse button. To move the entire group, drag any element within the selection to the target location.

Tip: Selected elements in the editor panel are highlighted in red.

It is also possible to copy blocks from one scenario to another using copy & paste operations (dragging is not possible in this case).

Setting block parameters

A double-click with the left-mouse button on a block opens a dialog where all parameters and settings of a block can be specified. An introduction to the different block types is provided in chapter 6.5, the settings of each individual block are detailed in chapter 10.

Enlarging blocks

Most blocks can also be enlarged by dragging the small circle (○) in the bottom-right corner of block. This only has “cosmetic” purpose without any other effect.

Connecting blocks

To connect two blocks, left-click on the input or output connector (>) of the first block. The connector gets highlighted in red. Then left click on the output or input port of the block to be connected.

Note: Loops are only allowed in *Feedback System* blocks. Otherwise an error is thrown.

Tip: Any selected “source” block can also be connected with a left-click on the “target” block while the Ctrl key is pressed. This connects unassigned output port(s) of the source block to unassigned inputs port(s) of the target block.

Tip: A single block output port can also be connected to the input port(s) of multiple blocks (with “Ctrl-Click”).

Editing connections

To edit the “route” (location) of a block connection, it needs to be selected with a left-click. This highlights the connection in red and also displays some “grabbers” (red dots in Figure 6-5). Any of these grabbers can be dragged to a desired location to modify the route of the connection.

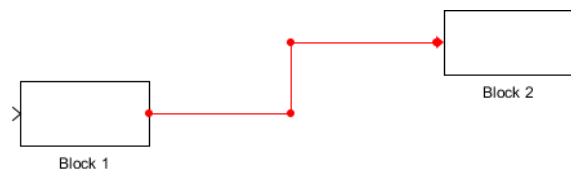



Figure 6-5: Editing block connections

Renaming blocks




Blocks can be renamed with a double-click on the block name. This opens a dialog where the new name can be defined. Block names need to be unique on a system level, they can however be identical on different levels (e.g. inside and outside a *Container* block).


Tip: The block names should be understood as unique, brief identifiers. For detailed user-defined descriptions, the -icon available in each block dialog should be used.

Flipping blocks

By default, a block is oriented such that its input ports are on the left, and its output ports are on the right. The orientation of input/output ports of selected blocks can be flipped using the right-click menu and choosing the option *Flip block(s)*.

Zooming

Zooming the view in the editor panel is possible by pressing and holding the right mouse button and moving the mouse to the left (zoom in) or to the right (zoom out). Alternatively, the respective toolbar buttons can be used (, ) for a stepwise zooming. The  -

button reset the zoom to the default state while the -button sets the zoom such that the entire editor content fits in the current window.

6.3.2 Execution Log

The **Execution Log** serves as scope to track the progress of the evaluation and issues occurring meanwhile. All messages are displayed in order of their appearance during the evaluation. A lateral scroll bar allows to display older messages. The following types of messages types are available (with colour-coding):

- **Evaluation status:** standard messages indicating the current state of the evaluation
- **Warnings:** issues that have been detected but do not prevent an evaluation
- **Errors:** severe errors in the setup or modelling that prevent an evaluation.

Descriptions of warnings and errors are usually start with *[scenarioName.peet -> Blockname]* which allows identification of setup mismatches in a certain block (if applicable) followed by a descriptive message (e.g. *Sigma must be non-negative numeric vector and with at least one non-zero element* in case a Gaussian with zero standard deviation on all axes has been defined). All further stack and ID information is usually not important for a user.

The **Execution Log** also features a right-click menu with the following items:

- **Clear log:** clears all content from the current log
- **Save log:** saves the content of the current log to a .txt file. A dialog opens which allows browsing to the desired location
- **Goto top / Goto end:** quickly navigate to the first/last line of the current log


6.4 Menus

The menu bar in the **System Editor** window gives access to all menus in PEET. The purpose of all menu and submenu items and their configuration options are explained in the following subsections.


6.4.1 File Menu

The items of the **File** menu are used to manage and handle scenarios. Furthermore, also **recently loaded scenarios** from previous sessions are listed in the **File** menu. Selecting such an item serves as shortcut to load one of the related scenarios.

New Scenario

This item opens a new empty scenario. It is the menu-equivalent for the -button in the toolbar.

Open...


This item is used to open an existing scenario. It is the menu-equivalent for the -button in the toolbar. A dialog opens to browse to the location of the desired .peet folder.

Alternatively, the path to the `.peet` file can also be specified directly, e.g. `C:\Budgets\MyBudget.peet`.

Close

This item closes the current scenario. If any modified unsaved data exists, a dialog shows up and allows saving the scenario first (**Yes**). **No** closes the scenario without modifications and **Cancel** leaves the scenario open.

Save

This item is used to save the current scenario, i.e. by overwriting the previously saved version. It is the menu-equivalent for the -button in the toolbar and is not available for a new, unsaved scenario.

Save As...

This item is used to save the current scenario as a new scenario based on the current content (with a different name and/or at a different location). The “source” scenario remains unmodified.

Configure Report...

This item is used to configure the content of the spreadsheet reports which can be (manually or automatically) created to export the results of a scenario evaluation. A detailed description of the options is provided in section 6.4.1.1.Exit

This item is used to exit the PEET application. If any modified unsaved data exists in any open scenario, a dialog shows up and allows saving the scenario(s) first (**Yes**). **No** closes all scenarios without modifications and **Cancel** aborts the exit operation.

6.4.1.1 Report Configuration

By default, the settings in the **Setup** → **Configure Report...** menu include all nominal information of a budget evaluation in one specific Excel report file for each requirement set. The content of a report can however be further configured to include additional auxiliary data or to neglect specific information which is not required.

Tip: Generated reports are always located in the `output\report` subfolder of the current scenario and named according to the requirement set for which the report is generated.

Tip: All reports generated with PEET include unique cell references for all data fields to simplify linking results to other sheets or documents.

Specification parameters

The **Specification parameters** panel provides the following options which can be enabled/disabled using the respective checkboxes:

- **Requirement definition:** Enabling this option creates a sheet “Requirement_Definitions” which contains information about the current requirement set (e.g. error index, window/stability time) and the set of requirement values (functions for spectral requirements) with corresponding IDs.
- **PES properties:** Enabling this option creates a sheet “PES_Properties” which contains a list of all PES defined in the scenario with related such information such the PES dimension, signal class and representation, user-defined description of the sources or as (if applicable) the ensemble domain linked to the sources.
- **PES dependency:** Enabling this option creates a sheet “PES_Dependency” which contains all phase relations, coherence factors and (temporal and ensemble) correlation matrices defined to describe the dependency between error sources.

Results

The **Results** panel provides the following options:

- **PES outputs:** Enabling this option creates a sheet “PES_Outputs” which contains all the mean values and standard deviation of the different error signal components (provided in the selected output unit).
- **System outputs:** This option creates a sheet “System_Outputs” with equivalent information as for the **PES outputs** for the output signals of all transfer and summation blocks in the pointing system.
- **PEC outputs:** This checkbox determines the presence of a sheet “PEC_Outputs” for the evaluated results at all **PEC** and **Total Error** blocks. There are two further refinements possible
 - **Total contribution only:** Provides only the total error contributions at the evaluation blocks in comparison to the specified requirements
 - **Total, Time-Constant & Time-Random contributions:** In addition to the above option, also the separation into the time-random and time-constant contribution that form the total error are included in the sheet.
- **Post-processing:** This checkbox determines if the results (plot and/or tabular data) of any auxiliary analysis defined in the **Analysis Features → Post-Processing** menu (see section 6.4.4.1) shall be included in the report. One additional sheet is created for each defined element with a name corresponding to the analysis type (or user-defined name).
- **PEC Percentages:** This checkbox determines if a tabular overview of the percentage contribution analyses (if enabled in the **Analysis Features** menu, see section 6.4.4.3) shall be included in the report). For each enabled analysis, one specific additional sheet is generated, i.e. “PEC_to_PEC_Percentages” and /or “PEC_to_PEC_Percentages”.

Independent of the selection above, a “General_Information” sheet is always available in each report. This sheet contains information about the working environment (MATLAB version, PEET version, scenario name, creation date) and the pointing system (number of sources, PECs and ensemble domain). Furthermore, it contains the activation list for the error sources as specified in the **Setup → Scenario Definition** menu.

Axis selection

The **Axis selection** panel allows ignoring the results any nominal axis (**x,y,z** and **LoS**) in the report (this has no effect on the **Post-processing** results in the report which can have arbitrary ‘axis’ as output).

Plot generation

The checkboxes in the **Plot generation** panel can be used to automatically generate and save plots when generating a report. This also includes a link to the selected plots directly in the spreadsheet. The file formats in which the plots are saved correspond to the ones specified in the **Setup → Preferences** menu (section 6.4.2.9).

The following plot types can be selected (for more information on the plot types, please refer to section 6.8):


- *PES source info plots* as displayed in the **Source Info** tab for periodic and random process PES in the [Budget Tree View](#)
- *PES/System output PDF plots* individually for each signal component as displayed in the [Budget Tree View](#)
- *Cumulated variance plots* for random process contributions as displayed for PEC blocks in the [Breakdown Tree View](#)
- *PEC output CDF plots* individually for *Time-Constant/Time-Random* and *Total* error contributions as displayed for PEC blocks in the [Breakdown Tree View](#)
- *PEC output PDF plots* individually for *Time-Constant/Time-Random* and *Total* error contributions as displayed for PEC blocks in the [Budget Tree View](#)

Report naming

By default, the report spreadsheets follow a fixed naming scheme combining the scenario name and the requirement set for which the report is generated. In case any user-defined *Prefix* or *Suffix* can be defined in the respective input field of the [Report naming](#) panel. Both inputs can either be directly specified as a string in single quotes (e.g. 'myString') or as a MATLAB variable that represents a string (without quotes).

Report generation

The options in the [Report generation](#) panel define some general aspects for the reporting:

- *Always create report after evaluation* automatically triggers the report generation when the respective checkbox is enabled. The current settings are then used to create reports automatically for all subsequent evaluations of the scenario).
- Nominally, a separate spreadsheet is generated for each requirement set containing all sheets selected in the [Specification parameters](#) and [Results](#) panels. The *Generate PEC overview (with 'compute all' only)* flag can be used to additionally create a spreadsheet named [scenarioName]_PEC_Overview which contains a summary of the PEC output results only, but for all defined requirement sets of a scenario in a single sheet. If enabled, this overview is then created when all requirement sets are subsequently evaluated (-button).
- The *Overwrite existing reports* checkbox determines the behaviour in the case a report file with the targeted name already exists in the scenario \output\report folder when generating a report. If checked, the existing file is overwritten; if unchecked, the existing report file is preserved and a time stamp of the form _yyyymmddTHHMMSS is appended to the file name.
- *File format* allows selecting the type of the spreadsheet file, either *.xls* or *.xlsx*.
- In case one or more requirement sets have already been computed in the current session for a scenario, the report for the last computed requirement set can also be immediately created using the *Create* button.

Note: Using the *Create* button to immediately generate a report is only possible for the last evaluated requirement set (as only the data for one requirement set is stored in the MATLAB workspace).

6.4.2 Setup Menu

The items of the **Setup** menu are used to configure a variety of global settings concerning the specification and evaluation of requirements and to define dependencies between different error sources and their axes.

Basically, none of these settings needs to be specified before all building blocks for the pointing system have been configured and connected in the editor panel of the **System Editor**.

The individual items of the **Setup** menu are arranged according the user workflow, i.e. it is convenient to specify settings from top to bottom.

Tip: Menu items followed by an asterisk (*) provide some advanced or special functionality. These settings are not mandatory and might not be required for every scenario.

6.4.2.1 Ensemble Domains...*

The **Ensemble Domains...*** menu can be used to categorize ensemble-random parameters of error sources into groups where each group is represented by a so-called domain.

Such domains could be e.g. "Manufacturing" (error contributions related to misalignments, displacements, multiple satellites, etc.) or "Observations" (error contributions that do not vary in time over a single observation, but due to varying conditions between different observations).

This domain concept has the main advantage, that the definition of a pointing requirement becomes more flexible, as a level of confidence or the statistical treatment of errors can be individually specified for the contributions of each domain, e.g. "The pointing error shall be smaller than X for the worst-case satellite (→ domain "Manufacturing") and for 99.7% of all observations (→ domain "Observations").

Furthermore, it reduces the size of the correlation matrix to be specified, as contributions from different domains are uncorrelated by definition.

Menu settings

A new domain can be introduced with a left-click on **Add...** in the **Domain definition** panel. This opens a dialog where an arbitrary name can be defined to identify the domain. This name then appears in the **Domain Name** table. To **Rename...** or **Delete** an existing domain, the respective buttons can be used after it was selected in the table.

When all desired domains are created, it is mandatory to map each PES to a domain in the **Error source assignment** panel. By default, all PES are assigned to the first domain.



Note: This also applies to PES which have no ensemble-random parameter as they (artificially) have to be assigned to one of the domains for the budget evaluation.

6.4.2.2 Scenario Definition


The **Scenario Definition** is the core menu for evaluation settings and requirements. Here mostly all requirement specification parameters according to [AD2] are defined.

The menu dialog consists of three main parts. A toolbar on top, a panel on the left which lists all defined "requirement sets" and a panel with two tabs on the right which serve for the configuration of each set.

Adding/Removing requirement sets


A new set of requirements can be introduced with a left-click on the  - icon (Similarly, a selected requirement set can be deleted using the  - icon). The new requirement set appears then appears in a list on the left side of the dialog. When selected in the list, it is possible to further configure a set using the **General** and **Requirement Specification** tabs.

Importing requirement sets

Requirement sets can also be imported from other .peet scenarios from the  - icon. This opens a file browser for navigation to the desired scenario which holds the requirement sets to be imported. After choosing of a scenario, a dialog lists all requirement sets available for import which can be selected/deselected with a single click on the item in the list. The selection is finally confirmed with the **Import** - button. The imported requirement(s) are then available in the current scenario with:


- all requirement parameter settings specified in the **General** tab
- maintained source activation status (see next subchapter) for all blocks with matching names in both the source and target scenario – unless the source activation is linked to a MATLAB variable; blocks only available (by name) in the source scenario are set to be active
- maintained requirement values and IDs specified in the **Requirement Specification** tab for all blocks with matching names and matching units; otherwise, no data is imported on that respect

Requirement set overview

Usually, all PEET reports are generated per requirement set (see 6.4.1.1). The  - icon triggers the generation of a report file with requirement specification parameters, requirements values and IDs of all requirement sets present in a scenario in a single file. In addition to general scenario information, one specific sheet is generated for all statistical (“Statistical_Requirements”) and all spectral requirements (“Spectral_Requirements”).

This report file is named [scenarioName]_RequirementOverview and has the file format as specified for all reports in the **File** → **Create Report...** menu.

6.4.2.2.1 General tab

First, a **Name** of the requirement set needs to be defined. This string is used to identify the set in reports or when displaying results in the **Budget Tree View** or **Breakdown Tree View**. A more complete description of a requirement can be realized using the button () which allows including a free text description which is also included in reports.

The first mandatory setting is the selection of a pointing **Error index** whose **Type** can be selected from a drop-down list. This list contains all performance and knowledge indices defined in [AD2] including the extended metrics introduced in [RD16] (the latter are marked with '*'):

- **Absolute Performance Error (APE) / Absolute Knowledge Error (AKE)**
- **Mean Performance Error (MPE) / Mean Knowledge Error (MKE)**
- **Relative Performance Error (RPE) / Relative Knowledge Error (RKE)**
- **Performance Drift Error (PDE) / Knowledge Drift Error (KDE)**
- **Performance Reproducibility Error (PRE) / Knowledge Reproducibility Error (KRE)**

- *Windowed Performance Drift (WPD) / *Windowed Knowledge Drift (WKD)
- *Windowed Performance Residual (WPR) / *Windowed Knowledge Residual (WKR)

Note: The metrics used for one of the performance/knowledge index pairs are identical, e.g. an RPE has the same effect for the evaluation as an RKE. However, it is possible to disable certain PES individually for each index in the **General** tab, e.g. to account for PES which only contribute to a performance index, but not to a knowledge index.

Whenever the transfer routes towards the total error differ significantly between performance and knowledge indices, it is recommended to set up two different PEET scenarios. Otherwise, also parallel transfer routes for performance and knowledge contributions can be realized in combination with respective enabling/disabling of the PES. Then, dependent on the selected pointing error index, up to two further parameters need to be specified in the respective panel:

- The *Window time* (for all indices but APE/AKE) in any [Time]-compatible unit.
- The *Stability time* (for PDE/KDE, PRE/KRE) in any [Time]-compatible unit.
- The dimensionless *Normalized location* between 0 and 1. This parameter is only available when the *Metric *Windowed Relative Stability* (for RPE/RKE) or **Windowed Residual Jitter* (for WPR/WKR) are selected. It describes the reference location within a time-window, i.e. 0 is equivalent to the beginning of a window, 0.5 is equivalent to the window centre and 1 to its end.
The *Metric Windowed Expected Distribution* (corresponding to the Windowed Variance metric for RPE/RKE in [AD2]) and the *Metric *Windowed Expected Jitter* (for WPR/WKR) have no additional parameters.

APE and AKE do not require any parameters at all.

More information about the error indices and metrics (recalling the expressions and frequency domain plots from [AD2] and [RD16]) are also available in the theoretical background section of the online-help.

The **Requirement settings** panel is used to configure how the budget shall be evaluated for the selected index by selecting the *Type* of the requirement.

- **Statistical**
This type of requirement is usually formulated in the following form:

“For the given conditions, the probability shall be greater than or equal to a level of confidence *LoC* in % that an error *e* does not exceed required value *e_r*.”

$$\text{Prob} [|e| < e_r] \geq LoC$$

The conditions for the requirements are expressed by the “statistical interpretation” (S.I.) in [AD1] and [AD2]. This concept is further generalized to a **Domain treatment** concept in PEET (which allows more flexibility in the requirement definition in case more than one ensemble domain is defined).

In the related panel, the treatment of errors in the *Temporal domain* and in the *Ensemble Domain* needs to be specified. Certain combinations of these settings then directly reflect a statistical interpretation (see Table 2). Possible options are **Worst Case** or **Statistical** in both cases.

Table 2: Mapping between “domain treatment” concept in PEET and “statistical interpretation” (SI) in [AD1] and [AD2]

| Temporal domain \ Ensemble domain | Statistical | Worst Case |
|-----------------------------------|-------------|-------------|
| Statistical | Mixed SI | Temporal SI |
| Worst Case | Ensemble SI | -* |

* This combination is not explicitly covered in [AD1] and [AD2] and leads to a very conservative result for the deterministic overall worst-case value.

Worst Case means that only the maximum value (over all times or all ensemble realizations) is taken into account in the evaluation, **Statistical** however accounts for the entire temporal or ensemble probability information.

The **Statistical method** panel allows the selection of the evaluation method for the error contributions. The **Simplified** method evaluates budget values based on mean values, standard deviations and a confidence factor ($n_p = 1, 2, 3$ equivalent to a 1σ -, 2σ - or 3σ -level of confidence of a Gaussian). This method was already realized in the PEET software prototype and is related to the method described in detail in [AD2].

The **Advanced** method in contrary exactly evaluates the error for a given level of confidence (see also [RD7]) by accounting for the entire PDF information available for the error contribution. This significantly improves the accuracy of the result in case dominant non-Gaussian error contributions are present.

After domain treatment and statistical method are defined, the **Level of confidence** in [%] needs to be specified in the respective panel (in case the **Advanced** statistical method is selected).

Note: For a Worst Case – Worst Case combination, it is not possible to specify a level of confidence, as the result is deterministic.

The budget value of the total error $e_{tot,LoC}$ for the given level of confidence LoC in [%] is then determined by:

$$LoC/100 = \int_0^{e_{tot,LoC}} p(|e|) de$$

For the **Simplified** statistical method, the budget value is computed from:

$$e_{tot,LoC} = |\mu_{tot}| + n_p \cdot \sigma_{tot} \quad (n_p = 1, 2, 3)$$

which is only exact for a Gaussian contribution. With this option the **Level of confidence** panel is replaced by a **Confidence factor** panel to define the integer value for n_p .

Independent from the statistical method, the related requirement values to be checked against are defined in the **Requirement settings** tab (fully optional).

Multiple ensemble domains

The provided level of confidence is always related to a combination of temporal and ensemble domain treatment (i.e. for one S.I.). If more than one ensemble domain is defined in the scenario ([Setup](#) → [Ensemble Domains...](#)), each is listed separately in the [Domain treatment](#) panel and can be configured individually.

- This also extends the options in the [Level of confidence](#) panel. It is either possible to select a [Common](#) or an [Individual Level of confidence evaluation](#).
- With the [Common](#) option, the error contributions e_i from N_D different domains i are summed first and then one total error $e_{tot,LoC}$ for the common level of confidence LoC in [%] is determined:

$$LoC/100 = \int_0^{e_{tot,LoC}} p(e) de \quad \text{with} \quad p(e) = P\left(\sum_{i=1}^{N_D} e_i\right)$$

In this case, only the “statistical interpretation” can be different for each domain, the evaluation follows the standard procedure.

- With the [Individual](#) option, the errors $e_{LoC,i}$ for the levels of confidence LoC_i defined for each of the N_D different domains are evaluated. Then these contributions are linearly summed to form the total error $e_{tot,LoC}$:

$$e_{tot,LoC} = \sum_{i=1}^{N_D} e_{LoC,i} \quad \text{with} \quad LoC_i/100 = \int_0^{e_{LoC,i}} p(|e_i|) de_i$$

Tip: More guidelines and hints on this topic are provided in chapter 9.

- For the simplified statistical method, the treatment for the [Individual/Common](#) cases with multiple domains is equivalent. The only difference is that the intermediate results are again just based on the means and standard deviations of the domain contributions.
- [Spectral](#)
This type of requirement is usually formulated in the following form:

“The (amplitude) spectral density of the total error shall be below the function $S_{req}(f)$ for all frequencies between lower bound a and upper bound b .”


$$\left| \sqrt{G_{e_{tot}}(f)} \right| \leq S_{req}(f) \quad \text{for } a \leq f \leq b$$

The only settings which need to be specified in this case are the [Lower bound](#) and the [Upper bound](#) of this [Bandwidth](#) in the respective panel. The functions $S_{req}(f)$ are then individually defined in the [Requirements Specification](#) tab.

Note: Only [PSD](#) and [BLWN](#) type error sources contribute to the budget for a spectral requirement. All other error source contributions are simply ignored.

The [Error source activations](#) panel finally lists all (connected) PES defined in the scenario. It allows disabling individual error sources for a certain requirement set. A disabled PES does not contribute to the total error of a requirement set. This is also indicated in the [Budget Tree View](#) where these PES are greyed out. The status of each

PES can be changed either individually using the checkboxes in the list or entirely using the *All/None* buttons.

Alternatively, the source activation status can also be read from a MATLAB variable. When enabling the respective checkbox () the following options are available:

- **Inactive:** a list (cell) of strings holding the block names of all inactive PES (all other PES get enabled)
- **Active:** a list (cell) of strings holding the block names of all active PES (all other PES get disabled)
- **All:** a cell array of size [numPES,2] holding the name strings of all connected PES in the first dimension and the associated status (0 or 1) for each PES in the second dimension

6.4.2.2.2 Requirement Specification tab

The layout and options in the **Requirement Specification** tab depend on whether **Statistical** or **Spectral** requirements have been selected in the **General** tab.

Tip: The specification of requirement values is completely optional. It is only required for the **Breakdown Tree View** to highlight violations, but not in case only a budget should be computed.

- **Statistical** requirement value specification:
*For the budget evaluation in the **Budget Tree View**, PEET determines the error value $e_{\text{tot,LoC}}$ for the given level of confidence in any case. If additionally a requirement value e_r is explicitly assigned, the following question can be accounted for in the **Breakdown Tree View**:*

“Does the error value $|e_{\text{tot,LoC}}|$ fulfil the requirement?”, i.e.

$$|e_{\text{tot,LoC}}| < e_r$$

*While above is the baseline which is directly accounted for in the results, implicitly (using the generated CDF plots in the **Breakdown Tree View**) also a second question can be faced:*

“What is the probability that the required error is not exceeded?”, i.e.

$$\text{Prob}[e_r]$$

The **Show/hide requirement components** panel provides various options to adapt the information displayed the requirement specification tables:

- Group 1: Values for **Time-constant**, **Time-Random** and/or **Total** errors
- Group 2: Values for **X**, **Y**, **Z** axis and/or for the **Line-of-sight** error

Selections from these two groups can arbitrarily be combined. This only serves to provide a better overview of data already specified, it does **not** enable or disable specifications.

The **Values and IDs** panel can finally be used to input scalar, non-negative requirement values and to associate an ID (which is an arbitrary string) to each value. This is possible

for all evaluation blocks defined in the scenario, i.e. not only for the *Total Error*, but also for every *PEC* block.

It is not necessary to specify the values and IDs for the entire table. The IDs are only used for display and tracking purposes in the *Breakdown Tree View* and reports. Similarly, empty fields of requirement values are ignored for the comparison with budget values. A zero value instead denotes that strictly no error is permitted for an axis/component.

Note: Requirement specification values must be provided in the same units as selected for the corresponding evaluation block (indicated in the table). There is no conversion applied to these values.

Tip: The requirement specification table directly supports MATLAB variable input (i.e. providing a name of a variable that holds the value is possible).

- **Spectral** requirement specification:

For the budget evaluation in the *Budget Tree View*, PEET determines and plots the spectral density of the random process error contribution in any case. If additionally a requirement function $S_{req}(f)$ is explicitly assigned, it can also be plotted and violations can be detected in the *Breakdown Tree View*.

Afterwards, the tables in the *Spectra and IDs* panel can be used to input the requirement functions and to associate an ID (which is an arbitrary string) to each function. This is possible for all evaluation blocks defined in the scenario, i.e. not only for the *Total Error*, but also for every *PEC* block.

Note: The requirement functions must be compatible to the units selected in the corresponding evaluation blocks (as indicated in the table). There is no conversion applied to these functions which are interpreted as being defined in [unit/ $\sqrt{\text{Hz}}$].

The *Representation* of the requirement functions can either be *Analytical* or *Numerical*.

- **Analytical**

For each dimension of an evaluation block, an expression for the requirement function (as function of frequency) and an associated string of an ID can be specified. It is not necessary to specify functions and IDs for the entire table. The IDs are only used for display and tracking purposes in the *Breakdown Tree View* and reports. Similarly, empty entries for requirement functions are ignored in the comparison with the budget.

Tip: The function definition accepts any MATLAB compatible notation (or variable holding such string), e.g.:

$$(f+5) \cdot ^2 ./ \text{sqrt}(f \cdot ^3 + 2 \cdot * f + 3) \quad \text{for} \quad S_{req}(f) = \frac{(f+5)^2}{\sqrt{f^3 + 2f + 3}}$$


Note: f needs to be used as frequency variable in the function and multiplication ($\cdot *$), division ($\cdot /$) and power ($\cdot ^$) operations need to be defined as element-wise, i.e. preceded by a dot.


- **Numerical**

In this case, requirement functions are defined numerically by a matrix of frequencies

and corresponding magnitude values for each dimension of the block. Each frequency is represented by a row in the input table.

The **Block** for which requirements shall currently be specified can be selected from the respective drop-down list. Consequently, for N_f frequency points and N_{ax} functions, in total an $N_f \times N_{ax}$ matrix of data needs to be provided for each evaluation block.

It is not necessary to numerical data and IDs for each block and component. The IDs are only used for display and tracking purposes in the **Breakdown Tree View** and reports. If no requirement data shall be assigned to an entire block, simply leave the data table empty (or remove any existing rows to clear previous data using the -icon). If no requirement data shall be assigned to certain axes of a block, simply fill the respective 'Magnitude' columns with zeros

Alternatively, also MATLAB variables can be used to specify the numerical data using the -icon. In this case, the **Requirement function** data must be provided as cell [dim+1] elements where the first element holds the frequency vector (in Hz) and the remaining elements hold the magnitude vector for each axis. Empty cell elements can be used to indicate that no requirement is assigned to a certain axis.

The associated **IDs** must be provided as cell with [dim] elements containing a string in each element (empty cell elements are supported).

6.4.2.3 Evaluation Settings

The **Evaluation Settings** menu defines several global settings for the scenario which do not depend on the individual requirement sets.

Frequency evaluation settings:

The initial evaluation grid to determine random process (i.e. periodic signal and PSD) variances is specified in the **Frequency evaluation settings** panel. The grid is defined by the resolution per frequency magnitude (**Grid points per decade**) of a log-spaced grid and the magnitudes n of the frequency bounds in 10^n Hz. All periodic signal frequencies are automatically accounted for in the grid within the given bounds.

Note: Random process error contributions **outside** the defined **Low frequency boundary** and **High frequency boundary** are **ignored** in the evaluation.

Tip: Optionally, the initial grid can further be adapted according to the available system information in the menus **Setup** → **Check Frequency Bandwidth...** and **Setup** → **Refine Frequency Grid...**

Tip: In case only spectral requirements are defined in a scenario (in **Setup** → **Scenario Definition**), the frequency bounds (or their "envelope" in case of multiple requirements with different bounds) should be set according to their **Bandwidth** settings as all other contributions are irrelevant.

Evaluation period:

The **Value** for the **Evaluation period** (in any [Time]-compatible unit) also belongs to the requirement specification parameters in [AD2]. This setting has no direct impact on the evaluation of a budget, but can be used for some consistency checks of the PES model definition (e.g. whether a periodic signal "appears" as periodic in the given evaluation period or just as drift in case the evaluation period is much smaller than the periodic signal period).

Line-of-sight axis

The *Selection* of the *Line-of-sight axis* is made from the drop-down list in the respective panel. Possible options are *x-Axis*, *y-Axis* and *z-Axis*. The error evaluation should always be understood as “local at the evaluation block” (*PEC* or *Total Error*), i.e. with respect to the coordinate axis of the block.

Tip: If the line-of-sight error does not correspond to one of the main axes, it can always be represented as main axis by using a suitable *Coordinate Transformation*.

Requirements check:

The *Requirements check* panel contains the option *Highlight if margin is less than X %*. The specified value is used globally in the *Breakdown Tree View* to indicate (by colour-coding) where a budget value is close to the specified requirement value. A zero value implies a simple distinction between “met” and “violated” only.

Units check:

Furthermore, *Checks and conversions* of *Units* can be *disabled* with a left-click on the respective toggle-button (e.g. to avoid numerous unit definitions for a quick-assessment with a data set where the unit consistency is already confirmed or known). In this case, only the units for input model parameters of the non-generic blocks (e.g. *Star Tracker*, *Rigid Plant*) are required, but the output and input of blocks is simply used without any conversion (The expected I/O quantity for these dedicated blocks can be found in chapter 10.26).

Fast-mode:

By default, mean values/standard deviations and PDF plots are generated and displayed for all signal components and all block outputs in the *Budget Tree View*. Apart from the *PEC* and *Total Error* blocks, the computation of this data represents only auxiliary information provided to the user which is not required for the actual budget compilation but consumes a majority of the computation time (especially the generation of the PDF data and the transfer of plot data between GUI and the MATLAB core).

In case computation time is of larger interest than an immediate inspection of detailed results (e.g. for parameter trade-offs or batch-mode operation), the evaluation of all non-PEC statistics can be individually suppressed/enabled in the *Fast-mode* panel per signal component by checking/unchecking the individual checkboxes or using the *All / None* buttons.

Note: This feature has **no effect** on the accuracy of the budget result.

PDF resolution:

As indicated in the previous paragraph, the generation of PDFs from the numerical samples is the driver of the computation time in PEET. In case the order of magnitude of budget values or the rough shape of the error PDF is of major interest – rather than the actual accuracy of budget values – it is possible to switch the *PDF resolution (sample number)* used for the evaluation.

In *Low-Resolution* mode, an order of magnitude less samples are used to represent the error signals (i.e. $\sim 10^5$ samples per component/axis instead of 10^6 in the nominal *High-Resolution* mode) which has a significant impact on the computation time.

Note: This feature **has an effect** on the accuracy of the budget results and should only be used for “quick assessments”.

6.4.2.4 Set Correlations...*

The **Set Correlations...*** menu is used to define the correlation between axes of each error source and between different error sources which are described as **random variables** (or represent an ensemble parameter defined for a PSD).

Correlation is defined in terms of correlation coefficients in the range [-1, 1]. These coefficients represent Spearman coefficients, i.e. rank correlation coefficients which assess how well the relationship between two variables can be described using a monotonic function (see [RD7]).


Available options for correlation depend on the error source definition. Thus, the size of the matrix is automatically derived by the software and only valid combinations are displayed. Furthermore, several tabs are available for different “types” of correlation.

Note: By default, all configurable correlation coefficients are initialized as zero. This is also valid for any newly added error source.

Note: Correlation can only be specified for PES blocks which have a connection to the **Total Error** block. All non-connected blocks are ignored.

The **Temporal** tab contains all time-random error sources where a correlation in time can be specified. The **Ensemble** tab contains all (none-discrete) time-constant error sources and/or (none-discrete) parameters of time-random error sources where a correlation over the ensemble can be specified.

In case more than one ensemble domain has been defined in **Setup → Ensemble Domains...**, the **Ensemble** tab is replaced by one tab for each individual domain with the respective domain name as – per definition – no correlation exists between different domains.

As the correlation matrix is symmetric, only a triangular matrix needs to be specified (with 1's on the main diagonal as every signal is fully correlated with itself). In case the correlation matrix is imported from MATLAB () or Excel (right-click menu, **Import from Excel**), it is convenient to link to a rectangular matrix, where only the **upper triangular part** is read by the software.

The desired correlation can only be realized if the provided correlation matrices are positive semi-definite. This can be checked by the tool using the **Check Feasibility** button. In case the matrix fails this criterion, a feasible matrix which is “close” to the desired correlation is provided (by adapting the matrix eigenvalues, see [RD7]). This alternative can be accepted (**Ok** button) or discarded (**Cancel** button) by the user. In the latter case, a feasible alternative matrix needs to be found manually and provided in the menu.

Note: When PEET is executed in the script-based mode (see chapter 7) and the provided input correlation matrix is infeasible, the feasible “close” matrix is used automatically and a related message is written to the command window. To avoid this, input matrices should be checked for positive semi-definiteness manually in this case before running the script-mode.

6.4.2.5 Set Coherence...*


The **Set Coherence...*** menu is the equivalent of the **Set Correlations...*** menu for **PSD type error sources**. Coherence is a measure for the “similarity” of random processes with

is usually expressed as a function of frequency between 0 (no coherence) and 1 (full coherence). PEET provides a simplified representation using a constant **coherence factor** (i.e. a constant coherence function over all frequencies) to setup the cross-power spectra.

Note: By default, all configurable coherence factors are initialized as zero. This is also valid for any newly added error source.

Note: Coherence can only be specified for PES blocks which have a connection to the *Total Error* block. All non-connected blocks are ignored.

Tip: Using constant coherence factors does not limit the software concerning the definition of cross-spectra between axes of an error source: An **Explicit Cross-spectrum** can always be defined directly in the block mask dialog of the *PES (Time-Random)* block.

As for the correlation matrix, only a triangular matrix needs to be specified (with 1's one the main diagonal as every signal is fully coherent with itself). In case the coherence matrix is imported from MATLAB () or Excel (right-click menu, **Import from Excel**), it is convenient to link to a rectangular matrix, where only the **upper triangular part** is read by the software.

6.4.2.6 Set Phase Relations...*

The **Set Phase Relations...*** menu is the equivalent of the **Set Correlations...*** menu for **periodic error sources**. It allows the definition of an arbitrary phase in any [Angle]-compatible unit and for each axes and frequency of the periodic error sources individually. While the phase of periodic signals is irrelevant for the simplified statistical method in the prototype (based on mean and variance only), it has an influence on the PDF of summed signals.


Note: By default, all configurable phases are initialized as zero. This is also valid for any newly added error source.

Note: Phases can only be specified for PES blocks which have a connection to the *Total Error* block. All non-connected blocks are ignored.

Tip: For two periodic signals at the same frequency, the phase difference $\Delta\varphi$ has a direct relation to the correlation coefficient ρ between two signals, i.e. $\rho = \cos(\Delta\varphi)$. For any pair of periodic signals at different frequencies, the temporal correlation is zero ($\rho = 0$).

By default, all periodic signal contributions are assumed to be in phase (i.e. zero is the reference phase and all periodic signals are interpreted as cosine functions). There is no restriction on the specified phase value itself, but several input options are disabled in the "phase matrix":

- 1D signals are defined by the x-axis entry, y-and z-axis are greyed out. This mapping is only artificial for the definition in a common matrix, there is no "physical" relation to an x-axis.
- The model for the periodic error contribution of the *Reaction Wheel* error source blocks implies a 90° phase shift between the x- and y-axis components. For that reason, it is not possible to set these values independently and the x-axis value is taken as reference.

Nevertheless, in case the phase matrix is imported from MATLAB () or Excel (right-click menu, **Import from Excel**), it is convenient to link to a rectangular matrix, where only the relevant entries are read by the software automatically.

6.4.2.7 Check Frequency Bandwidth...*

The **Check Frequency Bandwidth...*** menu can be used to provide an overview of all frequency related information in the scenario and to adapt the default settings (in **Setup** → **Evaluation Settings**) based on available system information.

Note: This menu is only relevant if random process type error sources (**Periodic**, **BLWN** or **PSD**) are present in the scenario.

The menu dialog is split into two panels. The **Frequency overview** panel on top visualizes the basic frequency information of the scenario (see Figure 6-6), in particular:

- The frequency bounds ($f_{e,min}$, $f_{e,max}$) currently selected for the evaluation of random process contributions (**Setup** → **Evaluation Settings**) as red vertical lines (dashed)
- The “minimum frequency” $1/T_{eval}$ related to the **Evaluation period** (**Setup** → **Evaluation Settings**) of the requirement as green vertical line (dashed).
- The frequency range of all **Spectral** requirements (**Setup** → **Evaluation Settings**) as magenta horizontal lines (if defined).
- The frequency range of **Numerical** and **Analytical** PSD error sources as black horizontal lines
- The frequency of poles (x) and zeros (o) of PSD error sources defined as **LTI Model** within the individual bandwidth for the sources (as horizontal line)
- The frequencies of all **Periodic** error sources grid on the horizontal axis (marked as black diamonds)
- The frequency of poles (x) and zeros (o) of all dynamic systems present in the scenario.

This overview can be used to quickly detect mismatches between the frequency range defined for the evaluation and the frequency ranges defined in the requirement. The panel contains several tabs where either all information is shown entirely in one plot (**All**) or selected information is extracted (for **PES**, **Systems** and **Spectral Requirements**).

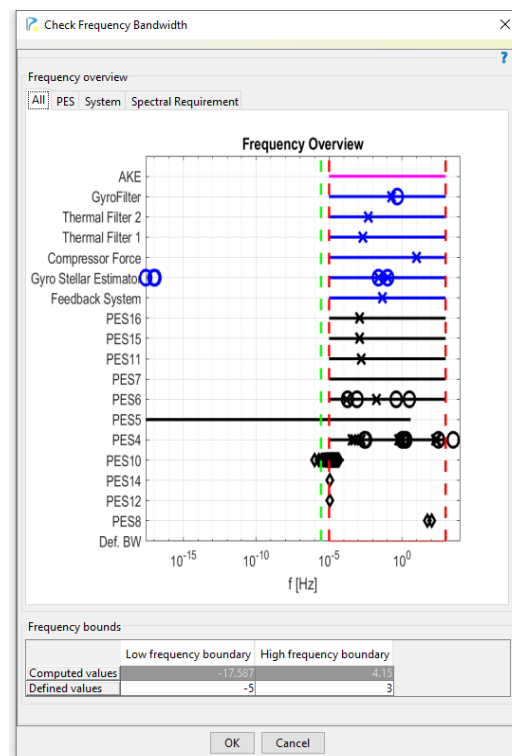


Figure 6-6: The Check Frequency Bandwidth menu

The lower panel provides numerical values for the current state of the **Frequency bounds**.

The first row in the table contains the **Computed values** for the bandwidth (in $[10^n \text{ Hz}]$) which cannot be edited. These bounds correspond to the envelope of all detected frequencies (plus an additional margin if a bound is determined by a pole or zero - which can change the slope of an LTI model).

The second row (**Defined values**) initially contains the default boundaries specified in the **Setup** → **Evaluation Settings** menu. It can be used to adapt these settings with to the suggested bounds or any manual bound modifications desired by the user. Any changes are applied using the menu's **OK** button (which leads to a re-initialization when re-computing a scenario).

In this case, also the number of grid points is adapted to the defined frequency range regarding the specified density and the bound values in the **Setup** → **Evaluation Settings** menu are updated accordingly. **Cancel** leaves all settings unchanged.

Note: The suggested frequency bounds should not be considered as ideal or optimal solution for a scenario. LTI models can have artificially located zeros and poles at very high/low frequency (e.g. the transmission zeros for the “Gyro Stellar Estimator” system model in Figure 6-6 above) which are irrelevant for the physical behaviour to be described. This can however not be detected by the tool.

6.4.2.8 Refine Frequency Grid...*

The **Refine Frequency Grid...*** menu can be used to refine the default log-spaced frequency grid (with resolution as specified in the between the **Setup** → **Evaluation Settings** menu), i.e. to adapt the grid density around “points of interest”.

Note: This menu is only relevant if random process type error sources (**Periodic**, **BLWN** or **PSD**) are present in the scenario.

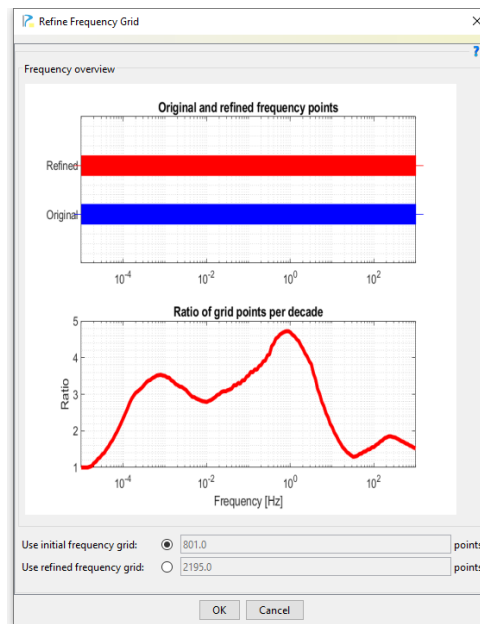


Figure 6-7: The Refine Frequency Grid menu

In particular, these points are all poles and zeros in LTI PES and system models within the selected evaluation bandwidth, as these points can represent peaks or changes in a slope of the spectrum or response. The figure in the **Frequency overview** panel shows both the initial grid and the refined grid (top part) and the ratio of grid points (per decade) of the refined grid with respect to the initial grid.

The radio buttons below can be used to select either the initial or the refined frequency grid for the evaluation. The selection is applied using the menu’s **OK** button. **Cancel** leaves all settings unchanged.

6.4.2.9 Preferences

The **Preferences** menu contains general settings which are not related to a specific scenario, but to the software behaviour itself. Thus, all available options explained below apply ‘globally’ to all scenarios.

Automatic backup before version conversion of the scenario

When loading a scenario created with an older PEET version (>V1.0) with a more recent version of PEET, the scenario data (XML) files are updated automatically to the new version. By default, also a backup of the original scenario files is created before the update and stored in a .zip file in the scenario folder. Using the respective checkbox, the backup generation can be suppressed.

Note: It is not possible to convert from a more recent version back to an older version of PEET.

Color highlighting of open scenarios

This feature is especially useful when working with multiple scenarios at a time. It simplifies the identification of all windows and dialogs which belong to a certain scenario by associating a specific color scheme to each. The following options are available:


- **None:** No distinction between different scenarios is applied.
- **Window Icons:** A specific color indicator is added next to the PEET logo in each window/dialog of a scenario.
- **Window Icons and Stripes:** Also the menu bars of windows and dialogs gets highlighted in a scenario specific color.

Recent units list

Recently created custom units are automatically made available in all unit drop-down lists for quick selection – in case they are compatible with the permitted unit of a block or parameter. In case certain or all recent units shall no longer be present, the entries in this list can be removed or entirely be reset using the *Remove/Clear All* buttons. Further, it is possible to define the *Maximum number of stored units* present in the list.

Plot options


This panel allows to refine the treatment and display of generated plots:

- *PDF plots* can either be displayed as **bar** plot or as normal **line** plots. Changes to the settings are not applied immediately in the GUI, but after the next evaluation of a scenario.
- *Correlation plots* can either be displayed as **scatter** plots or as **heatmap**; this option only applies to plots to be generated by the MATLAB core, i.e. only scatter plots are displayed in the GUI.
- *Save format* allows selecting one or more supported image file types (**.fig**, **.png**, **.jpg**, **.bmp** or **.pdf**) for MATLAB plots with are either saved automatically during the report generation (see chapter 6.4.1.1) or manually () in a **Plot** window (see also chapter 6.8). For saved GUI plots, the file format can explicitly be selected in the respective dialog.
- *Overwrite:* All saved plot follow a specific naming scheme (including e.g. the associated block name). The checkbox determines the behaviour in case a plot file name already exists in the scenario \output folder when a file is saved. If checked, the existing file is overwritten; if unchecked, the existing plot file is preserved and a time stamp of the form **_yyyymmddTHHMMSS** is appended to the new plot file name.

6.4.3 Database Menu

The **Database** menu is used to display all existing building blocks in PEET and to manage user-defined or -configured block groups.

Show Block Database

This item is the menu-equivalent of the -icon in the toolbar to open the **Database Browser** (see chapter 6.5).

Export Block...

Any block (with all current parameter settings) or even entire subsystems built by multiple blocks can be exported as one user-defined block to the [Block Database](#). This is realized following the steps below:

- Copy the block (or the desired system of blocks with all connections) into a [Container](#) block and add as many [Input Ports/Output Ports](#) as required to the [Container](#).
- Select the [Container](#) with a left-click and open the [Database](#) → [Export Block...](#) menu.
- Define an arbitrary name for the [Block type](#) and assign the block to a [Category](#). It is also possible to create a [New Category](#) by selecting the respective item in the drop-down list. This enables the input for the [Category name](#).
- Confirm the export with the [OK](#) button.

After these steps, the block is permanently saved in the [Block Database](#) (in the tab of the selected [Category](#)).

Note: The block database is defined by an XML file `block_database.xml` which is located in a subfolder of the system's user directory (for Windows: `[userdir]\AppData\Roaming\Astos Solutions\PEET`). After installation, it is first created from a copy from the reference `block_database.xml` located in the bin folder of the PEET installation. All modifications are only applied to the local copy in the user directory.

Delete User-Defined Block...

This menu item allows the removal of previously exported user-defined blocks. It opens a dialog where the [Block type](#) to be deleted needs to be selected from a drop-down list which contains all user-defined blocks saved in the local `block_database.xml`. A left-click on the [OK](#) button confirms the permanent deletion of the block (group).

6.4.4 Analysis Features Menu

The [Analysis Features](#) menu provides different additional analyses which can be applied error contribution block data ([PEC](#) and [Total Error](#)). These analyses are all “external”, i.e. they do not modify the block output signals which are routed through a scenario itself.

6.4.4.1 Post-Processing

In the context of PEET, “post-processing” must be understood as any additional analysis operation applied to nominal budget results at each error contribution block ([PEC](#) and [Total Error](#)), i.e.

- the PDF of time-constant, time-random and total error contributions in PEC coordinates (x, y, z and/or line-of-sight) for **statistical** requirements
- the power spectral density in PEC coordinates (x, y, z) for **spectral** requirements.

Such operation could be, for instance, a conversion of data to another coordinate representation or applying a specific function or complete algorithm the axes data input (somewhat similar to the ‘fcn’ or ‘MATLAB Function’ blocks in MATLAB Simulink).

The **Post-Processing** menu provides dedicated analyses with a predefined parameterization but also permits including user-defined algorithms supported by specific templates.

The menu dialog is shown in Figure 6-8. It provides a separated tab for each error contribution block present in the scenario which allows defining or omitting specific analyses individually for each block.

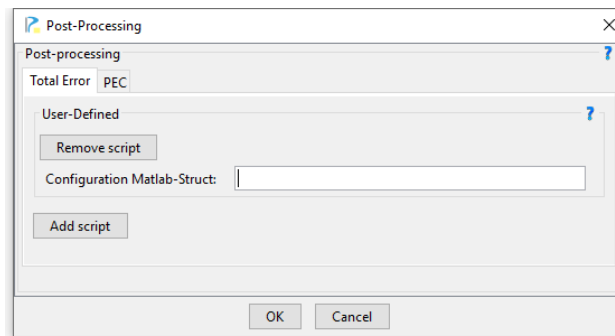


Figure 6-8: Post-Processing menu dialog

A click on the **Add script** button opens the **Select script type** dialog which contains a drop-down list with all possible post-processing analyses for the selected block. The options available from the drop-down further depend on the dimension and unit specified for a block (see next subchapters). After confirming a selection with **OK**, the menu displays the analysis parameters in a subpanel.

It is possible to enable multiple analyses for any block (also of the same type) by repeated use of the **Add script** button which appends a new parameter panel below existing ones. Analyses can also be removed again individually using the **Remove script** button in the respective panel.

Closing the menu dialog with **OK** confirms all current settings. When evaluating a budget via the **▶** - or **▶▶** -buttons, the analysis are automatically performed and results are presented in the **Breakdown Down Tree** on additional tabs for each defined element.

Further, post-processing results are also included in Excel reports if the respective flag **Post-processing** flag is checked in the **Results** panel of the **File → Create Report...** menu (see chapter 6.4.1.1). In this case, one additional sheet is appended in the report for each analysis element specified.

6.4.4.1.1 Azimuth/Elevation

This post-processing element converts pointing errors in "Cartesian coordinates" (i.e. around x, y and z-axis) to respective azimuth and elevation errors with respect to a given reference attitude.

The analysis is available from the drop-down list in the **Select script type** dialog for each PEC block with:

- **3D Signal dimension**
- [Angle]-compatible **Unit**

It is applied to all PEC contributions (time-constant, time-random and total) and all domains for any **statistical** requirement.

Analysis Parameters

The following settings need to be specified in the **Azimuth/Elevation** panel:

Reference azimuth / elevation

The scalar reference angles (in any [Angle]-compatible unit) with respect to which w.r.t. the x/y/z error angles shall be analysed. Azimuth and elevation are defined relative to the selected line-of-sight axis of the block reference frame.

Azimuth / Elevation error requirement

The scalar requirement value for the respective error angle in any [Angle]-compatible unit. The permitted azimuth range is equivalent to $[0^\circ, 360)$, the permitted elevation range equivalent to $[-90^\circ, 90^\circ]$.

It is also possible to specify a requirement value separately for each of the n_{Req} statistical requirement sets defined in the **Setup** → **Scenario Definition** menu. In this case, the name of a variable representing an $[n_{Req} \times 2]$ cell array must be provided. The first dimension of the cell array contains the strings of all requirement set names and the second dimension the respective scalar requirement values (individual values can also be empty).

Azimuth / Elevation error requirement ID

An optional string for the requirement IDs assigned to above requirement values.

As for the requirement values, also an ID for each requirement set can be defined by providing an $[n_{Req} \times 2]$ cell array. In this case, the second dimension holds the ID strings associated to each requirement.

Tip: MATLAB variables and notation can be used directly used in all parameter input fields.

Analysis Output

Analysis results are displayed in the **Breakdown Tree View** in a tab named **Azimuth/Elevation**. If multiple analyses of that type are defined for the block, they are numbered consecutively. If multiple domains are present, each is provided on a different tab (the tab name is extended by “@[domainName]” in this case).

The information present is equivalent to the one displayed for the standard Cartesian results, i.e.:

- a table holding the budget values for time-constant, time-random and total contributions in comparison to the requirement values and the level of confidence which was used for the evaluation
- a CDF plot of the absolute value (of the azimuth and elevation errors)

When generating an Excel report, the same information is provided and plots are automatically saved and linked to the report.

In both cases, the reference azimuth/elevation and selected line-of-sight axis are provided as additional descriptive information.

Background

The algorithm applies the following steps:

- Express the reference azimuth φ_{Ref} and elevation θ_{Ref} as components of a unit vector \mathbf{u}_{Ref} in the block frame, i.e. with components (assuming the z-axis as line-of-sight, for any other LoS direction selected in the **Setup** → **Evaluation Settings** menu, the axes are permuted accordingly, see Figure 6-9):

$$u_{x,\text{Ref}} = \cos(\theta_{\text{Ref}}) \cdot \cos(\varphi_{\text{Ref}})$$

$$u_{y,\text{Ref}} = \cos(\theta_{\text{Ref}}) \cdot \sin(\varphi_{\text{Ref}})$$

$$u_{z,\text{Ref}} = \sin(\theta_{\text{Ref}})$$

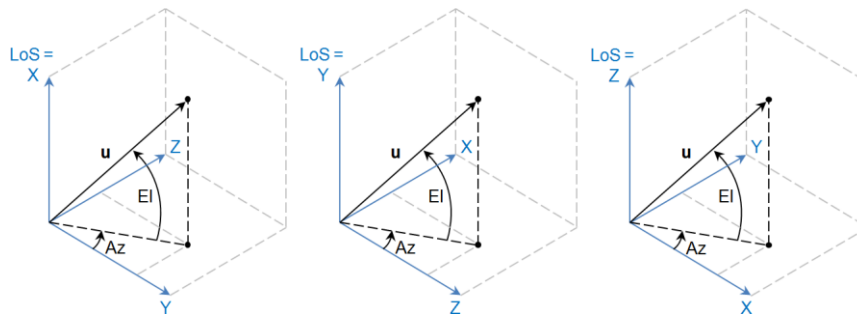


Figure 6-9: Azimuth/elevation definition dependent on selected LoS-axis

- Create a direction cosine matrix $\mathbf{T}(e_x, e_y, e_z)$ for each realization of the error angle input
- Compute the erroneous direction vector \mathbf{u}_e by multiplying \mathbf{T} and \mathbf{u}_{Ref}
- Convert the erroneous direction vector back to azimuth/elevation, i.e. (assuming again the z-axis as line-of-sight, and permutation for other cases):

$$\varphi_e = \tan^{-1}(u_{y,e}, u_{x,e})$$

$$\theta_e = \tan^{-1}(u_{z,e}, \sqrt{u_{x,e}^2 + u_{y,e}^2})$$

- Compute the PDF (and CDF) of the absolute value of the difference between reference azimuth/elevation and the disturbed azimuth/elevation for each realization (i.e. a vector respectively).

$$p(|\varphi_e - \varphi_{\text{Ref}}|)$$

$$p(|\theta_e - \theta_{\text{Ref}}|)$$

Note: Whether these PDFs describe a temporal or an ensemble behaviour depend on the statistical interpretation of the requirement set.

6.4.4.1.2 Coverage (BPE)

This post-processing element represents an analysis feature applicable to geostationary telecommunication missions or any application where a beam pointing error (BPE) with respect to a given reference latitude/longitude on Earth is of interest.

The model assumes a geostationary orbit and a block reference coordinate frame where the x/y/z axes correspond to roll/pitch/yaw respectively. It provides an error budget for the half-cone pointing errors with respect to the direction from the satellite to the provided reference location on Earth and error budgets for an alternative representation of attitude errors in terms of North/South and East/West pointing error angles.

The analysis is available from the drop-down list in the [Select script type](#) dialog for each PEC block with:

- [3D Signal dimension](#)
- [Angle]-compatible [Unit](#)

It is applied to all PEC contributions (time-constant, time-random and total) and all domains for any **statistical** requirement

Analysis Parameters

The following settings need to be specified in the [Coverage \(BPE\)](#) panel:

[Reference pointing direction longitude / latitude](#)

The scalar reference pointing direction sub-point angles in any [Angle]-compatible unit. The permitted latitude range is equivalent to $[-90^\circ, 90^\circ]$, the permitted longitude range equivalent to $[-180^\circ, 180^\circ]$.

[Satellite longitude](#)

The longitude of the satellite in any [Angle]-compatible unit. The longitude must be provided in a range equivalent to $[-180^\circ, 180^\circ]$.

[Half-cone BPE requirement](#)

The scalar requirement value for the half-cone beam-pointing error in any [Angle]-compatible unit. The permitted range is equivalent to $[0^\circ, 180^\circ]$.

It is also possible to specify a requirement value separately for each of the n_{Req} statistical requirement sets defined in the [Setup → Scenario Definition](#) menu. In this case, the name of a variable representing an $[n_{Req} \times 2]$ cell array must be provided. The first dimension of the cell array contains the strings of all requirement set names and the second dimension the respective scalar requirement values (individual values can also be empty).

[Half-cone BPE requirement ID](#)

An optional string for the requirement ID assigned to above requirement value.

As for the requirement values, also an ID for each requirement set can be defined by providing an $[n_{Req} \times 2]$ cell array. In this case, the second dimension holds the ID strings associated to each requirement.

[Half-cone BPE level of confidence \[%\]](#)

An optional scalar level of confidence to be used for the evaluation of the BPE. If not provided, the respective value(s) defined for the evaluated requirement set in the [Setup → Scenario Definition](#) menu is used.

Tip: MATLAB variables and notation can be used directly used in all parameter input fields.

Analysis Output

Analysis results are displayed in the [Breakdown Tree View](#) in a tab named [Coverage \(BPE\)](#). If multiple analyses are defined of that type for the block, they are numbered

consecutively. If multiple domains are present, each is provided on a different tab (the tab name is extended by “@[domainName]” in this case).

The information panel contains two sub-panels:

- The **NS-EW errors** panel provides a PDF plot of the attitude errors expressed as North/South and East-West angles and lists the derived yaw coupling coefficients that were used for the conversion
- The **Half-cone BPE** panel provides a tabular overview of the time-constant, time-random and total beam pointing error budget values vs. its requirement value (if specified). Further, the beam pointing error CDF is plotted and the latitude/longitude parameters for the analysis are displayed as additional descriptive information.

When generating an Excel report, the same information is provided and plots are automatically saved and linked to the report.

Background

The model assumes that x, y and z error signals of the block correspond to roll, pitch and yaw angle errors respectively and that the satellite is in a geostationary orbit.

Under these assumptions, first the coupling coefficients of the yaw movement to North/South (K_{NS}) and East/West (K_{EW}) directions can be computed from the analysis parameters:

$$K_{NS} = \frac{(\cos(l) \sin(L_C - L_S))^2}{\sqrt{\left(\frac{R_0}{R_E}\right)^2 + 2(1 - \cos(l) \cos(L_C - L_S)) \left(1 + \left(\frac{R_0}{R_E}\right)\right)}}$$

$$K_{EW} = \frac{\sin^2(l)}{\sqrt{\left(\frac{R_0}{R_E}\right)^2 + 2(1 - \cos(l) \cos(L_C - L_S)) \left(1 + \left(\frac{R_0}{R_E}\right)\right)}}$$

where L_C is the longitude the reference pointing direction sub-point, L_S is the satellite longitude, l is latitude the reference pointing direction sub-point, R_E is the equatorial radius of the Earth (6378.137 km) and R_0 is the distance from the satellite to the sub-satellite point (i.e. 35786 km for a geostationary orbit).

The NS/EW error angles e_{NS} and e_{EW} are computed from the x/y/z error angles e_x , e_y and e_z using these coefficients:

$$e_{NS} = \sqrt{e_x^2 + (K_{NS} \cdot e_z)^2}$$

$$e_{EW} = \sqrt{e_y^2 + (K_{EW} \cdot e_z)^2}$$

The half-cone beam pointing error e_{BPE} is then computed using the NS/EW angles:

$$e_{BPE} = \sqrt{e_{NS}^2 + e_{EW}^2}$$

The PDFs $p(\mathbf{e}_{NS})$, $p(\mathbf{e}_{EW})$ and $p(\mathbf{e}_{BPE})$ of these angles are obtained by evaluating above expressions for each realization of the error angles and computing the respective histograms.

Note: Whether these PDFs describe a temporal or an ensemble behaviour depend on the statistical interpretation of the requirement set.

6.4.4.1.3 Coverage (Single-Spot)

This post-processing element represents an analysis feature applicable to telecommunication missions or any application where a single-spot antenna pointing performance for a given coverage area is of interest. Both the performance of specific points inside the coverage area as well as the overall spot performance is evaluated.

The coverage area is defined as a set of azimuth/elevation angles as seen from the satellite, i.e. defined with respect to the line-of-sight axis selected in the [Setup → Evaluation Settings](#) menu (chapter 6.4.2.3), see Figure 6-9. It can be provided as contour defined by vertex points (automatic generation of analysis grid points inside the contour) or by providing explicit points inside the coverage area which shall be analysed.

The analysis is available from the drop-down list in the [Select script type](#) dialog for each PEC block with:

- [3D Signal dimension](#)
- [Angle]-compatible [Unit](#)

It is applied to all PEC contributions (time-constant, time-random and total) and all domains for any **statistical** requirement.

Analysis Parameters

The following settings need to be specified in the [Coverage \(Single-Spot\)](#) panel:

[Grid definition](#)

This selection determines how the grid inside the coverage grid to be analysed is defined. Two options are available from the drop-down list:

- [Contour](#)

In this case, the coverage grid is initialized by first providing a set of N [Coverage contour vertices](#) as [Nx2] matrix. Each row in the matrix defines one vertex as azimuth-elevation pair in any [Angle]-compatible unit. The permitted azimuth range is equivalent to [0°, 360), the permitted elevation range equivalent to [-90°, 90°] for each vertex. Further, a non-negative integer number M of [Grid points](#) (per direction) defines the density of the grid inside the contour, i.e. a value of 10 leads to a 10x10 rectangular base grid. Before the rectangular grid is generated, the contour vertices are first converted to direction vectors in the block frame and projected to the plane perpendicular to the line-of-sight. The grid is then overlaid to the coverage contour (from min. to max. of the point coordinates in the projected plane) and all points within or on the contour are used for the analysis (i.e. the effective number of used grid points is smaller than M^2 for a non-rectangular contour, see Figure 6-10).

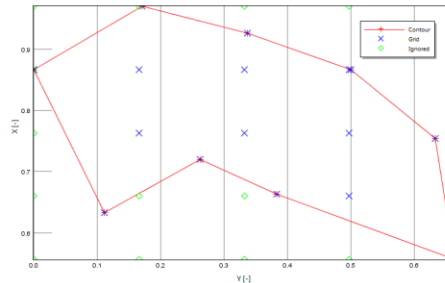


Figure 6-10: Exemplary contour in Az/EI-Map and M = 5

All vertices provided are always included in the analysed grid. Thus, for $M = 0$, only the vertices are taken into account. If the vertices represent a line, then the grid points are placed along the line with equal spacing. If only a single vertex is provided, only one point is analysed.

- Explicit**
 In this case, the grid points to be used for the analysis is directly provided by the *Spot (azimuth/elevation)* input which is an $[N \times 2]$ matrix of azimuth-elevation pairs. The permitted azimuth range is again equivalent to $[0^\circ, 360^\circ)$, the permitted elevation range equivalent to $[-90^\circ, 90^\circ]$ for each grid point.

Note: An overall grid point number of 100 should not be exceeded. If exceeded, the number of samples used for each spot to generate the overall spot performance is reduced automatically for memory and performance reasons.

Single-spot requirement

The scalar half-cone requirement value in any [Angle]-compatible unit and in an equivalent range of $[0^\circ, 180^\circ]$ to be used for the evaluation of PDFs.

It is also possible to specify a requirement value separately for each of the n_{Req} statistical requirement sets defined in the **Setup** → **Scenario Definition** menu. In this case, the name of a variable representing an $[n_{Req} \times 2]$ cell array must be provided. The first dimension of the cell array contains the strings of all requirement set names and the second dimension the respective scalar requirement values (individual values can also be empty).

Single-spot requirement ID

An optional string for the requirement ID assigned to above requirement value.

As for the requirement values, also an ID for each requirement set can be defined by providing an $[n_{Req} \times 2]$ cell array. In this case, the second dimension holds the ID strings associated to each requirement.

Level of confidence [%]

The level of confidence to be used for the evaluation of the single-spot error analysis.

Tip: MATLAB variables and notation can be used directly used in all parameter input fields.

Analysis Output

Analysis results are displayed in the **Breakdown Tree View** in a tab named **Coverage (Single-Spot)**. If multiple analyses are defined of that type for the block, they are numbered consecutively. If multiple domains are present, each is provided on a different tab (the tab name is extended by “@[domainName]” in this case).

The information panel contains three sub-panels:

- The **Overall spot performance** panel provides the budget values and CDF plots (for the time-constant, time-random and total contributions) in terms of elevation, azimuth and half-cone error. For the latter, also a comparison to the requirement values is provided (if specified).
- The **Grid point performance overview** panel provides a plot of the budget values (in terms of azimuth, elevation and half-cone error) individually for each grid point. The applied level of confidence and the selected line-of-sight axis is available as additional descriptive information.
- The **Azimuth-Elevation map** panel provides a plot visualizing the coverage contour and the used grid points

When generating an Excel report, the same information is provided, but the plot in the **Grid point performance overview** panel is replaced by a table with equivalent content. All plots are automatically saved and linked to the report.

Background

A **Grid definition** by **Contour** first requires the detection of grid points inside the area encircled by the coverage contour and on the contour boundaries. For this purpose, only standard MATLAB functions (e.g. 'inpolygon', 'polyarea') are used.

The procedure to determine the azimuth and elevation errors is identical to the one already described in section 6.4.4.1.1, but repeatedly executed for each grid point. The half-cone errors are computed for each grid point using the dot-product between the reference unit direction vector \mathbf{u}_{Ref} and the unit direction vector \mathbf{u}_e with the attitude error vector applied, i.e.

$$e_{\text{half-cone}} = \cos^{-1}(\mathbf{u}_e^T \mathbf{u}_{\text{Ref}})$$

Further, above operations are executed for each realization of the x/y/z attitude angle vector which leads to vectors of attitude errors in terms of azimuth, elevation and half-cone errors for each grid point. Then, the error PDFs are evaluated separately for each grid point and each of the three vectors.

The overall spot performance is computed by appending the half-cone error vectors (azimuth and elevation vectors respectively) errors of each spot into single large error vector. Then, the PDF of this large vector is computed.

This algorithm implies the computation of a large number of histograms (3 per grid point and domain) and computing a histogram from a large dataset (for the overall spot performance) which can have a significant impact on memory consumption and computation time (in the range of several minutes).

Thus, it is recommended to reduce the grid point number as far as possible (< 100 as rule of thumb). To avoid memory conflicts, the samples used to construct the overall spot PDF from the data of each spot is internally restricted. If the length of the overall spot PDF vector would exceed a length of to 10^8 , only a subset of the error data from of each spot is used for the combination to ensure that this limit is not exceeded.

6.4.4.1.4 Coverage (Multi-Spot)

This post-processing element represents an analysis feature applicable to telecommunication missions or any application where a multi-spot antenna pointing performance is of interest.

The orientation of each spot in the antenna array is defined as azimuth/elevation pair as seen from the satellite, i.e. defined with respect to the line-of-sight axis selected in the [Setup → Evaluation Settings](#) menu (chapter 6.4.2.3), see Figure 6-9. The worst-case thermo-elastic deformation of the antenna array can be taken into account in the model.

The analysis is available from the drop-down list in the [Select script type](#) dialog for each PEC block with:

- [3D Signal dimension](#)
- [Angle]-compatible [Unit](#)

It is applied to all PEC contributions (time-constant, time-random and total) and all domains for any **statistical** requirement.

Analysis Parameters

The following settings need to be specified in the [Coverage \(Multi-Spot\)](#) panel:

[Spots \(azimuth/elevation\)](#)

An [Nx2] matrix defining each of the N spots by an azimuth-elevation pair in any [Angle]-compatible unit. The permitted azimuth range is equivalent to [0°, 360), the permitted elevation range equivalent to [-90°, 90°]. If not already included in the input matrix, a reference spot along the line-of-sight axis of the block frame (i.e. 90° elevation, 0° azimuth) is automatically generated.

[Reference spot requirement](#)

The scalar half-cone requirement value in any [Angle]-compatible unit for the performance of the reference spot along the line-of-sight axis. The value must be provided in an equivalent range of [0°, 180°] and applies to the total error contribution.

It is also possible to specify a requirement value separately for each of the n_{Req} statistical requirement sets defined in the [Setup → Scenario Definition](#) menu. In this case, the name of a variable representing an [$n_{Req} \times 2$] cell array must be provided. The first dimension of the cell array contains the strings of all requirement set names and the second dimension the respective scalar requirement values (individual values can also be empty).

[Reference spot requirement ID](#)

An optional string for the requirement ID assigned to above requirement value.

As for the requirement values, also an ID for each requirement set can be defined by providing an [$n_{Req} \times 2$] cell array. In this case, the second dimension holds the ID strings associated to each requirement.

[Worst-case spot requirement](#)

The scalar half-cone requirement value in any [Angle]-compatible unit for the resulting spot with the worst performance. The value must be provided in an equivalent range of [0°, 180°] and applies to the total error contribution.

As for the reference spot, an [$n_{Req} \times 2$] cell array can be used to specify requirement values for each requirement set.

Worst-case requirement ID

An optional string for the requirement ID assigned to above requirement value.

As for the reference spot, an $[n_{Req} \times 2]$ cell array can be used to specify requirement IDs for each requirement set.

Level of confidence [%]

The level of confidence to be used for the evaluation of the performance for each spot.

Maximum daily amplitude

This scalar non-negative value determines the maximum daily amplitude of the thermo-elastic errors acting on the multi-spot antenna array. It is defined for the spot with the largest nominal radial distance from the reference spot and can be provided in any [Angle]-compatible unit.

TE-scale definition

This selection determines how the thermo-elastic (TE) deformation of the multi-spot antenna is defined.

- **Radial Distance**

In this case, the maximum daily TE amplitude is directly applied to “farthest” spot from the reference spot, i.e. such that their angular distance increases in elevation by the specified maximum daily amplitude value.

For all other spots, the TE amplitude value is scaled with the ratio of each spots’ radial distance from the reference spot and the radial distance of the farthest spot.

- **Explicit**

In this case, dimensionless *TE scale factors* in the range [-1,1] are explicitly provided as $[N \times 1]$ vector. Each entry in the vector is used to scale the maximum daily TE amplitude. The result is then applied individually to each of the N spots such that their radial distance from the reference spot is increased or decreased accordingly.

Tip: MATLAB variables and notation can be used directly used in all parameter input fields.

Analysis Output

Analysis results are displayed in the [Budget Tree View](#) in a tab named **Coverage (Multi-Spot)**. If multiple analyses are defined of that type for the block, they are numbered consecutively. If multiple domains are present, each is provided on a different tab (the tab name is extended by “[domainName]” in this case).

The information panel contains five sub-panels:

- The **Reference spot performance** panel provides the reference spot budget values and CDF plots (for the time-constant, time-random and total contributions) in terms of elevation, azimuth and half-cone error. For the latter, also a comparison to the requirement values is provided (if specified).
- The **Worst-case spot performance** panel provides the budget values (for the time-constant, time-random and total contributions) for the worst-case spot in terms of elevation, azimuth, thermo-elastic error and half-cone error. For the latter, also a comparison to the requirement values is provided (if specified).
- The **Spot performance overview** panel provides a plot of the budget values (in terms of azimuth, elevation and half-cone error) individually for each spot. The applied level of confidence and the selected line-of-sight axis is available as additional descriptive information.

- The **TE error contribution overview (elevation)** panel provides a plot of the thermo-elastic error contributions in terms of elevation individually for each spot.
- The **Azimuth-Elevation map** panel provides a plot visualizing the spot locations

When generating an Excel report, similar information is provided, but the overview plots in the **Spot performance overview** and **TE error contribution overview (elevation)** panels are replaced by tables with equivalent content. All plots are automatically saved and linked to the report.

Background

The procedure to determine the azimuth, elevation and half-cone errors for each spot is similar to the one described for the single-spot case described in section 6.4.4.1.3 (with each 'spot' in this model equivalent to a 'grid point' of the single spot case).

The only difference is that not only the x/y/z attitude error input, but also the thermo-elastic errors affect the de-pointed unit direction vector \mathbf{u}_e . Thus, first the elevation errors due to the thermo-elastic deformation are applied to the nominal spot direction unit vectors before further applying the rotation due to the x/y/z error angles. As the TE effect implemented in a worst-case sense, i.e. using the maximum daily amplitude, its contribution is constant (but different for each spot). Consequently, the unit vectors with the TE effect applied to each spot are determined once already during the scenario initialization and passed as a parameter to the core function to avoid unnecessary repetition of computation steps during the scenario evaluation.

6.4.4.1.5 User-Defined

Different to the other analyses, this option type is available from the drop-down list in the **Select script type** dialog for all PEC blocks, i.e. with:

- 1D or 3D **Signal dimension**
- any block **Unit**

It uses a generic interface which can be used to include any kind of user-defined algorithm which shall be applied to the error signal available at a PEC block (similar to a 'MATLAB Function' blocks in MATLAB Simulink) in the evaluation routine of a scenario. Further, results can be automatically be displayed in the GUI and/or included in Excel reports in tabular format or as plot.

If selected, only one parameter **Configuration Matlab-Struct** needs to be provided which is the variable name of a MATLAB structure which must be available in the MATLAB workspace when initializing a scenario. Further, a MATLAB function m-file is required which contains the user-defined algorithm.

Their content and format are described in the following subsections.

Tip: Templates exist for both the post-processing configuration structure and function m-file (`postProcConfigStructTemplate.m` and `postProcFunctionTemplate.m`) in the `\templates` subfolder of the PEET installation which can be copied for reuse to any location.

Post-Processing Configuration Structure

The configuration structure [`configStruct`] which can have any variable name supported by MATLAB contains the most essential parameters for the execution of a user-defined

post-processing analysis. The mandatory and optional fieldnames are explained below, but also directly in the template file `postProcConfigStructTemplate.m`.

Tip: To ensure that the configuration structure is always present when needed, it is recommended to include it in a *MATLAB initialization script* and set the path to this script in the **Setup → Evaluation Settings** menu (see chapter 6.4.2.3).

The following three field names are **mandatory**:

- `configStruct.evalName`
An arbitrary name string for the post-processing analysis. It is used for internal identification of the element, but also defines the name of the tab (or sheet) where the results get displayed in the **Breakdown Tree View** (or Excel report).

Note: Acting as an identifier, the name string must be unique for each PEC block in case multiple analyses are defined.

Note: The name string must have less than 31 characters due to limitations for Excel sheet names.

- `configStruct.filePath`
The absolute path of the associated post-processing function m-file which contains the user-defined algorithm (see next subchapter).
- `configStruct.applyToReqType`
A string (either 'statistical' or 'spectral') indicating to which type of requirement the function applies. As the input data and format is different for these two cases, a distinction is necessary.

All remaining structure fields are **optional**:

- `configStruct.extParamName`
A string representing the name of an arbitrary parameter structure which can be used to implicitly access any additional workspace variables in the function.
- `configStruct.description`
A string with arbitrary content which can be used to describe/comment the post-processing procedure in reports or the information panel of the **Breakdown Tree View**. If longer descriptions over multiple lines are intended, please use `char(10)` as newline character for explicit line breaks.
- `configStruct.applyToDomain`
A single string or a cell array of strings to restrict ensemble domain(s) to which the function shall be applied. Each string can be the name of an explicit user-defined domain (if present) or 'overall' – which is also the default setting if this field is not provided.
- `configStruct.applyToComponent`
A single string or a cell array of strings to restrict the components to which the analysis shall be applied. Any combination of 'tc', 'tr', 'tot' (for the time-constant, time-random and total error contributions) or 'all' are valid respectively. If the field is not provided, 'tot' is used by default.

Note: This setting only applies to **statistical** requirements. If provided for spectral requirements, the setting is ignored.

Post-Processing Function m-File

Any user-defined function to be used for a post-processing analysis must have the following form with 3 input arguments and one output argument:

```
y = postProcFunctionName(u, uUnit, extParam)
```

`postProcFunctionName` is a placeholder and any MATLAB compatible function name can be used. The function m-file must have the same name and correspond to the one defined in the configuration structure (see last chapter). Further sub-functions can be used in the m-file without restriction.

The function is called for every domain and every component specified in the configuration structure – but only if the input for this domain/component is not empty. Thus, there is no need to account for empty inputs in the function body.

Further, there is no explicit restriction on the function body implemented by a user (obviously excluding syntax errors).

During the scenario initialization, the function is executed and fed by dummy inputs of proper dimension for the block and requirement type. This checks the consistency of the function itself, but is also necessary to retrieve information about the output size, type and additional optional settings which can be used to refine the output data (see **Function Output** paragraph below).

Function Inputs

The three input arguments `u`, `uUnit` and `extParam` are always available and have the following content and format:

`u`:

The content and format of `u` depends on the requirement type assigned in the configuration structure.

Note: The input data is always provided in the unit associated to the block.

- For a **statistical** requirement, it is a numerical matrix of size [`numSamples` x (`dim`+1)] holding the samples for each axis and the line-of-sight component.

Note: `numSamples` can also be 1 in a discrete case or worst-case.

- For a **spectral** requirement, `u` is a structure with two fields:
 - `u.freq`
a [`numFreq` x 1] vector of frequencies in [Hz]
 - `u.psd`
a [`dim` x `dim` x `numFreq`] matrix of PSD data

Tip: Apply the square root to `u.psd` to obtain data in [`unit`/√Hz].

`uUnit:`

This input represent a `data.Unit` object for the related PEC block with the following structure:

- `uUnit.conversionFactor`
A conversion factor to convert the current unit data to its SI equivalent.
- `uUnit.name`
The full name of the unit (e.g. 'Radian')
- `uUnit.shortName`
An abbreviation of the unit string (e.g. 'rad')

Tip: The methods `uUnit.toSI` and `uUnit.fromSI` can be applied as factor to quickly convert data from and to the defined unit and its SI equivalent.

`extParam:`

This input represents the MATLAB structure containing any parameters that shall be made accessible in the function. It corresponds to the variable linked in the `extParamName` field of the configuration structure.

In addition to any user-defined parameters in the structure, the following data is always made available in the parameter structure:

- `extParam.currentComponent`
A string for the component currently evaluated ('tc','tr','tot' or 'spectral') when the function is called.
- `extParam.currentDomain`
A string for the domain currently evaluated ('overall' or any user-defined domain if present) when the function is called.

This provides the possibility to apply any component- or domain- specific operations in the algorithm.

Function Output

The generic interface supports only a single output argument `y`. However, `y` needs to be provided as cell array, such that a configurable number of outputs `numOut` can effectively be realized.

The content of each cell element is generally arbitrary (and can be accessed from the scenario object in the workspace), but if the results shall be displayed in the GUI or included in a report, it needs to be defined as structure with one of the following fieldnames (`idx` is a placeholder for any cell element or output respectively):

- `y{idx}.tableData`
The output represents numerical tabular data with configurable headers and description.
- `y{idx}.plotData`
The output represent plot data with configurable description and layout options.

- `y{idx}.pdfCdfData`
The output represents sample data to be evaluated as nominal error samples at a PEC block, i.e. applying an automatic evaluation and display of
 - the PDF/CDF plot of the provided data
 - a tabular overview of the budget values for a given level of confidence and a comparison to requirement values (if specified)

Tip: An arbitrary number of different output types can be used for one post-processing analysis, but only one per cell element/output, e.g.

```
y{1}.plotData.[paramFields] = ...  
y{2}.plotData.[paramFields] = ...  
y{3}.pdfData.[paramFields] = ...  
y{4}.tableData.[paramFields] = ...  
y{5}. ...
```

The sequence of the outputs also defines the sequence in which the results are displayed in the GUI.

Note: Output data for any type above is always used as provided (for display in the GUI and in reports), i.e. no further unit conversion is applied by the tool.

Note: If no unit string is provided for any output type, the evaluation unit assigned to the block (`uUnit`) gets displayed.

Tip: To indicate a dimensionless output, use '-' as unit string.

The mandatory and optional fields for each output option are explained below, but also directly described in the template file `postProcFunctionTemplate.m`.

Definition of `tableData` output structure

- `tableData.values`
This is the **only mandatory field** for this output type. It must be provided as a numeric matrix of size `[nRow x nCol]` which represents all results per column item.
- `tableData.title`
A string representing the title of the element which is used as panel name in the [Breakdown Tree View](#) or header in a report.
If not provided, 'Table data (Output [idx])' is used.
- `tableData.description`
A string with arbitrary content which can be used to describe/comment the table data in reports or the information panel of the [Breakdown Tree View](#). If longer descriptions over multiple lines are intended, please use `char(10)` as newline character for explicit line breaks.
If not provided, the description text is empty.
- `tableData.colHeaders`
A cell array of strings of size `[1 x nCol]` representing the identifiers for each data column.
If not provided a successive labeling is applied (`Value_1, Value_2, etc.`).

- `tableData.units`
A string (or a cell array of strings of length `nCol` defining the unit(s) associated to all (each of the `nCol`) columns in the table.
If no unit is provided, the unit string from the block is used.
- `tableData.rowHeaders`
A cell array of strings of size `[nRow x 1]` representing the identifiers for each data row.
If not provided, row headers are entirely ignored.
- `tableData.reqValues.[fields]`
The requirement values associated to a column in the data set. The field names are dependent on the requirement type.
For the **statistical** case – if provided – it must be a structure either with a combination of the fields `tc`, `tr`, `tot` or with a single field `all` to represent the components to which the requirement values apply. Each field itself is then:
 - a cell of size `[1 x nCol]` with each cell element defining a scalar numeric requirement value assigned to the `nCol`-th data set (or an empty element if no requirement is assigned)
 - a single `[1x1]` cell in case a single requirement value shall be assigned to all `nCol` data setsFor the **spectral** case – if provided – it must be a structure with a single field `spectral` with the same format as in the statistical case.
If not provided, all requirement values are empty.
Note: An empty cell element implies no requirement while a zero-entry implies that no error is acceptable.
- `tableData.reqIds.[fields]`
The requirement IDs associated to above requirement values. The fieldnames are again dependent on the requirement type.
For the **statistical** case – if provided – it must be a structure either with a combination of the fields `tc`, `tr`, `tot` or with a single field `all` to represent the components to which the IDs apply. Each field itself is then:
 - a cell of size `[1 x nCol]` with each cell element defining the string for the requirement ID assigned to the `nCol`-th data set (or an empty element if no ID is assigned)
 - a scalar `[1x1]` cell in case a single requirement ID shall be assigned to all `nCol` data setsFor the **spectral** case – if provided – it must be a structure with a single field `spectral` with the same format as in the statistical case.
If not provided, no requirement IDs are used.
- `tableData.displayFlags`
A `[1 x 2]` logical vector to determine if the output data should be included in reports (1st flag value) and/or displayed in the GUI (2nd flag value).
If not provided both flags are set to true.

Definition of `plotData` output structure

- `plotData.x`
This is a **mandatory field** for this output type. It must be provided as a numeric matrix of size $[m \times n]$ or $[m \times 1]$ which contains the abscissa values of the n plot data sets. In case a $[m \times 1]$ matrix is provided, the abscissa values are commonly used for each of the n data sets.
- `plotData.y`
This is a **mandatory field** for this output type. It must be provided as a numeric matrix of size $[m \times n]$ which represents the ordinate values of the n plot data sets.

Note:

If the length m of the datasets to be plotted exceeds 10^4 , only every i -th point is displayed (with i such that the maximum length is not exceeded).

The maximum number of elements in a plot n is restricted to 25.

- `plotData.title`
A string representing the title of the element which is used as panel name in the [Breakdown Tree View](#) or header in a report.
If not provided, 'Plot data (Output [idx])' is used.
- `plotData.description`
A string with arbitrary content which can be used to describe/comment the plot data in reports or the information panel of the [Breakdown Tree View](#). If longer descriptions over multiple lines are intended, please use `char(10)` as newline character for explicit line breaks.
If not provided, the description text is empty.
- `plotData.reqData.[fields]`
The requirement data values associated to the n plot data sets. The field names are dependent on the requirement type.

For the **statistical** case – if provided – it must be a structure either with a combination of the fields `tc`, `tr`, `tot` or with a single field `all` to represent the components to which the IDs apply. Each field itself is then:

- a cell of size $[1 \times n]$ with each cell element representing the requirement plot data as $[p \times 2]$ matrix of abscissa (and ordinate values for the n -th plot data set (or an empty element if no requirement is assigned)
- a $[1 \times 1]$ containing a $[p \times 2]$ matrix in case a single requirement ID shall be assigned to all n data sets

For the **statistical** case – if provided – it must be a structure with a single field `spectral` with the same format as in the statistical case.

If not provided, all requirement values are empty.

Note: An empty cell element implies no requirement while a zero-entry implies that no error is acceptable.

Note: For compliance checks, budget and requirement plot data is interpolated on a merged grid of budget and requirement abscissa data in the range provided by the requirement abscissas.

Interpolation is either performed on a linear or log-scale dependent on the chosen `axisStyle` for the plot data.

- `plotData.reqIds.[fields]`
The requirement IDs associated to above requirement values for the n data sets. The field names are again dependent on the requirement type.

For the **statistical** case – if provided – it must be a structure either with a combination of the fields `tc`, `tr`, `tot` or with a single field `all` to represent the components to which the IDs apply. Each field itself is then:
 - a cell of size [1 x n] with each cell element defining the string for the requirement ID assigned to the n-th plot data set (or an empty element if no ID is assigned)
 - a single [1x1] cell in case a single requirement ID shall be assigned to all n plot data sets
For the **spectral** case – if provided – it must be a structure with a single field `spectral` with the same format as in the statistical case.
If not provided, no requirement IDs are used.
- `plotData.legend`
A cell array of strings of size [1 x n] representing the identifiers for each plot data set.
If not provided a successive labeling is applied (`Data_1`, `Data_2`, etc.).
- `plotData.xUnit`
This is a ("short"-) string representing the unit of the abscissa data (e.g. 's' for seconds).
If not provided, the unit string from the related PEC block is used.
- `plotData.xLabel`
This is a single string to define the abscissa data label.

If `xUnit` is provided, it is automatically appended to the label in plot. If no `xLabel` is provided, the `xUnit` is used as label.
- `plotData.yUnit`
This is a ("short"-) string representing the unit of the ordinate data (e.g. '°' for degrees).
If not provided, the unit string from the related PEC block is used.
- `plotData.yLabel`
This is a single string to define the ordinate data label.

If `yUnit` is provided, it is automatically appended to the label in plot. If no `yLabel` is provided, the `yUnit` is used as label.
- `plotData.axisStyle`
A single string ('lin', 'semilogx' or 'loglog') defining the axis style for the plot.
If not provided 'lin' is used.

- `plotData.lineStyle`
This is either a common string for the line style to be used for all plot data sets or a cell array of strings of size [1 x n] with individual line-styles for each of the n plot data sets.
Possible options are '-', ':', '-.', '--' or 'none' with the same meaning as for MATLAB plots (i.e. solid, dotted, dash-dotted, dashed or no line).
If not provided, solid lines ('-') are used.
- `plotData.markers`
This is either a common string for the plot point markers to be applied to all plot data sets or a cell array of strings of size [1 x n] with individual markers for each of the n plot data sets.
Possible options are '+', 'x', 's', 'd' or 'none' with the same meaning as for MATLAB plots (i.e. plus, cross, square, diamond or no markers).
If not provided, no markers ('none') are used.
- `plotData.displayFlags`
A [1 x 2] logical vector to determine if the output data should be included in reports (1st flag value) and/or displayed in the GUI (2nd flag value).
If not provided both flags are set to true.

Definition of `pdfData` output structure

- `pdfData.samples`
This is a **mandatory field** for this output type. It must be provided as a numeric matrix of size [m x n] or [m x 1] which contains the data samples for which the PDF and CDF data shall be computed (always for both the sample vectors and their absolute values).
Note: The maximum number of sample vectors n is restricted to 25.
- `pdfData.locValues.[fields]`
This defines the levels of confidence to be used when evaluating the CDF data to obtain the budget values.
Providing these values is **optional for statistical requirements** (if not provided, the values as specified for the requirement set are used) and **mandatory for spectral requirements** (as no level of confidence values are nominally necessary for such requirement type).
For **statistical** requirements (in case different levels of confidence than defined in a requirement set shall be applied or different values shall be applied to individual sample data sets) a structure with either one or more [fields] representing the individual domain names (e.g. simply 'overall' if no user-defined domain is present) or 'all' in case the LoC values shall be applied commonly to all domains selected for the evaluation (`applyToComponent` field in the configuration structure).
Each field itself is then:

- a cell of size [1 x n] with scalar numeric values (between 0 and 100 percent) representing LoC values to be applied n-th sample set (or empty cell elements for those components where the requirement set value shall be used)
- a cell of size [1x1] with a scalar numeric value (between 0 and 100 percent) representing a common LoC value to be applied to all n sample sets of a domain

For **spectral** requirements, the format is identical but all domains must be covered and no empty cell elements are allowed.

- `pdfData.title`
A string representing the title of the element which is used as panel name in the [Budget Tree View](#) or header in a report.
If not provided, 'Sample data (Output [idx])' is used.
- `pdfData.description`
A string with arbitrary content which can be used to describe/comment the output data in reports or the information panel of the [Breakdown Tree View](#). If longer descriptions over multiple lines are intended, please use `char(10)` as newline character for explicit line breaks.
If not provided, the description text is empty.
- `pdfData.unit`
This is a ("short"-) string representing the unit of the sample vector data (e.g. 'rad' for radians).
If not provided, the unit string from the related PEC block is used.
- `pdfData.legend`
A cell array of strings of size [1 x n] representing the identifiers for each of the n sample sets (used as legend in plots and headers in tables).
If not provided a successive labeling is applied (`e_1`, `e_2`, etc.).
- `pdfData.axisStyle`
A single string ('lin', 'semilogx' or 'loglog') defining the axis style for the PDF/CDF plots.
If not provided 'lin' is used.
- `pdfData.reqValues.[fields]`
The requirement values associated to the n sample data sets. The field names are dependent on the requirement type.
For the **statistical** case – if provided – it must be a structure either with a combination of the fields `tc`, `tr`, `tot` or with a single field `all` to represent the components to which the requirement values apply. Each field itself is then:
 - a cell of size [1 x n] with each cell element defining a scalar numeric requirement value assigned to the n-th data set (or an empty element if no requirement is assigned)
 - a scalar [1x1] cell in case a single requirement value shall be assigned to all n data sets

For the **spectral** case – if provided – it must be a structure with a single field `spectral` with the same format as in the statistical case.

If not provided, all requirement values are empty.

Note: An empty cell element implies no requirement while a zero-entry implies that no error is acceptable.

- `pdfData.reqIds.[fields]`
The requirement IDs associated to above requirement values. The fieldnames are again dependent on the requirement type.

For the **statistical** case – if provided – it must be a structure either with a combination of the fields `tc`, `tr`, `tot` or with a single field `all` to represent the components to which the IDs apply. Each field itself is then:

- a cell of size [1 x n] with each cell element defining the string for the requirement ID assigned to the n-th data set (or an empty element if no ID is assigned)
- a single [1x1] cell in case a single requirement ID shall be assigned to all n data sets

For the **spectral** case – if provided – it must be a structure with a single field `spectral` with the same format as in the statistical case.

If not provided, no requirement IDs are used.

- `pdfData.displayFlags`
A [1 x 2] logical vector to determine if the output data should be included in reports (1st flag value) and/or displayed in the GUI (2nd flag value).

If not provided both flags are set to true.

- `pdfData.reportTypes`
This is a cell array of strings which determines what kind of plots shall be generated and saved during the report generation (if the corresponding flag above is enabled).

Possible options are 'all' or any combination of 'pdf', 'cdf', 'pdfabs' and 'cdfabs' for the PDF/CDF of samples or the PDF/CDF of the absolute values of the samples.

If no `reportTypes` field is provided, {'pdfabs','cdfabs'} is used by default. If the `reportTypes` field is provided but empty, no plots are saved during the report generation.

6.4.4.2 Weighted Evaluation

This analysis feature allows a weighted evaluation of the results from multiple requirement sets, i.e. a combination of the form:

$$\text{Error}_{\text{Weighted}} = \frac{\sum_i \text{Error}_{\text{Req},i} \cdot \text{Weight}_i}{\sum_i \text{Weight}_i}$$

As an example of a potential application of this feature, assume that the requirement sets present in a scenario correspond to different operational modes of a system and the fraction of time spent in each mode differs over the considered period. In this case, the overall

performance over the entire lifetime could be represented by applying (in above 'equation') a weight to the result of each requirement set according to the fraction of time spent in each mode.

The menu dialog showing the configuration option for the analysis is shown in the figure below.

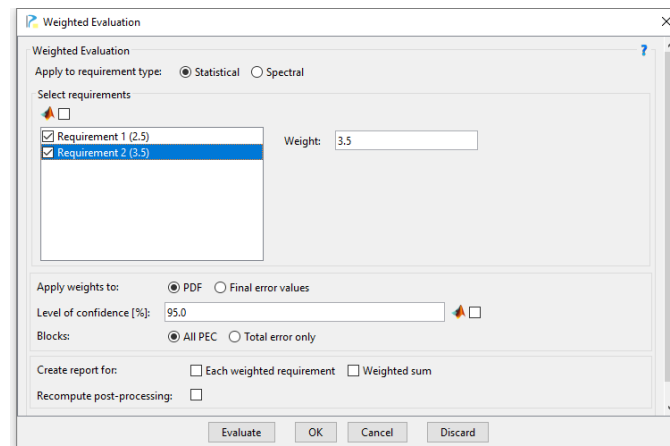



Figure 6-11: Weighted Evaluation menu dialog

First, the requirement type - i.e. **Statistical** or **Spectral** – for the analysis needs to be selected. This is necessary as the two types cover entirely different results which cannot be combined.

Dependent on this decision, the **Selected requirements** panel shows a list of all compatible requirement sets. The checkboxes in the list are used to enable or disable a certain requirement set for the evaluation. The corresponding (non-negative) **Weight** for each requirement set can be specified in the respective input field. When importing the **Requirements** from MATLAB () , cell array of size [numReq x 2] is expected, where the first dimension contains the requirement names and the second dimension the weights assigned to each requirement.

In the **Statistical** case, it is possible to further define how the weights w_i shall be applied to obtain the error $e_{weighted}$:

- To the scalar **Final error values** of each requirement set $e_{LoC,i}$, i.e. **after** each requirement set is nominally evaluated (with the levels of confidence LoC_i as specified in the **Setup** → **Scenario Definition** menu, i.e.:

$$e_{weighted} = \frac{\sum e_{LoC,i} \cdot w_i}{\sum w_i}, \quad LoC_i / 100 = \int_0^{e_{LoC,i}} p(|e_i|) de_i$$

- Alternatively, the weights can also be applied to the **PDF**, i.e. on the individual errors e_i of each requirement set **before** summing the results and evaluating the combined error with a given common **Level of confidence [%]** $LoC_{weighted}$ i.e.

$$LoC_{weighted} / 100 = \int_0^{e_{weighted}} p \left(\frac{\sum |e_i \cdot w_i|}{\sum w_i} \right) de$$

Tip: For reasonable results, all involved requirement sets should have the same statistical interpretation applied to avoid mixing ensemble and temporal statistics.

In the **Spectral** case, the weights w_i are applied to the power spectral densities $G_i(f)$ resulting from each requirement set as:

$$G(f)_{weighted} = \frac{\sum G_i(f) \cdot w_i^2}{\sum w_i^2}$$

All other settings are available for both **Statistical** and **Spectral** requirements. The weighted evaluation can either be applied to **All PEC** blocks or to the **Total Error** only.

The result of this analysis is not displayed in the GUI, but only exported to a specific spreadsheet report. The report file is stored in the scenario \output folder of the scenario with the naming convention [scenarioName]_WeightedEvaluation.

The information provided with the report depends on the status of the **Each weighted requirement** and **Weighted sum** checkboxes. With the first option, the report includes the budget results for each single requirement with the weights applied, the second includes the results for the overall weighted sum.

Finally, the status of the **Recompute post-processing** checkbox determines if any compatible post-processing analyses defined in the **Analysis Features** → **Post-Processing** menu shall be recomputed using the weighted results and included in the report (not available in case the weighting is applied to the **Final error values** of **Statistical** requirements, as the post-processing algorithms are applied before the final error values are obtained by integrating the PDF for a given level of confidence).

The **Evaluate** button triggers the weighted evaluation while **OK** only confirms and saves all changes. **Discard** can be used to reset all previously saved or configured settings.

6.4.4.3 PES-/PEC-to-PEC Percentages

The menu item **PES-to-PEC Percentages** can be used to analyse the impact of each error source block alone on all error contribution blocks (**PEC** and **Total Error** blocks) present in a scenario. It returns the ratio of a block's contribution and the overall contribution in percent individually for the time-constant, time-random and total error components for a statistical requirement set. **PEC-to-PEC Percentages** performs a similar analysis between all error contribution blocks respectively.

Both analyses can be enabled or disabled by a left-click on the respective entry in the **Analysis Features** menu. When enabled, they are automatically executed when the budget for a requirement set is (▶/▶) buttons in the **System Editor** window) and the current status is displayed in the MATLAB command window.


After evaluation, the results of the analysis are presented in tabular form in the **Budget Tree View** on a specific **PEC Percentages** tab for each error source/contribution block.

These tabular overviews can also be included in reports by checking the **PEC Percentages** option in the **File** → **Create Report...** menu (chapter 6.4.1.1) before generating a report file. In this case, additional sheets ("PES_to_PEC_Percentages" and/or "PEC_to_PEC_Percentages") are generated which summarize the data in one large table.


6.4.5 Windows Menu

All items present here are the menu-equivalents of toolbar buttons in the [System Editor](#).


Reset View

This item resets the zoom level in the editor panel to default (equivalent to the -icon).

Show Budget Tree

This item opens the [Budget Tree View](#) of the current scenario or brings it to the foreground if already opened (equivalent to the -icon).

Show Breakdown Tree

This item opens the [Breakdown Tree View](#) of the current scenario or brings it to the foreground if already opened (equivalent to the -icon).

6.4.6 Info Menu

The [Info](#) menu is used to display the current software version and contains the access to the [License Manager](#) (see also chapter 4.3).

6.5 Database Browser

The [Database Browser](#) provides access to the block database. It organizes all available block types in categories. For each block category, a tab is shown in the browser window.

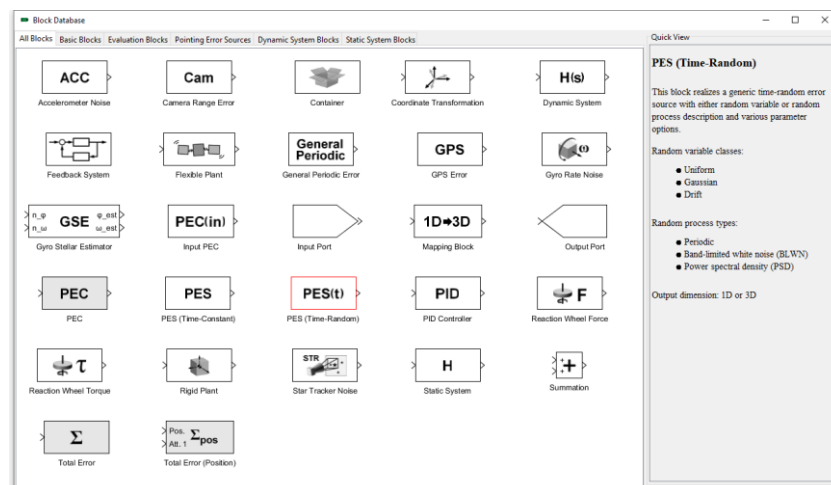


Figure 6-12: Database Browser

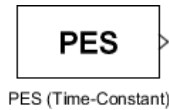
The first tab contains all block types available in PEET. All other tabs contain a subset of the block types. To add blocks from the database to the pointing system, the desired block can be dragged from the [Database Browser](#) to the [System Editor](#).

The [Quick View](#) panel on the right gives some brief summary information for each block selected in the database (e.g. purpose and I/O dimensions).

6.5.1 Pointing Error Source Blocks

This block category comprises models related to the characterization of error sources following AST-1 in [AD2]. It is possible to choose from both time-constant and time-random

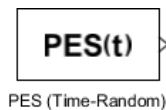
error sources. Independent from the temporal behaviour, most models also support an additional ensemble distribution of model parameters. Furthermore, dedicated error models for selected sensors and actuators are available as well as generic models which can be used to describe an arbitrary error source.



PES (Time-Constant)

This block realizes a generic time-constant error source with either 1D or 3D output. It supports different options for the statistical distribution of the error:

- Delta (No Distribution)
- Uniform
- Gaussian
- Bimodal (Arcsine)
- Rayleigh
- Truncated Gaussian



PES (Time-Random)

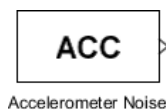
This block realizes a generic time-random error source (1D or 3D) with either random variable or random process description and various parameter options.

Supported random variable *Signal classes*:

- Uniform
- Gaussian
- Drift

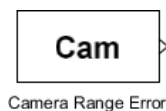
Supported random process *Types*:

- Periodic
- Band-limited white noise (BLWN)
- Power spectral density (PSD)



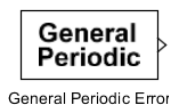
Accelerometer Noise

This block realizes a spectral model for the acceleration noise of a linear acceleration sensor which is based on a standardised model ([RD1], [RD2]). When set up as a 3D model, identical spectra are assumed for all axes.



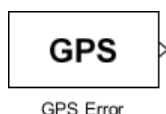
Camera Range Noise

This block realizes a simplified model for the distance measurement error of a camera based on ESA inputs. It covers both measurement bias and a range-dependent noise contribution.



General Periodic Error

This block realizes a general periodic (i.e. non-sinusoidal) error source with different options for the 'shape' of the error signal.



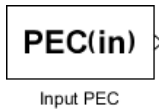
GPS Noise

This block realizes a simplified model for the position and velocity measurement error of a GPS sensor based on ESA inputs. It covers bias and spectral contributions (white noise and random walk).



Gyro Rate Noise

This block realizes a spectral model for the rate noise of a gyroscopic sensor which is based on a standardised model ([RD1]). When set up as a 3D model, identical spectra are assumed for all axes.

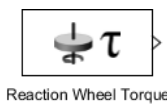


Input PEC

This block allows the definition of a:

- Time-Constant PDF
- Time-Random PDF
- Power spectral density (PSD)

separately for each requirement set as numerical data. For this data, it is assumed that error index contribution analysis (PDF, PSD) and statistical interpretation (PDF only) was already applied.

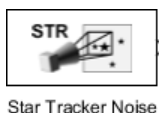


Reaction Wheel

The *Reaction Wheel (Force)* block realizes a model (based on ESA inputs) for the 3D disturbance forces induced by a single reaction wheel. It includes periodic and random disturbances for axial and radial modes.



The *Reaction Wheel (Torque)* similarly realizes a model (based on ESA inputs) for the 3D disturbance torques induced by a single reaction wheel. It includes periodic and random disturbances for the rotational mode.



Star Tracker Noise

This block realizes a 3D parametric spectral model for the field-of-view and pixel noise of a star-tracker based on ESA inputs.

6.5.2 Basic Blocks

This block category contains auxiliary blocks for the error signal routing from pointing error sources to the total error.

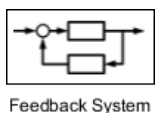


Container

This block can be used to group several blocks in a "subsystem" similarly as in Simulink.

A double-click on a container in the [System Editor](#) opens a lower level editor where the internal blocks can be placed. In the [Budget Tree View](#), the content of container can be displayed (or hidden) when right-clicking on the container (or its content).

Note: No loops are allowed inside a *Container* block!



Feedback System

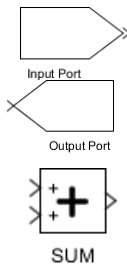
This block converts an arbitrary feedback loop model built up from various database blocks to a single dynamic system (i.e. no intermediate lower level results are provided).

Incompatible building blocks:

- All error source blocks
- All evaluation blocks (*PEC*, *Total Error*)
- *Container* block
- *Mapping* block

The loop model is created in a dedicated editor window after a double-click on the block mask.

Tip: For a proper display of a feedback system structure, internal blocks can also be flipped using the right-click menu.



Input Port & Output Port

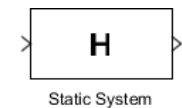
These blocks serve as connections for blocks defined in for *Container* or *Feedback System* blocks to higher levels in the **System Editor**.

Summation

This block can be used to both sum and subtract a variable number of error signals.

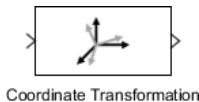
6.5.3 Static System Blocks

This block category contains generic and dedicated blocks for the static manipulation of error signals in the system transfer analysis (AST-2 in [AD2]).



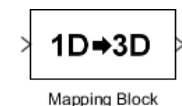
Static System

This block realizes a constant 3x3 matrix (a scalar “gain” in the 1D case) which is multiplied with the input signal.



Coordinate Transformation

This block generates a 3x3 rotation matrix from a set of Euler angles and a given rotation sequence.

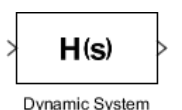


Mapping Block

This block converts a 1D signal into a 3D signal based on a user-defined configuration matrix (e.g. an actuation matrix describing the position and orientation of thrusters).

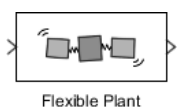
6.5.4 Dynamic System Blocks

This block category contains generic and dedicated blocks for linear time-invariant (LTI) dynamic system transfer of error signals in the system transfer analysis (AST-2 in [AD2]).



Dynamic System

This block realizes a generic dynamic system which is represented either as a state-space, transfer function or zero-pole-gain model. Both 1D and 3D inputs/outputs are supported.



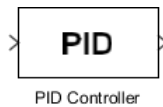
Flexible Plant

This block realizes a 3D model which describes the torque-to-attitude dynamics of a body taking into account a variable number of flexible modes.



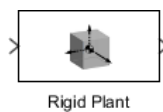
Gyro-Stellar Estimator

This block realizes an attitude estimator with user-defined gains which is fed by attitude and rate noise signals. Outputs are the attitude estimation error and the rate bias estimation error signals. The dimensions of the input and output signals need to be 3D.



PID Controller

This block realizes an ideal Proportional-Integral-Derivative (PID) controller with user-defined gains. It can be used, for instance, in loops created inside a *Feedback System* block. Supported dimensions are 1D and 3D.

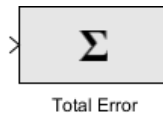


Rigid Plant

This block realizes a 3D model which describes the torque-to-attitude dynamics of a rigid body defined by its inertia matrix.

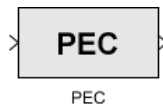
6.5.5 Evaluation Blocks

This block category contains blocks for the evaluation of the error signal for the given level of confidence (AST-4 in [AD2]).



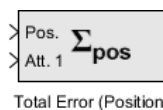
Total Error

This block defines the "endpoint" of each budget, i.e. the total error level. The complete (1D or 3D) error evaluation with respect to the given requirement(s) is executed for its input signal. Only one single *Total Error* block is allowed in a scenario.



PEC Block

The *PEC* blocks serve as equivalent substitutes for the *Total Error* blocks at lower levels of the pointing system. It is also possible to allocate sub-requirements for each block. Both 1D and 3D input signals are supported. Concerning the system transfer, a *PEC* block basically only acts as a scope, i.e. it has no effect on the input signal itself and the output signal is equivalent to the input.



Total Error (Position)

This block is a specialized realization of a 3D *Total Error* block which applies e.g. to formation flying missions where both attitude and relative position budgets are of interest. The latter is computed using a set of coupling vectors as parameters that map a (variable) number of attitude errors to effective position error contributions.

6.6 Budget Tree View

The *Budget Tree View* serves to analyse error contributions and the content of error signals of the entire pointing system. Figure 6-13 shows an example of the *Budget Tree View*. The window consists of a tool bar, an information panel on the right and a tree-like representation of the pointing system.

Different to the *System Editor*, blocks and connections cannot be moved inside this window as the blocks and connections are automatically organized in a tree-like structure starting

from the error sources on the uppermost level and ending with the *Total Error* block on the lowest level.

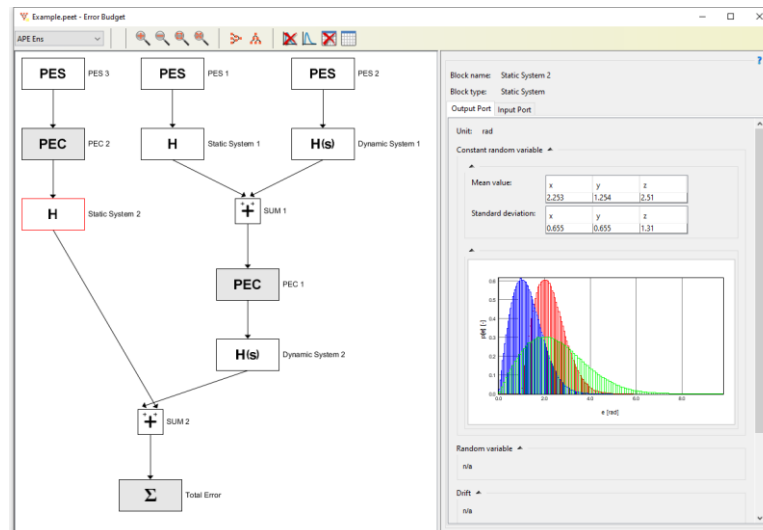


Figure 6-13: The Budget Tree View window

By default, the content of a *Container* block is not shown. To display all its internal blocks, right-click on the *Container* and select *Expand Container*. To collapse the content again, similarly right-click on any contained block and *Collapse Parent Container*.

To open or collapse all containers at once, the right-click options *Expand All* and *Collapse All* are available.

In case multiple requirement sets are defined in a scenario, first the requirement set to be analysed has to be selected from the left drop-down list in the toolbar. Similarly, in case of multiple ensemble domains, first the domain of interest needs to be selected from the drop-down list on the right of the toolbar.

Tip: In case a scenario was already evaluated for a requirement set before any further changes have been applied to block parameters or menu settings, an asterisk (*) in the requirement set drop-down list indicates that the displayed results might be outdated.

Information Panel

In case multiple requirement sets are defined in a scenario, first the requirement set to be analysed has to be selected from the left drop-down list in the toolbar. Similarly, in case of multiple ensemble domains, first the domain of interest needs to be selected from the drop-down list on the right of the toolbar.

Note: The results shown for the input/output port signals in the *Budget Tree View* represent the contributions after pointing error index analysis and statistical interpretation.

Error information is provided after a block has been selected with a left-click. Then, apart from the *Block name* and *Block Type* the information panel shows individually for each *Input Port* and *Output Port* on a separate tab the signal's *Unit*, *Mean value* and *Standard deviation* of the:

- Time-constant random variable error contribution together with a PDF plot

- Time-random variable error contribution (RV) together with a PDF plot
- Random process error contribution (RP) together with a PSD plot
- Periodic error contribution (P) together with a PDF plot
- Drift error contribution (D) together with a PDF plot

In case a signal does not contain all of the above components, N/A is displayed for the respective part in the panel. In case these auxiliary results are suppressed for a component (via 'fast-mode' in the [Setup](#) → [Evaluation Settings](#) menu), [Disabled](#) (fast-mode settings) is displayed. Further, all [Input Port](#) tabs also show the [Source](#) block of the input signal.

For **evaluation blocks** ([PEC](#) and [Total Error](#)), different tabs are present:

- **Signal components**

This tab is always present and the information provided is identical to the one described above.

- **Overall**

This tab is always present. Its content depends on the requirement type of the selected requirement set (in [Setup](#) → [Scenario Definition](#)). For a [Spectral](#) requirement, simply the PSD of the overall random process distribution is displayed. For [Statistical](#) requirements, the following information is provided:

- The level of confidence ([LoC](#)) specified for the requirement set
- The [Error Contributions](#) of [Time-constant](#), [Time-random](#) and the [Total](#) error for each axis and the line-of-sight error ([LoS](#))
- The PDF of above-mentioned contributions

- **Domain contribution** tabs

These tabs are only available, if more than two ensemble domains are defined in the scenario. In this case, similar information is present as on the [Overall](#) tab, but only for the error contributions of a certain domain.

- **PEC Percentages**

This tab is only available, if [Analysis Features](#) → [PES-To-PEC Percentages](#) is enabled. In this case it shows a tabular overview of the contribution of the selected block to all other evaluation blocks ([PEC](#) and [Total Error](#)) present in the scenario.

For all **error source blocks**, an additional tab [Source Info](#) is present which summarizes selected basic setup options for the block. In case an error source contains a random process contribution, also the "raw" initial PSD is plotted (i.e. before computing the error index contribution). In case of a periodic contribution, a map of the initially defined frequency-amplitude range is displayed (i.e. again before applying statistical treatment and error index contribution). For [Reaction Wheel](#) model outputs, the plot shows the frequency-amplitude range for each harmonic specified.

If [Analysis Features](#) → [PES-To-PEC Percentages](#) is enabled, an additional tab [PEC Percentages](#) is present which shows a tabular overview of the contribution of the selected errors source block to all evaluation blocks ([PEC](#) and [Total Error](#)) present in the scenario.

Finally, for all **dynamic system** blocks, an additional tab [System Response](#) is present which show the frequency response plots of all input/output combinations of the system.







Plots

The plots in the information panel are only previews that provide a quick-look of the error signal contribution. A double-click on a preview opens the [Plot](#) window where various modifications can be applied to the plotted data (see chapter 6.8). Furthermore, additional data can be plotted.

For PDF plots, this comprises also scatter plots to indicate the correlation between different axes of a signal (and different signals). For PSD plots also the cross-spectra between different axes of a signal (and different signals) can be displayed.

Showing/hiding subpanels

For a better overview (e.g. for showing only relevant results in screenshots) in the information panel, the following functionality is accessible via a left mouse click:

- Show all plots in subpanels using the -icon
- Hide all plots in subpanels using the -icon
- Show all tables in subpanels using the -icon
- Hide all tables in subpanels using the -icon
- Show one specific subpanel using the -icon
- Hide one specific subpanel using the -icon

6.7 Breakdown Tree View

The [Breakdown Tree View](#) can be used to check the compliance of the budget with defined requirement values in the [Requirement Specification](#) tab of the [Setup](#) → [Scenario Definition](#) menu.

Figure 6-13 shows an example of the [Breakdown Tree View](#). As the [Budget Tree View](#) window, it consists of a tool bar, an information panel on the right and a tree-like representation of the pointing system.

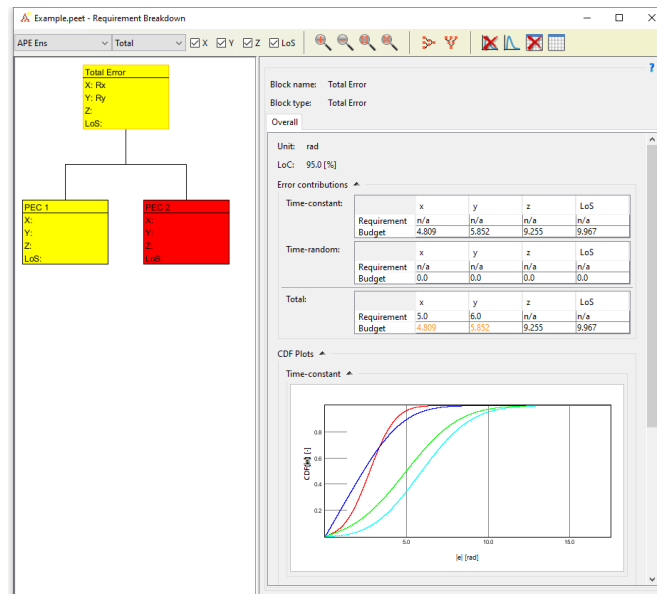


Figure 6-14: The Breakdown Tree window

Tree Panel & Toolbar

Different to the [Budget Tree View](#), the [Breakdown Tree View](#) only shows evaluation blocks (*PEC* and *Total Error*) in the tree structure. The information displayed in the information panel depends on several selections in the toolbar:

- The requirement set from the left drop-down list.

Tip: In case a scenario was already evaluated for a requirement set before any further changes have been applied to block parameters or menu settings, an asterisk (*) in the requirement set drop-down list indicates that the displayed results might be outdated.

- The error contribution (**Time-Constant**, **Time-Random** or **Total**) from the adjacent drop-down list on the right (only in case of a **Statistical** requirement)
- The component(s) to be displayed by enabling the checkboxes for the axes **x,y,z**, and the line-of-sight (**LoS**, only in case of a **Statistical** requirement)

Each block in the tree panel first shows requirement IDs ([Requirement ID](#)) in case any have been assigned to this block for the given selection. Furthermore, the block indicates the status of the selected requirements as a quick-look using the following colour-coding:

- No requirement has been specified.
- The requirement is entirely met.
- The requirement is met, but the margin less than specified in the [Setup → Evaluation Settings](#) menu.
- The requirement is violated.

In case multiple axes are the selected, the block colour always represents the “worst-case” of the selection (e.g. if the x-axis requirement is met, but the y-requirement is violated, the block colour is red, if both axes are selected)

For **Spectral** requirements, the “worst-case” point over frequency is used to determine the requirement status (i.e. the point with the largest violation or smallest margin).

Information Panel

The content of the first part of the information panel content is similar to the one for evaluation blocks in the **Budget Tree View**. For the selected requirement set (left drop-down list in the toolbar), the content of the **Overall** tab depends on whether a **Spectral** or **Statistical** requirement has been defined.

For a **Spectral** requirement, the budget PSD is plotted together with the requirement function $S_{req}(f)$ if specified (see chapter 6.4.2.2).

For a **Statistical** requirement, the tab contains the **Level of confidence** specified for the requirement set and the **Error Contributions** of **Time-constant**, **Time-random** and the **Total** error for each axis and the **Line-of-sight error**. In addition, also the related requirement values are shown where available. Below the numerical results, the CDFs of the error contributions’ absolute value are shown.

If PSD contributions are present, also the **Cumulated variance (normalized)** is displayed. This quantity gives an indication of how sensitive the evaluated random process variance is with respect to the chosen evaluation bandwidth. If the function increases significantly at the upper frequency bound, an increase of the bandwidth would lead to a larger contribution and the evaluation bandwidth might need to be increased (dependent on the application). If it is flat, the contribution does not change (likely, as higher frequency information is not available) and the bandwidth could even be reduced by the flat part.

As for the **Budget Tree View**, further **Domain contribution** tabs with equivalent information for each domain are available, if user-defined ensemble domains are present in the scenario.







Finally, additional tabs are present for all blocks for which one or more post-processing analyses are defined via the **Analysis Features** → **Post-Processing...** menu. The content of these tabs depends on the selected analysis (see chapter 6.4.4.1).

Plots

The CDF, PSD and cumulated variance plots in the information panel are only previews that provide a quick-look of the error contribution. A double-click on a preview opens the **Plot** window where various modifications can be applied to the plotted data (see chapter 6.8).

Showing/hiding subpanels


For a better overview (e.g. for showing only relevant results in screenshots) in the information panel, the following functionality is accessible via a left mouse click:

- Show all plots in subpanels using the -icon
- Hide all plots in subpanels using the -icon
- Show all tables in subpanels using the -icon
- Hide all tables in subpanels using the -icon
- Show one specific subpanel using the -icon
- Hide one specific subpanel using the -icon



6.8 Plot Window

The [Plot](#) window opens after a double-click on any preview plot in the [Budget Tree View](#) or the [Breakdown Tree View](#). By default, the window just shows an enlarged version of the preview.


Zooming

Zooming in a certain area of the plot is possible by pressing the right mouse button and dragging a frame around the desired area while keeping the button pressed. For a further zoom, this action can be repeated. A left-click on the -icon in the toolbar resets the view to its original state.

Picking a data point

The exact value of a certain point in a plot can be accessed using the data picker (-icon in the toolbar). Select it with a left-click and then right-click on the data point from which the value should be read. This displays the ordinate and abscissa value at this point. To leave the picking mode, left-click the -icon again.


Saving MATLAB style plots

The current plot can be saved based on an equivalent MATLAB figure using the -icon in the toolbar. A figure `[filename].[fileExt]` is then created in the scenario subfolder `\output\figures\[requirementSetName]\.`

`[fileExt]` – i.e. format of the image file – and the style of the plot depend on the selection in the [Plot options](#) panel of the [Setup](#) → [Preferences](#) menu (see 6.4.2.9). The `[filename]` is created automatically and printed in the [Execution Log](#).

Note: These plots are saved in the “default state”, i.e. neglecting any zoom states or deselected axes.

Saving GUI style plots

The current plot can also be directly saved as displayed in the GUI (including current axis selection, zoom state, etc) using the -icon in the toolbar. This opens a dialog where the filename, image format and location can be specified.

Selecting data to be plotted

The plot options depend on what type of plot is present. For some types, a second plot tab exist where a different related quantity can be plotted.

- PDF plots from the [Budget Tree View](#)
For the plots on the [PDF](#) tab, the axis information to be displayed can individually be configured using the *x*, *y*, and *z* (plus *LoS* for *PEC* and *Total Error* blocks) checkboxes in the toolbar.
Additionally, a second tab [Correlation](#) is present which shows a scatter plot for a selected combination together with the computed correlation coefficient. The first checkbox defines the axis of the current [Component](#) plotted on the abscissa. The drop-down lists on the right are used to select the “target data” for the correlation to be plotted on the ordinate:
 - Block name

- Output port
- Axis
- Component

To create the scatter plot for the current selection, left-click on the refresh icon (🔄).

Note: As **Correlation** plots are not created a priori, it is only possible to retrieve the data for the last evaluated requirement set.

- PSD plots from the [Budget Tree View](#)
For the plots on the **PSD** tab, the axis information to be displayed can individually be configured using the **x**, **y**, and **z** checkboxes in the toolbar.


Additionally, a second tab **CPSD** is present to show the cross-power spectra between the different axes of the signal by default.

Similar to the correlation plots described above, it is also possible to check the cross-power spectrum between the current block and any other block that contains a random process contribution. To do so, select any the “target” block and its output port using the two drop-down list on the right of the toolbar. It is possible to choose from all possible combinations of axes: **xx**, **yy**, **zz**, **xy**, **xz**, **yz**, **yx**, **zx**, and **zy**. The first letter corresponds to the axis of the current block, the second letter to the axis of the “target” block. To create the CPSD plot for the current selection (multiple axis combinations are possible), left-click on the refresh icon (🔄).

Note: As **CPSD** data between different blocks is not computed a priori, it is only possible to retrieve the data for the last evaluated requirement set.

- CDF plots from the [Breakdown Tree View](#)
For the plots on the **CDF** tab, the axis information to be displayed can individually be configured using the **x**, **y**, **z** and **LoS** checkboxes in the toolbar.
- PSD plots from the [Breakdown Tree View](#)
For the plots on the **PSD** tab, the axis information to be displayed can individually be configured using the **x**, **y**, and **z** checkboxes in the toolbar. In addition to budget spectrum, also the corresponding requirement function is plotted (if defined) with the same colour as dashed line.
- Cumulated variance plots from the [Breakdown Tree View](#)
For the plots on the **Cumulated Variance** tab, the axis information to be displayed can individually be configured using the **x**, **y**, and **z** checkboxes in the toolbar.
- Post-processing analysis plots from the [Breakdown Tree View](#)
Depending on the type of the analysis different generic plot types are available. For user-defined analyses, properties as axis style, line style, markers, etc. can also be customized (see section 6.4.4.1.5).


7 Script-Based Execution

Once a pointing scenario has been set up and saved as a `.peet` file, it can be controlled, modified and evaluated using MATLAB commands (e.g. executing a MATLAB script file). This is especially useful when parameters in the scenario are defined by MATLAB variable names (using the -icon) as related parameters can then be easily modified between computation runs by changing the MATLAB variable itself.

Note: Please do not manually modify the `PointingSystemDefinition.xml` files of the scenario for batch-mode purposes. This may permanently corrupt the scenario and make it even unreadable by the GUI.

Tip: An application example of the script-based execution is available in the `\examples` subfolder of the PEET installation (`Example6.peet`). The MATLAB script `Example6_scriptbased.m` contains a procedure to repeatedly evaluate a scenario with different parameter values.

The major purpose of the script-based execution is the modification of parameter values during different computation runs. The parameter variation can be realized by the following two means:

- Change the value of a variable in the MATLAB workspace that is linked to a parameter using MATLAB notation (-icon) in the scenario.
- Change and save values stored in an Excel-file which is linked to the scenario.

All functions described in the following sections do not manipulate values of error signal parameters themselves, but modify scenario settings which are usually defined in menus in the GUI. Further, they serve for exporting and inspecting analysis results or for configuring additional analysis features.

7.1 Common Function Parameters and Input Default Values

The following list contains input parameters which are used frequently in many functions. Their meaning – and default values for optional inputs - are described once in the list below.

Tip: Optional inputs are displayed in square brackets in the following subsections.

- `obj` is the identifier (which can have any user-defined name) for the analysis object in the MATLAB workspace
- `blockName` is a string that contains name of a block. When used for functions related to a PEC block, this input argument is often optional. If not provided (or empty) in this case, the *Total Error* block of the scenario is queried.
- `reqSetName` is the name of a requirement set. If optional and not provided (or empty) in this case, the currently selected active requirement is queried.
- `outputPort` defines the port number of the block (always 1 for single output blocks). When optional and not provided, the first input port is queried.
- `domain` is a string that identifies the domain from which the data should be retrieved in case several ensemble domains have been defined in the scenario. If optional and not provided or empty, the first/default domain is queried. For *PEC*

and *Total Error* blocks, also 'overall' is a valid option for the overall contributions from all domains (which is also the default value here in case the input is optional and not provided).

Note: When accessing data for a *Total Error (Position)* block, the domain is defined differently as different contribution levels are available:

```
'TotalPositionError.[domain] '  
'PositionContribution.[domain] '  
'AttitudeContribution.[domain] '
```

- `plotAxis` is a string that determines which component of the signal should be considered, i.e. 'x', 'y', 'z'. In case of a *PEC* or *Total Error* block, also 'los' is a valid option.

Tip: A help to each function is also available from the MATLAB command window using `help [obj].[functionName]` for all functions described in this section. `[obj]` is a placeholder for the scenario object variable. Alternatively access the help via `help engine.Analysis.[functionName]`.

Tip: Calling any of the functions in the form `obj.functionName(inputArguments)` is equivalent to using the scenario object as first input argument, i.e. `functionName(obj,inputArguments)`.

7.2 Selecting a Scenario

First, it is necessary to select an existing scenario and create a corresponding analysis object. This is realized by the following function:

```
[obj] = engine.Analysis(scenarioURI)
```

`obj` (which can have any user-defined name) is the identifier for the analysis object in the MATLAB workspace and `scenarioURI` is a string with the absolute path to the scenario folder, e.g. `C:\MyScenario.peet`.

Tip: The scenario object `obj` is named `pointing_system` by default in the MATLAB workspace when loaded from the GUI (with consecutive naming in case several scenarios are present, i.e. `pointing_system_1`, `pointing_system_2`, etc.).

7.3 Initialising a Scenario

Before any evaluation, the scenario first has to be initialised (i.e. all settings are read from the scenario XML files). In case multiple requirement sets exist, the initialisation only needs to be executed once before evaluating the first set. Whenever values or settings (e.g. stored in MATLAB variables or in linked Excel files) are modified, a re-initialisation is necessary (automatically checked by the tool).

Several functions exist for that purpose:

Tip: The `code` and `msg` outputs are always optional. `code` provides information about the execution status of the function (0 = success, 1 = warning, 2 = error) and `msg` a corresponding message in warning or error cases.

```
[code,msg] = obj.initialiseSystem
```

This function initialises the scenario. This step is **mandatory** and has to be carried out in any case before computing a budget.

```
[code,msg] = obj.initialiseCorrelation
```

This function initialises correlation and coherence of error sources. This step is **optional**; if omitted, all error sources are assumed to be uncorrelated and incoherent.

```
[code,msg] = obj.initialiseRelativePhase
```

This function initialises relative phase of periodic signals in error source blocks. This step is **optional**; if omitted, the relative phase between all periodic signals is assumed to be zero.

```
[code,msg] = obj.initialiseFrequencyGrid(refineFlag)
```

This function initialises – dependent on the logical value of the `refineFlag` - a refined frequency grid or the default log-spaced grid for random process error signal propagation. This step is **optional**; if omitted, the setting stored in the scenario XML file is used (as set in the [Setup → Refine Frequency Grid...](#) menu, see chapter 6.4.2.8.)

7.4 Inspecting Scenario Settings

Once a scenario is initialised with all settings stored in the scenario XML files, it is possible to inspect the most relevant settings. The following paragraphs provide helper functions belonging to different groups alphabetically by function name.

Block and General Settings

The following functions can be used to access blocks settings, properties and dependencies as well as basic scenario information:

```
idx = obj.getActiveErrSrcIdx([reqSetName])
```

This function returns a list of block indices (w.r.t. the scenario block structure, i.e. `obj.blocks{index}`) of all active error sources for an inquired requirement set. If no `reqSetName` is provided, the currently active requirement set is queried.

```
names = obj.getActiveErrSrcNames([reqSetName])
```

This function returns a list with the names of all active error sources for an inquired requirement set. If no `reqSetName` is provided, the currently active requirement set is queried.

```
idx = getBlockIndex(obj, 'name', blockName)  
idx = getBlockIndex(obj, 'enum', enum.BlockType.xx)
```

The first option for this function directly returns the index of a block (w.r.t. the scenario block structure, i.e. `obj.blocks{index}`) given a string for the block name. The second option

returns the block index (or indices) of all blocks of a certain block type `xx`. The block type is defined by a respective enumeration. All options for this enumeration can be displayed in the MATLAB command window by executing `enumeration.BlockType`.

```
portNum = getBlockPortNum(obj,blockId)
```

This function provides returns the number of output port requested blocks. `blockId` can be either a single block name string (or numeric index) or a cell array of block names or a vector of block indices.

```
[blockNames,accessIndex,corrMat] =  
obj.getCorrelationSettings(type,[domainName])
```

This function provides a list of the defined error source dependencies of a requested `type` (possible types are `'ensemble'`, `'temporal'` and `'coherence'`). The second argument `domainName` is only required, when ensemble correlation is requested and user-defined ensemble domains are present in the scenario.

The `blockNames` output is a cell array of strings of length `N` which contains all error source block names in the sequence of how they are stored in the correlation/coherence matrix `corrMat`. `accessIndex` is a `2xN` matrix which contains the start and end index of the entries related to each block in the overall `corrMat`.

```
[idx,domNames] = obj.getErrorSourceIndexBySignal(component)
```

If called with one output argument `idx` only, this function returns a list with block indices (w.r.t. the scenario block structure, i.e. `obj.blocks{index}`) of all error source blocks which contain the requested error signal `component` (`'crv'`, `'rv'`, `'rp'`, `'d'`, `'p'` for time-constant random variable, time-random variable, random process, drift or periodic components respectively).

When called with both output arguments, `domNames` is a cell array which contains the names of all present ensemble domains. In this case, `idx` is also a cell array of the same length and each element contains the list of block indices for the corresponding domain.

```
flags = obj.getFastModeFlags
```

This function returns the current flag values that determine the auxiliary evaluation (1 = skipped) of signal components (see chapter 6.4.2.3). The sequence of the flags corresponds to the components as follows: [CRV, RV, RP, D, P] for time-constant random variable, time-random variable, random process, drift or periodic components respectively.

```
index = obj.getMetricIndex([reqSetName])
```

This function returns the index of the metric data for the inquired requirement set in the scenario data structure, i.e. `obj.metric(index)`.

```
settings = obj.getMetricSettings([reqSetName])
```

This function returns metric settings for the inquired requirement set (e.g. error index and window time parameters).

```
[names, freqNum] = obj.getRelativePhaseBlockNames
```

This function returns the error source block names present in the relative phase matrix (see 6.4.2.6) and the number of frequencies present in the periodic signal for each block.

```
[relativePhaseData] = obj.getRelativePhaseSettings
```

This function returns the relative phase settings for all periodic error sources in a scenario. `relativePhaseData` is a cell array with itself a [1 x 2] cell array in each cell. The first embedded cell contains the name of the block. The second embedded cell is a [$N_f \times (\text{dim}+1)$] matrix `mat`. The first column of this matrix is the frequency vector of the block's output signal, the second to last columns are the relative phases (in [rad]) for each frequency component and corresponding axes (x,y,z or only x). Thus, the output `relativePhaseData` is of the form `{{blockName, mat}, {blockName, mat}, ...}`.

```
[blockNames, status] = obj.getPESActivations([reqSetName])
```

This function returns the activation status of all error source block for the requested requirement set. `blockNames` is a cell array of strings with the PES block names, `status` is a vector of the same length with corresponding status values (1 or 0).

```
[settings] = obj.getScenarioSettings(paramString, [blockName])
```

This function returns information about different scenario settings in cell array dependent on the provided `paramString` (`blockName` is only required for the last option below):

- A list of all block names defined in the scenario, their corresponding index (in `obj.blocks`) and their type:

```
'blocks' settings = {{index, name, type}, {index, name, type}, ...}
```

- A list of all ensemble domain names defined in the scenario and their corresponding index (in `obj.stat.eDomain`). The string `Ensemble_Domain` is returned if no user-defined domain is present.

```
'domain' settings = {{index, name}, {index, name}, ...}
```

- A list of all requirement set names defined in the scenario, their corresponding index (in `obj.evalSet.requirementSet`) and their type (statistical or spectral):

```
'reqSet' settings = {{index, name, type}, {index, name, type}, ...}
```

- The unit information (conversion factor and name) of a block's output signal identified by the string `blockName`. In case a block has multiple output ports, an cell array of unit structures is returned.

```
'unit' settings = {unitInfo, unitInfo, ...}
```

```
[suggestedBW,initialBW] = obj.getSuggestedBandwidth
```

This function returns the initial and the suggested frequency bandwidth after analysis of all user-defined bandwidth settings and system data. Both outputs are 2x1 vector where the first entry hold the lower and the second entry holds the upper bound frequency bandwidth exponent (i.e. n for 10ⁿ).

Plot Settings

The following functions provide access to the settings usually available from the **Plot options** panel in the **Setup → Preferences** menu (chapter 6.4.2.9).

```
type = obj.getCorrelationPlotType
```

This function returns current plot type to be used for correlation plots ('scatter' or 'heatmap' plot).

```
type = obj.getPdfPlotType
```

This function returns current plot type to be used for PDF plots ('line' or 'bar' plot).

```
formats = obj.getPlotFileFormat
```

This function returns a cell array with the current file formats applied when saving plots.

```
flag = obj.getPlotOverwriteFlag
```

This function returns a flag which determines if existing plots are overwritten (true) or if a time stamp is appended to the name of existing plots instead (false).

Report Settings

The following functions provide access to the settings usually available from the **Report configuration** panel in the **File → Create Report...** menu (chapter 6.4.1.1).

```
flag = obj.getAutomaticReportFlag
```

This function returns a flag which determines the status of the automatic report generation during the budget evaluation (in the `computeBudget` function).

```
flags = obj.getReportAxisFlags
```

This function returns flag values that determine the current display status of certain axes (x,y,z,LoS) in report tables.

```
flag = obj.getReportOverwriteFlag
```

This function returns the flag to determine if existing reports are overwritten (true) or if a time stamp is appended to the name of existing reports instead (false).

```
prefixString = obj.getReportPrefix
```

This function returns the current prefix string to be prepended to the autogenerated name of a report.


```
plotFlagStruct = obj.getReportPlotFlags
```

This function returns the current plot generation flags for reports. The flags are stored in a structure with the various plot types as fieldnames. The sequence of flag vectors corresponds to the sequence in the [Plot configuration](#) subpanel of the [File → Generate Report...](#) menu (section 6.4.1.1).

```
sheetFlagStruct = obj.getReportSheetFlags
```

This function returns the current sheet report generation flags stored in a structure with the respective sheet names as fieldnames.

```
suffixString = obj.getReportSuffix
```

This function returns current suffix string to be appended to the autogenerated name of a report.

Analysis Features Settings

The functions in this section can be used to inspect the settings of analysis features defined via the GUI (6.4.4) and/or to check modifications applied with the functions in the next chapter.

```
flags = obj.getPecPercentageFlags
```

This function returns the current flag values that determine the evaluation of PES-to-PEC and PEC-to-PEC percentage contributions in a 2x1 vector.

```
flag = obj.getPostProcEvalState
```

This function returns the current flag value that determines the automatic evaluation of post-processing analyses defined for PEC blocks when computing a budget.

```
[outputElements, applyToReqType, applyToComponent, applyToDomain, ...  
description] = obj.getPostProcDefs(postProc, [blockName])
```

This function returns the definition data (see also section 6.4.4.1) of a post-processing element defined by its name or index (`postProc` input) and a given PEC `blockName`. If no `blockName` is provided, the *Total Error* block of the scenario is queried. The function outputs are:

- `outputType`:
A vector containing an identifier for the type of each output (plot = 1, table = 2, PDF/CDF data = 3)
- `applyToReqType`:
A string representing the requirement type for which the post-processing is defined (statistical or spectral)
- `applyToComponent`:

A cell array of strings containing the components to which the post-processing is applied ('tc', 'tr', 'tot' for time-constant, time-random or total contribution)

- `applyToDomain`:

A cell array of strings containing the ensemble domains to which the post-processing is applied

- `description`:

A string representing the user-defined description of the post-processing

```
[postProcIds,postProcNames] = obj.getPostProcNames([blockName])
```

This function returns a cell array `postProcNames` with the names of all post-processing elements defined for a PEC block together with the vector `postProcIds` of corresponding numeric IDs in the post-processing data structure of the block. If no `blockName` is provided, the *Total Error* block of the scenario is queried.

7.5 Modifying Scenario Settings

Note: All changes applied to the scenario object by the functions in this chapter are **only temporary**, i.e. they are overwritten by the settings stored in the scenario XML files whenever the scenario is reinitialised (`obj.initialiseSystem`). Consider this in the calling sequence of your MATLAB script or apply the changes via the PEET GUI to store them permanently in the scenario XML files.

Block and General Settings

The following functions can be used to modify settings usually available from the [Setup → Evaluation Settings](#) (chapter 6.4.2.3) and [Setup → Scenario Definition](#) (chapter 6.4.2.2) menus.

```
obj.setFastModeFlags(flags)
```

This function sets the flag values that determine the auxiliary evaluation (1 = skipped) of signal components. The sequence of the flags corresponds to the components as follows: [CRV, RV, RP, D, P] for time-constant random variable, time-random variable, random process, drift or periodic components respectively.

```
obj.setSourceActivation(reqSetName,blockName,flag)
```

```
obj.setSourceActivation(reqSetName,blockData)
```

This function activates or disables error sources for the evaluation of the specified requirement set (`reqSetName`).

When called with 3 input arguments, `blockName` is either a string with the source block name to be modified or a cell array of strings if a group of blocks shall be modified. `flag` is a logical or numeric vector of the same length that holds the activation status for each block (1/true = active, 0/false = disabled).

Alternatively, the function can be called with two arguments. In this case, `blockData` is a `[numBlocks x 2]` cell array where the first dimension contains the blocks names and the second dimension the corresponding status flags for all blocks.

Plot Settings

The following functions allow a modification of settings usually available from the **Plot options** panel in the **Setup → Preferences** menu (chapter 6.4.2.9).

`obj.setCorrelationPlotType(type)`

This function sets the plot `type` to be used for correlation plots (`'scatter'` or `'heatmap'`).

`obj.setPdfPlotType(type)`

This function sets the plot `type` to be used for PDF plots (`'line'` or `'bar'`).

`obj.setPlotFileFormat(formats)`

This function sets the file formats to be applied to saved plots. `formats` is a cell array of strings with any combination of the possible options `'fig'`, `'png'`, `'jpeg'`, `'bmp'`, `'pdf'`.

`obj.setPlotOverwriteFlag(flag)`

This function sets the `flag` to determine if existing plots are overwritten (true) or if a time stamp is appended to the name of existing plots instead (false).

Report Settings

The following functions allow a modification of settings usually available from the **Report configuration** panel in the **File → Create Report...** menu (chapter 6.4.1.1).

`obj.setAutomaticReportFlag(flag)`

This function sets the `flag` which determines the automatic report generation during the budget evaluation (in the `computeBudget` function).

`obj.setReportAxisFlags(flags)`

This function sets flag values (4x1 logical vector) that determine the display of certain axes (x,y,z,LoS) in report tables.

`obj.setReportOverwriteFlag(flag)`

This function sets the flag to determine if existing reports are overwritten (true) or if a time stamp is appended to the name of existing reports instead (false).

`obj.setReportPrefix(prefixString)`

This function sets the prefix string to be prepended to the autogenerated name of a report.

```
updatedFlagStruct = obj.setReportPlotFlags(flagStruct)
```

This function sets the flags for the automated plot generation in reports via a structure `flagStruct`. To obtain a valid structure, simply use `obj.getPlotSheetFlags` to get the full structure, then modify the values as needed (the sequence of the flag vectors corresponds to the sequence in the **Plot generation** subpanel of the **File → Generate Report...** menu. The optional output `updatedFlagStruct` can be used to cross-check the applied changes.

```
updatedFlagStruct = obj.setReportSheetFlags(flagStruct)
```

This function sets the report sheet generation flags stored in a structure `flagStruct`. To obtain a valid structure, simply use `obj.getReportSheetFlags` to get the full structure, then modify the values as needed. The optional output `updatedFlagStruct` can be used to cross-check the applied changes.

```
obj.setReportSuffix(suffixString)
```

This function the suffix string to be appended to the autogenerated name of a report.

Analysis Features Settings

The functions in this section can be used to manipulate analysis features defined in the scenario XML file via the GUI (6.4.4). In addition, such analysis features can also be added from scratch or removed temporarily during the script-based execution (but not saved or removed permanently to/from the scenario XML files).

```
obj.addPostProc(res, configStruct, [blockName])
```

This function adds a post-processing element to a PEC block. If no `blockName` is provided, the **Total Error** block of the scenario is queried. The `configStruct` for the elements must be compatible to the one for a user-defined post-processing as described in section 6.4.4.1).

```
obj.addWeightedEvaluation(reqType, reqList, weightVector)
```

```
obj.addWeightedEvaluation(reqType, weightList)
```

This function introduces a weighted evaluation analysis (see section 6.4.4.2) to the scenario. In a first step, only the fundamental parameters need to be specified. `reqType` requirement type for which the analysis shall be applied ('statistical' or 'spectral'). `reqList` is a cell array of strings which defines the names of the requirement sets which shall be taken into account and `weightVector` defines the numerical weight to be applied to the results for each requirement set. Alternatively, this information can also be provided in a single [`numReq` x 2] cell array `weightList` where the first dimension holds the requirement names and the second the corresponding weights.

Further refinement of the analysis parameters is possible via dedicated setter functions. `help engine.Analysis.addWeightedEvaluation` from the MATLAB

command window provides a detailed description of all further options.

```
obj.removePostProc(configStruct, [blockName])  
obj.removePostProc(postProcName, [blockName])
```

This function removes a post-processing element from a PEC block. If no `blockName` is provided, the *Total Error* block of the scenario is queried. Either the `configStruct` for the element or its `postProcName` (i.e. `configStruct.evalName`) must be provided as input argument.

```
obj.setPecPercentageFlags(flags)
```

This function sets flag values (2x1 vector) that determine the automatic evaluation of PES-to-PEC and PEC-to-PEC percentage contributions when computing a budget.

```
obj.setPostProcEvalState(flag)
```

This function sets flag value that determines the automatic evaluation of post-processing analyses defined for PEC blocks when computing a budget.

7.6 Evaluating a Scenario

After initialisation, the requirement set to be evaluated needs to be selected by:

```
[code,msg] = obj.setActiveRequirementSet(reqSetName)
```

This function sets the active requirement set for the budget computation and evaluation. `reqSetName` is a string that corresponds to the name of a requirement set existing in the scenario (note that this function replaces the deprecated function `setActiveIndex` which is however still supported). This function call is **mandatory**.

Then the actual computation can be started:

```
[code,msg] = obj.computeBudget([checkLinks])
```

This function prepares all error source signals data, propagates the error signals through the entire system and applies the error evaluation at the PEC blocks. Further, reports, plots and analysis features are automatically generated when the respective flags are set in menus or manually via the setter functions in section 7.5. This function call is **mandatory** to compute any budget result.

By default, the scenario data is checked for any modified values in Excel tables and or MATLAB variables. To skip this check, the optional input `checkLinks` can be set to false.

Tip: The `code` and `msg` outputs are always optional. `code` provides information about the execution status of the function (0 = success, 1= warning, 2=error) and `msg` a corresponding message in warning or error cases.

7.7 Evaluating Specific Analysis Features

The functions in this section trigger the evaluation of analysis features defined in the scenario XML file via the GUI (6.4.4) or manually configured used the specific functions presented in section 7.5.

```
obj.evalPostProc(postProcIndices, [blockName])
```

```
obj.evalPostProc(postProcNames, [blockName])
```

This function evaluates one or more post-processing element defined for a PEC block. The elements can either be provides as a cell array containing the `postProcNames` (which can be retrieved with the `getPostProcNames` function) or their corresponding numeric `postProcIndices`. If no `blockName` is provided, the *Total Error* block of the scenario is queried.

Tip: No explicit call of this function is required when the respective flag is set via the `setPostProcEvalState` function. In this case, all post-processing elements for all PEC blocks are evaluated automatically with the `computeBudget` routine.

```
obj.evalPecPercentages([info])
```

This function computes the contributing percentage of each PES to all PEC(s) and/or of PEC to higher level PEC(s) dependent on the specified flag status. If the logical `info` flag is provided and true, execution status information is printed to the MATLAB command window.

Tip: No explicit call of this function is required when the respective flags are set via the `setPecPercentageFlags` function. In this case, the analysis is performed evaluated with the `computeBudget` routine.

```
obj.evalWeightedRequirements([info])
```

This function computes a budget based on a weighted combination of multiple requirement sets (see also chapter 6.4.4.2). If the logical `info` flag is provided and true, execution status information is printed to the MATLAB command window.

Before this analysis can be performed:

- either the function `obj.initialiseWeightedEvaluation` must be called to load settings stored the scenario XML file
- or an analysis must be manually defined via the `obj.addWeightedEvaluation` routine.

7.8 Analysing and Inspecting Scenario Results

Results of an evaluation can be either inspected using specific helper functions (recommended) or manually by “browsing” the analysis object.

7.8.1 Functions

Note: All results saved in the scenario object are represented by equivalent SI units, i.e. generally **not** in the unit initially specified in the GUI.

Tip: To convert data retrieved by the functions below back to the specified output unit:

- get the block output unit object:

```
myUnit = obj.getScenarioSettings('unit',blockName)
```

- apply the conversion factor to the extracted data:

```
myData_SI = myData * myUnit.fromSI
```

General Budget Results

The functions in this section allow access to the numeric data which is displayed in the [Budget Tree View](#) and [Breakdown Tree View](#) tables of the GUI. Further, functions are available which can be used to determine the presence of certain signal components in the output signals of all blocks.

```
[data] = obj.getBudgets(budgetType,[blockName],[domain])
```

This function returns the budget values of a certain *PEC/Total Error* block. *data* is a 4x1 vector (x, y, z and line-of-sight) in case of a 3D signal and scalar value in case of a 1D signal. If no (or an empty) *blockName* is provided, the *Total Error* block is queried; if no *domain* is provided, the 'overall' domain is used.

```
flags = obj.getPECErrorsFlags([blockName],[domain])
```

This function returns *PEC/Total Error* block error contribution flags which indicate the presence of a component. The entries in the 3x1 flags vector correspond to the presence of time-constant, time-random and total error contributions. If no (or an empty) *blockName* is provided, the *Total Error* block is queried; if no *domain* is provided, the 'overall' domain is used.

```
data = obj.getSignalStatistics(blockName,component,[outputPort],  
[domain])
```

This function returns block output signal mean values and standard deviations for a given block name, output port and signal component ('crv', 'rv', 'rp', 'd' and 'p' for time-constant random variable, time-random variable, random process, drift or periodic components respectively). If no (or an empty) *outputPort* is provided, the first port is queried; if no *domain* is provided, the first/default domain is used.

```
dim = obj.getSignalDimension(blockName,[outputPort])
```

This function returns scalar block output signal dimension. If no (or an empty) *outputPort* is provided, the first port is queried.

```
flags = obj.getSignalFlags(blockName,[outputPort],[domain])
```

This function returns a 5x1 flags vector which indicates the presence of signal components in a block output (sequence: 'crv', 'rv', 'rp', 'd' and 'p' for time-constant random variable, time-random variable, random process, drift or periodic components respectively). If no (or an empty) *outputPort* is provided, the first port is queried; if no *domain* is provided, the first/default domain is used.

Plot Data of Results

The functions in this section allow access to the data which is used to generate the various plots displayed in the [Budget Tree View](#) and [Breakdown Tree View](#) of the GUI. The common output arguments `xData` and `yData` are vectors with the abscissa values and ordinate values. If a requested component is not available, empty outputs are returned. The presence of data can first be determined using the “flag-functions” in the previous paragraph.

```
[xData,yData] = obj.getBlockCPSD(blockName1,outputPort1,  
    blockName2,outputPort2,plotAxes,[domain],[reqSetName])
```

This function returns the CPSD plot data between different blocks for given block names, output ports and axis components ('xy', 'xz', 'yx', etc.). If no (or an empty) `domain` is provided, the first/default domain is queried; if no `reqSetName` is provided, the currently active requirement set is used.

```
[xData,yData] = obj.getPECCDFPlot(component,plotAxis,  
    [blockName],[domain])
```

This function returns the CDF plot data between different blocks of a certain *PEC/Total Error* block, axis and component ('TimeConstant', 'TimeRandom' or 'Total'). If no (or an empty) `blockName` is provided, the *Total Error* block is queried, if no `domain` is provided, the 'overall' domain is used.

```
[xData,yData] = obj.getPECCumVarPlot(plotAxis,  
    [blockName],[domain])
```

This function the cumulative variance plot data of a certain *PEC/Total Error* block. If no (or an empty) `blockName` is provided, the *Total Error* block is queried, if no `domain` is provided, the 'overall' domain is used.

```
[xData,yData] = obj.getPECPDFPlot(plotType,plotAxis,  
    [blockName],[domain])
```

This function returns the PDF plot data between different blocks of a certain *PEC/Total Error* block, axis and component ('TimeConstant', 'TimeRandom' or 'Total'). If no (or an empty) `blockName` is provided, the *Total Error* block is queried, if no `domain` is provided, the 'overall' domain is used.

```
[xData,yData] = obj.getPECPSDPlot(plotAxis,[blockName],[domain])
```

This function returns the PSD data of a certain *PEC/Total Error* block. If no (or an empty) `blockName` is provided, the *Total Error* block is queried; if no `domain` is provided, the 'overall' domain is used. If no random process contribution is present in the signal, the both outputs are empty.


```
[xData,yData,rho] = obj.getPECSampleCorrelation(component1,  
    plotAxis1,component2,plotAxis2,[blockName],[domain],  
    [reqSetName],[heatMapFlag])
```

This function returns correlation plot data and the correlation coefficient ρ for a given PEC block name, selected components ('TimeConstant', 'TimeRandom' or 'Total') and axes. If an empty `blockName` is provided, the **Total Error** block is queried; if no (or an empty) `reqSetName` is provided, the currently active requirement set is used; if no `domain` is provided, the 'overall' domain is used; if no `heatMapFlag` is provided, the specified type in the plot option preferences is used.

```
[xData,yData,flag] =  
    obj.getPSDRequirementPlot(plotAxis,[blockName])
```

This function returns the plot data of a spectral requirement function defined for a given PEC block. If an empty `blockName` is provided, the **Total Error** block is queried. The `flag` output determines if the requirement is met for given budget values (0), close to the specified margin (1) or violated (2).

```
[xData,yData,rho] = obj.getSampleCorrelation(blockName1,  
    outputPort1,component1,plotAxis1,blockName2,outputPort2,  
    component2,plotAxis2,[domain],[reqSetName],[heatMapFlag])
```

This function returns correlation plot data and the correlation coefficient ρ for given block names, output ports, components ('TimeConstant', 'TimeRandom' or 'Total') and axes. If no (or an empty) `domain` is provided, the 'overall' domain is queried; if no (or an empty) `reqSetName` is provided, the currently active requirement set is used; if no `heatMapFlag` is provided, the specified type in the plot option preferences is used.

```
[xData,yData] = obj.getSignalPlot(blockName,component,  
    plotAxis,[outputPort],[domain])
```

This function returns the output signal plot data for given block name, output port, component ('crv', 'rv', 'rp', 'd' and 'p' for time-constant random variable, time-random variable, random process, drift or periodic components respectively) and plot axis. For random process components ('rp') PSD plot data is returned and also cross-axis ('xy', 'xz', 'xy', etc.) can be selected as `plotAxis`, For all other components PDF plot data is provided. If no (or an empty) `outputPort` is provided, the first port is queried; if no `domain` is provided, the first/default domain is used.

```
[xData,yData,numArcs] = obj.getSourceInfoFreqAmpPlot(  
    blockName,plotAxis)
```

This function returns the "raw" frequency-amplitude map plot data for periodic error sources as shown in the [Source Info](#) tab of the [Budget Tree View](#). `yData` consists of subsequent pairs of the minimum and maximum amplitude at a given frequency. `xData` thus contains duplicated frequency values. `numArcs` is a scalar value corresponding to the number of frequencies present in a signal and serves as a helper to decompose `xData` and `yData` again.

```
[xData,yData] = obj.getSourceInfoPSDPlot(blockName,plotAxis)
```

This function return the "raw" PSD (or CPSD) plot data for random process error sources as shown in the [Source Info](#) tab of the [Budget Tree View](#). `plotAxis` can represent a single axis ('x', 'y', 'z') or a cross-axis component ('xy', 'xz', 'yz', etc.).

```
[xData,yData,sysDim] = obj.getSystemResponse(blockName,  
[outputPort],[inputPort])
```

This function returns the system response plot data of a dynamic system block. If no (or an empty) `outputPort` or `inputPort` is provided, the first port is queried respectively.

Analysis Features Results

The following functions provide access to the results from analysis features which are displayed in the [Budget Tree View](#) (PES/PEC percentage contributions) and the [Breakdown Tree View](#) (post-processing analyses).

```
[values,rowHeaders,colHeaders] = obj.getPecToPecPercentages(  
srcPecName,tgtPecName,[domain])
```

This function returns the table data of a PEC-To-PEC contribution analysis for a given source PEC (`srcPecName`) to a given target PEC block (`tgtPecName`). If no (or an empty) `domain` is provided, the data for all analysed domains is returned. The first output `values` is a cell array of size `[nDomain*3, n]` which contains the percentage values ($n=4$ in the 3D case for x,y,z and LoS, $n=1$ in the 1D case) for the time-constant, time-random and total contributions in the requested domain(s). The second and third output arguments `rowHeaders` and `colHeaders` contain a description of the data in each row/column in a cell array of size `[nDomain*3, 1]` and `[1, n]` respectively.

```
[values,rowHeaders,columnHeaders] = obj.getPesToPecPercentages(  
pesName,tgtPecName,[domain])
```

This function returns the table data of a PEC-To-PEC contribution analysis for a given PES (`pesName`) to a given PEC block (`tgtPecName`). If no (or an empty) `domain` is provided, the data for all analysed domains is returned. The first output `values` is a cell array of size `[nDomain*3, n]` which contains the percentage values ($n=4$ in the 3D case for x,y,z and LoS, $n=1$ in the 1D case) for the time-constant, time-random and total contributions in the requested domain(s). The second and third output arguments `rowHeaders` and `colHeaders` contain a description of the data in each row/column.

```
postProcStruct = obj.getPostProcResults(postProcName,[pecName])
```

This function returns the entire object substructure containing the results of a post-processing element defined by its `postProcName` at a given PEC block. If no (or an empty) `pecName` is provided, the **Total Error** block is queried.

```
[isEmpty,varargout] = obj.getPostProcData([blockName],postProc,  
component,[domain],dataType,dataIdx)
```

This function returns a specific result subset for a post-processing analysis element at a PEC block.

- `postProc` is either the name of this element or its numeric index in the block structure (the `getPostProcNames` function can be used to identify all elements).
- `component` must be 'TimeConstant', 'TimeRandom', 'Total' for statistical requirements and 'spectral' for spectral requirements.
- `dataType` determines the type of data which is requested ('plot', 'reqPlot', 'pdf', 'cdf', 'pdfabs', 'cdfabs' or 'table').
- `dataIdx` is the index of the function output of the requested post-processing element.

`help engine.Analysis.getPostProcData` from the MATLAB command window provides a detailed description of all returned elements for each `dataType`.

7.8.2 "Browsing" the Analysis Object

Note: The description of the scenario object below is not exhaustive, but just provides a rough orientation. The recommended way to access data is to use the functions provided and described in the other subsections. A manipulation of any setting is nonetheless only possible via specific functions.

The entire analysis object data can also be directly accessed as any usual MATLAB structure object using a dot ("."), e.g. `obj.blocks{2}.outputSignal(1)` to access the first output signal data of the second block in the scenario. As indicated with this expression, the subfields of `obj` can also be cell arrays or arrays of structures.

Budget information typically of interest is directly stored in the `.blocks` structure where each entry in the cell array represents one block present in the scenario.

All blocks also provide an `outputSignal` substructure (which is an array if a block has more than output port) which stores the entire evaluated signal data (error source blocks have in addition a similar `sourceSignal` structure which contains the raw input).

Each output signal of a standard block stores the error signal data separately for each component, e.g. in a field `constRandVar` for the constant random variable component (when multiple domains are defined, this is an array again). Each component stores – amongst other data - information such as the numerical PDF and CDF data, the mean value and standard deviation of the signal and its unit information.

In addition, PEC blocks hold in their `.blocks` structure the time-constant (`constError`), time-random (`randError`) and total error (`totalError`) contributions to statistical requirements with equivalent substructure and contributions to spectral requirements (`spectralError`) which all are an array for each domain.

The statistical error classes contain the PDF and CDF information of the absolute value of the errors, the applied level of confidence and the resulting budget values. The spectral component mainly holds the PSD data and cumulated variance information.

Default plots from the scenario object

Certain data (e.g. PDF, CDF and cumulated variance data) in the analysis object can directly be plotted by appending `.plot` to the requested data object, e.g.:

```
obj.blocks{1}.outputSignal.constRandVar.numpdf.plot
```

to obtain a plot for the PDF of the time-constant variable component of a block. These functions simply plot the “raw” data in SI units for a quick assessment of the evaluated data.

To create a user-defined plot, the plot data can also be obtained using the functions described above or it can be “manually” extracted, e.g. for the above example:

```
obj.blocks{1}.outputSignal.constRandVar.numpdf.x
```

```
obj.blocks{1}.outputSignal.constRandVar.numpdf.y
```

to extract the vectors with abscissa (x) and ordinate data (y) for all present axes.

Tip: Generally, all access functions in the previous subchapter or save functions in the following chapter provide a more comfortable way of dealing with plot data

7.9 Exporting Scenario Results

Helper functions to create and export plot data and to generate spreadsheet reports are available. They rely on the default settings specified in the scenario XML file unless the configuration has been modified using the functions described in chapter 7.5.


Reports

```
obj.generateReport(reqSetName)
```


This function generates an Excel report spreadsheet for the requested requirement set. It is equivalent to using the [Create](#) button in the [File](#) → [Create Report...](#) menu (chapter 6.4.1.1)

Tip: No explicit call of this function is required when the respective flag is set via the `setAutomaticReportFlag` function. In this case, a report is automatically created whenever the `computeBudget` routine is executed.

```
obj.generateRequirementOverview
```

This function generates an Excel report spreadsheet with an overview of all requirement settings present in a scenario. It is equivalent to using the -button in the [Setup](#) → [Scenario Definition](#) menu (chapter 6.4.2.2).

Plots

The functions to save plots are basically equivalent to the functionality behind the -button in the [Plot Window](#) of the GUI. In addition to those already described in section 7.1, the following common parameters are used throughout these functions:

- `fileExt` is a cell array of strings or a string defining the file extensions ('fig', 'png', 'jpeg', 'bmp' or 'pdf') used in function that save plots. When optional and not provided, the scenario plot preferences are used.

- `saveFlag` is logical or numerical flag which determines if the requested plot shall be saved (1) or only its auto-generated file name shall be returned (01). When optional and not provided, the plot data is saved.

Tip: Plots saved by these functions automatically take into account the respective unit assigned to the output.

Tip: Plots can also be automatically saved when generating a report (the plot types which are taken into account depend on the respective flag values set in the **File → Generate Report...** menu (section 6.4.1.1) or manually set via the `setReportPlotFlags` function.

```
fileName = obj.saveCorrelationPlot(  
    blockName1,outputPort1,component1,plotAxis1,  
    blockName2,outputPort2,component2,plotAxis2,  
    [domain],[reqSetName],[fileExt],[saveFlag])
```

This function saves a correlation plot (or returns the figure name) for two given (non-PEC) block names, output ports, components ('TimeConstant', 'TimeRandom' or 'Total') and axes ('x', 'y', or 'z').

```
fileName = obj.saveCPSDPlot(  
    blockName1,outputPort1,blockName2,outputPort2,  
    [domain],[reqSetName],[fileExt],[saveFlag])
```

This function saves a CPSD plot (or returns the figure name) for given (non-PEC) block names and output ports. All cross-axis combinations ('xy', 'xz', 'yx' etc.) are included.

```
fileName = obj.savePECCorrelationPlot(  
    [blockName],component1,plotAxis1,component2,plotAxis2,  
    [domain],[reqSetName],[fileExt],[saveFlag])
```

This function saves a correlation plot (or returns the figure name) for a given PEC block and a selected component ('TimeConstant', 'TimeRandom' or 'Total') and axes ('x', 'y', 'z' or 'los') combination.

```
fileName = obj.savePECPlot(  
    [blockName],signalType,plotCompo,plotType,  
    [domain],[reqSetName],[fileExt],[saveFlag])
```

This function plots PEC block data dependent on the selected signal type, plot component and plot type.

For statistical data (signalType: 'statisticalError'):

- `plotCompo` can be 'TimeConstant', 'TimeRandom' or 'Total'
- `plotType` can be 'abspdf', 'abscdf', 'lospdf' or 'loscdf' for the PDF/CDF of the 'nominal' or absolute value of the error

For spectral data (signalType: 'spectralError'):

- `plotCompo` can only be random process ('rp')
- `plotType` can be PSD ('spectra'), CPSD ('crossspectra'), budget comparison to a requirement ('budgetVsRequirement') or cumulated variance ('cumvar')

```
fileName = obj.savePostProcPlot(  
    [blockName], postProc, dataType, dataId, component,  
    [domain], [reqSetName], [fileExt], [saveFlag], [dimRange])
```

This function saves a plot for a post-processing analysis element at a PEC block.

`postProc` is either the name of this element or its numeric index in the block structure (the `getPostProcNames` function can be used to identify all elements). `dataType` determines the type of data to be plotted ('plot', 'reqPlot', 'pdf', 'cdf', 'pdfabs' or 'cdfabs'). `dataId` is the assigned title string or output index for the requested data in the post-processing element.

The optional argument `dimRange` can be used to plot only a subset of the entire data. If not provided or empty, all output data is included in the plot.

```
fileName = obj.saveSignalPlot(  
    blockName, outputPort, signalType, plotType,  
    plotCmd, [domain], [reqSetName], [fileExt], [saveFlag])
```

This function plots (non-PEC block) data dependent on the selected signal type, plot component and plot type.

For all blocks, the output signal (`signalType: 'outputSignal'`) data can be plotted. For error source blocks with periodic or random process data, also the "raw" signal - w/o statistical interpretation or metrics applied - is available (`signalType: 'sourceSignal'`). Further:

- `plotCompo` can be 'crv', 'rv', 'rp', 'd', or 'p' for time-constant random variable, time-random variable, random process, drift or periodic components respectively
- `plotType` can be 'abspdf', 'abscdf', 'pdf' or 'cdf' for the PDF/CDF of the 'nominal' or absolute value of the error signal; for random process signals 'psd' or 'cumvar' are available to save PSD and cumulated variance data plot data

```
fileName = obj.saveSystemResponsePlot(  
    blockName, [outputPort], [inputPort], [fileExt], [saveFlag])
```

This function saves the system response plot of a dynamic system block for a given input/output combination.

7.10 Support Functions

This section describes a couple of helper functions which are not directly related to the budget evaluation with the tool, but might be useful obtain compatible input data or generate auxiliary output data.

Check realization of General Periodic Error block models

```
checkGeneralPeriodicErrorApproximation(typeID, period, varargin)
```

This function allows a comparison of Fourier series approximation used for the General Periodic Error block (see section 10.9) to the exact error signal. It provides provides the

mean value, standard deviation, worst-case value, CDF plot and related relative errors of the approximation. The following syntax is used for the different options:

- `checkGeneralPeriodicApproximation(1,period,onOffRatio,[order])`
- `checkGeneralPeriodicApproximation(2,period,onOffRatio,[order])`
- `checkGeneralPeriodicApproximation(3,period,decayRate,[order])`
- `checkGeneralPeriodicApproximation(4,period,decayRate,cosFreq,[order])`
- `checkGeneralPeriodicApproximation(5,period,[order])`

where:

- `typeID` is the ID for type of approximation to be checked:
 - 1: rectangular (order in PEET: n=25)
 - 2: triangular (n=25)
 - 3: exponential decay (n=25)
 - 4: exponentially decaying cosine (n=25-50)
 - 5: drift (n=25)
- `period` is the fundamental period T_p of the signal (reset time for drift) in [s]
- `onOffRatio` is the 'duty cycle' for rectangular/triangular options in [%]
- `decayRate` is the positive decay rate r (i.e. $\exp(-r)$) for exponential decay options
- `cosFreq` is the cosine frequency f in [Hz]
- `order` is an optional argument to enforce a series approximation order different to PEET

Time-series generation from PEC data

```
[timeSeries,timeVec,(shapingFilter)] = obj.generateTimeSeries(  
    endTime,samplingTime,[blockName],[domain])
```

This function generates time-series data from the error data at a PEC block of a given length (`endTime`) and sampling (`samplingTime`). The output is a time vector (`timeVec`) and a structure `timeSeries` which contains in each field the time-series data individually for each signal component (fields `crv`, `rv`, `rp`, `d` and `p` for time-constant random variable, time-random variable, random process, drift or periodic components respectively) and the summed overall signal (field `all`). If called with three output arguments, also the shaping filter transfer function which was used to generate time series for the random process signal is returned.

The optional logical `plotFlag` input can be used to suppress the automatic figure generation by setting it to `false` in this case.

PSD Estimation from time-series data

```
[spectrum,freqVec] = getPsdFromTimeSeries(  
    timeVec,timeSeries,[outType],[scale],[outFormat])
```

This function estimates the spectral density from a time-series input. The input must be provided as an $[n \times 1]$ time vector (`timeVec`) and corresponding time series data (`timeSeries`) of size $[n \times m]$. Further:

- `outType` defines the type of spectra which shall be returned. Possible options are 'auto', 'cross' or 'all' to return only the auto- or cross-spectra or both. By default, all information is returned.
- `scale` defines if a power ('psd') or amplitude spectral density ('asd', default) is returned as output.
- `outFormat` defines if the output spectrum shall be returned as matrix ('matrix') or as MATLAB frd object ('frd', default).

Note: This function does not apply any de-meaning or de-trending and uses a rectangular window without overlap/segmentation only. It is only intended to obtain a “quick-look” into the underlying spectrum of time-series (in certain cases, the result can even be further approximated manually by a numerical PSD with only a few points describing linear sections). More detailed algorithms for PSD estimation are considered out of scope for the tool. The implemented method is based on [RD15] which is similar the MATLAB's *pwelch* method

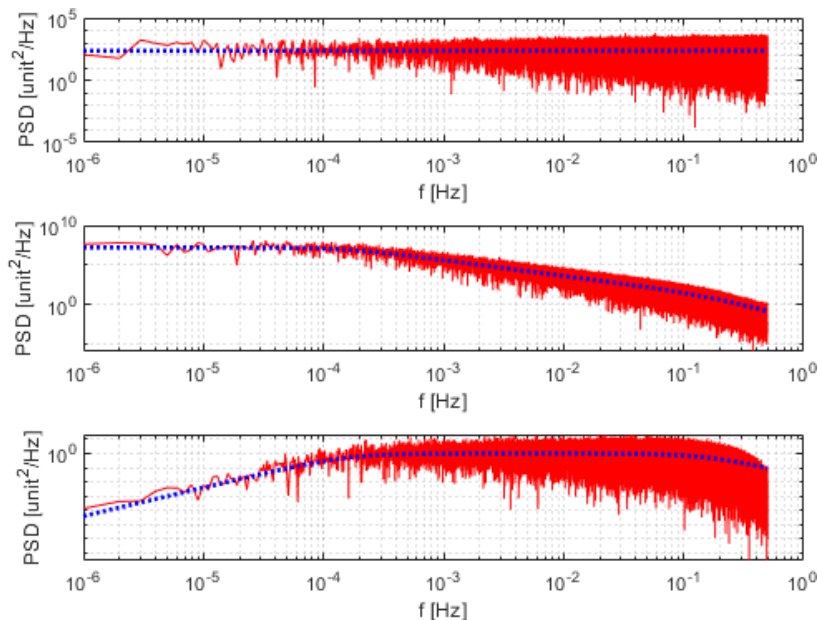


Figure 7-1: Exemplary results from the simple estimation (red) and actual underlying spectrum (blue)

Validity of a Correlation Matrix

```
validCorrMat = obj.getValidCorrelationMatrix(inCorrMat)
```

This function returns a valid correlation matrix `validCorrMat` based on a given input matrix `inCorrMat`. If already valid, the input matrix is returned, if not, a matrix 'close' to the input matrix with feasible values is returned (see [RD7], chapter “Feasibility of Correlation Matrix”).

8 Example Scenarios

This chapter provides simple step-by-step examples (Examples 1-5) for setting up and evaluating a pointing scenario with the tool. The examples are “sequential” in a sense that certain steps are only detailed for the first appearance and considered as known afterwards.

Further, examples for advanced features Examples (6-7) an example of a complex and more realistic scenario study case is provided (Example 8).

Tip: All example scenarios are also available in the `\examples` subfolder of the PEET installation (write-protected). When using these scenarios templates, it is recommended to save them first to another location before executing or modifying them manually ([File → Save As...](#)).

8.1 Example 1: Statistical Budgets

This example shows how to set up and evaluate a scenario for statistical budgets, how to map from the ECSS “statistical interpretation” concept to the “domain treatment” concept in PEET and how to change error source units.

8.1.1 Description

Assume, a set of statistical requirements for an absolute performance error index are defined as follows (not intended to be a realistic set of requirements!) for the overall error:

- “Ensemble” statistical interpretation applies with a 3-Sigma level of confidence
- “Temporal” statistical interpretation applies with a 2-Sigma level of confidence
- “Mixed” statistical interpretation applies with a 1-Sigma level of confidence

The pointing scenario is fully defined by two time-random variable error sources (e.g. assume that these sources already represent the collected budgets for two main driving subsystems):

| PES | RV distribution | Distributed Parameter | Parameter values [mrad] |
|-----|-------------------------------|--|---|
| 1 | Uniform: $U(-C, C)$ | Range C: Bimodal BM(C_{min}, C_{max}) | C_{min} : [0 2 4] C_{max} : [3 6 10] |
| 2 | Gaussian: $G(\mu, \sigma)$ | Standard deviation σ : Uniform $U(\sigma_{min}, \sigma_{max})$ | μ : [0 0 0] σ_{min} : [1 1 2] σ_{max} : [2 3 3] |

8.1.2 Setup in PEET

- 1) Open PEET by typing `peet` in the MATLAB command window.
- 2) Open the [Database Browser](#), drag two *PES (Time-Random)*, a *Summation* and a *Total Error* block in the [System Editor](#) window. Connect the source blocks to the *Summation* block and the latter to the *Total Error* block.

- 3) Double-click on the names of the two *PES (Time-Random)* blocks and rename them to “PES 1” and “PES 2” respectively.

The **System Editor** should look like Figure 8-1 now. In the next steps, the error sources need to be configured.

- 4) Open the setup dialog of the first *PES (Time-Random)* block, and set the *Signal class* to **Uniform** and the *Distributed parameter* to **Range**. Then enter the C_{\min} values for the *Lower bound* and the C_{\max} values for the *Upper bound*.
- 5) Choose **Custom...** from the *Output unit* drop down-list to open the **Change Unit** dialog.
- 6) As **Radian** is already selected from the *Angular Unit group*, only the prefix of the unit needs to be adapted. Choose **Milli-** from the *Prefix & unit* checkbox, then press the top **Add** button to add the unit “mrad” to the nominator. Confirm the settings with **OK** button.
- 7) Confirm all settings for the PES again with the **OK** button to close the dialog window.

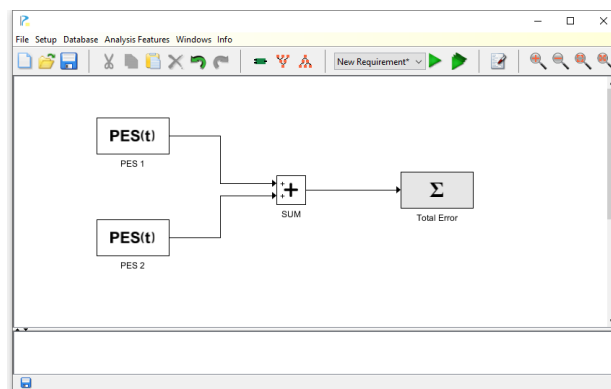


Figure 8-1: System Editor window after step 3

- 8) Open the setup dialog of the second *PES (Time-Random)* block, and keep the *Signal class* as **Gaussian**. Then set the *Distributed parameter* to **Standard deviation**. As all mean values are already zero by default, it is only necessary to define the **Uniform Distribution** parameters in the **Standard deviation** panel. Enter the σ_{\min} values for the *Minimum* and the σ_{\max} values for the *Maximum*. Confirm the settings with the **OK** button respectively.

Now, all error sources are completely defined, the requirement sets need to be specified.


- 9) Open the **Setup** → **Scenario Definition** menu.
- 10) Choose “APE Ensemble” as *Name* for the first requirement set and **Absolute Performance Error** as *Type* for the **Error index**.

“Ensemble” statistical interpretation is equivalent to worst case treatment of temporal properties and a statistical treatment of ensemble properties (mouse-over the ? -icon to obtain a quick-help for the mapping). Thus:

- 11) Set the **Domain treatment** for the *Ensemble domains* to **Statistical**.

- 12) Set the **Value** for the **Level of confidence** to 99.73 [%] equivalent to a “3-Sigma level”.


The first requirement set is now fully specified.

- 13) Use the  - icon add the template for the next set and enter “APE Temporal” as **Name**. Choose **Absolute Performance Error** as **Type** for the **Error index**.

“Temporal” statistical interpretation requires the opposite domain treatment as the ensemble interpretation, i.e. a worst-case treatment of ensemble properties and a statistical treatment of temporal properties). Thus:

- 14) Set the **Domain treatment** for the **Temporal domain** to **Statistical**.
- 15) Set the **Value** for the **Level of confidence** to 95.45 [%] equivalent to a “2-Sigma level”.

Finally, the last first requirement set needs to be configured. “Mixed” statistical interpretation implies statistical treatment in all domains.

- 16) Use the  - icon add the template for the next set and enter “APE Mixed” as **Name**. Choose **Absolute Performance Error** as **Type** for the **Error index**.
- 17) Set the **Domain treatment** for the **Ensemble domains** and the **Temporal domain** to **Statistical**.
- 18) Set the **Value** for the **Level of confidence** to 68.27 [%] equivalent to a “1-Sigma level”. Confirm all settings with the **OK** button.

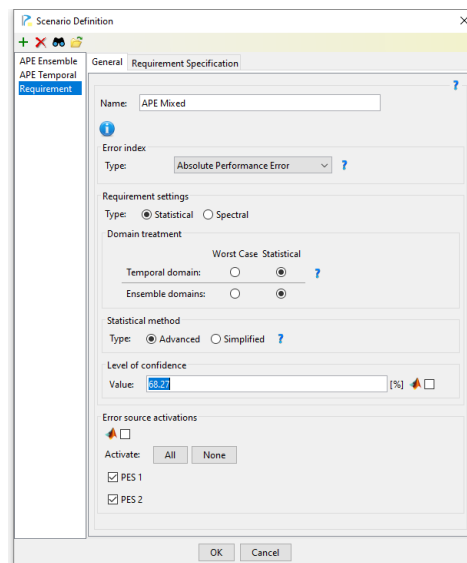





Figure 8-2: Scenario Definition menu after step 15

At this point, the scenario is fully prepared for an evaluation. The definition of value for the requirements and additional PES or evaluation dependencies are treated in later examples.

- 19) Save the scenario as “Example1.peet” to an arbitrary location using the  - button.
- 20) Evaluate all requirement sets using the  -icon.

The GUI gets locked during the computation and shows the currently executed steps in the [Execution Log](#) until the computation is completed and GUI gets unlocked again. Then, the results can be inspected in the [Budget Tree View](#).

- 21) Click the  -icon to open the [Budget Tree View](#). Make sure that the set **APE Ensemble** is selected in the drop-down list on the top-left of the window. Then click first on “PES 1” and inspect the information panel on the right, then repeat the inspection for “PES 2”:

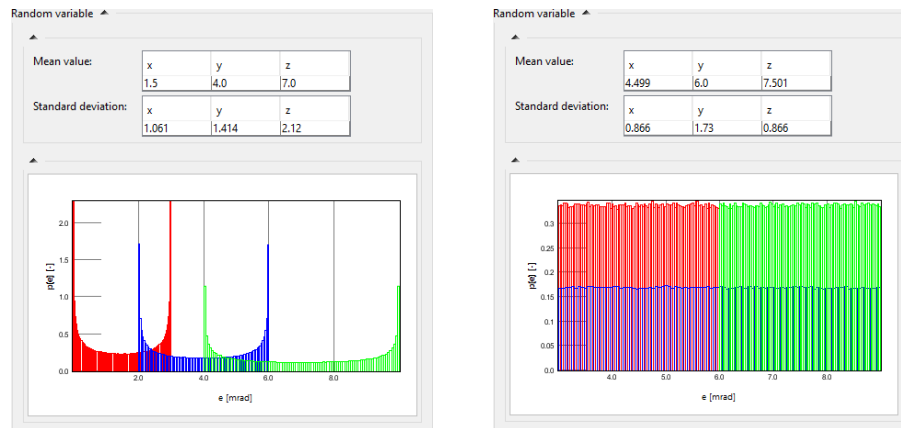


Figure 8-3 Information panel content for PES 1 (left) and PES 2 (right) for “APE Ensemble”

As only random variables are defined, the **Random variable** panel is also the only signal contribution where content is present. For both PES, the **Mean value** and **Standard deviation** of the current signal are shown together with a plot preview for the underlying PDFs. As “ensemble interpretation” applies, the plots correspond to the distribution of the worst-case values in time that occur for each realization of the ensemble parameter. For PES 1 this corresponds to a bimodal distribution, for PES 2 to a uniform distribution.

- 22) Select **APE Temporal** from the drop-down list on the top-left of the [Budget Tree View](#) window. Repeat the inspection for PES 1 and PES 2.

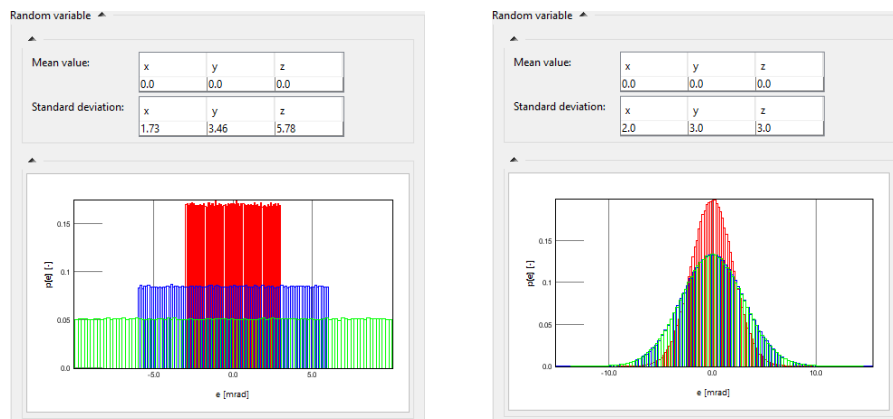


Figure 8-4 Information panel content for PES 1 (left) and PES 2 (right) for “APE Temporal”

In this case, “temporal interpretation” applies. Thus, the PDF plots correspond to the temporal distribution that contains the worst-case value out of all realizations of the ensemble parameter. For PES 1 this leads to a uniform distribution with a range given by C_{max} , for PES 2 to a Gaussian distribution with the maximum value for the standard deviation σ_{max} .

- 23) Select **APE Mixed** from the drop-down list on the top-left of the **Budget Tree View** window. Repeat the inspection for PES 1 and PES 2.

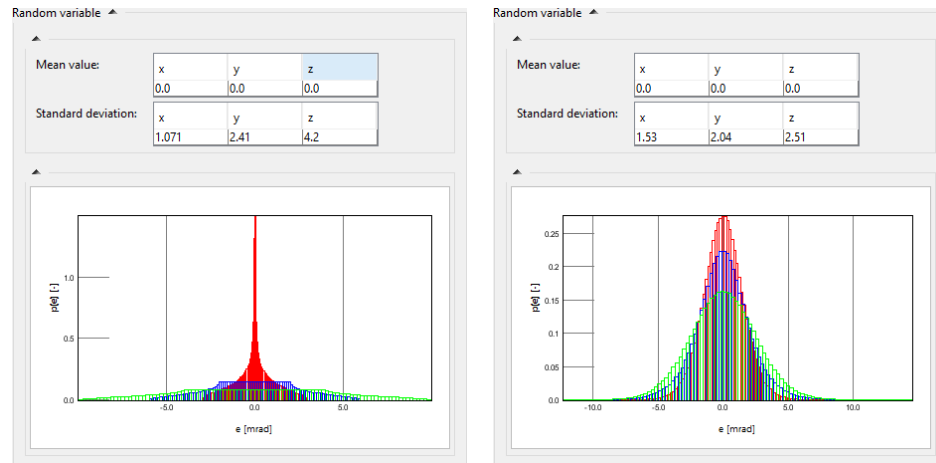


Figure 8-5 Information panel content for PES 1 (left) and PES 2 (right) for “APE Mixed”

Finally, “mixed interpretation” applies. Thus, the PDF plots correspond to the statistics of all temporal realizations with for all realizations of the ensemble parameter². For PES 1 this leads to a uniform “core” around zero (in the common range covered by all ensemble realizations) and “bimodal tails” between C_{min} and C_{max} . The situation is generally identical for the Gaussian (“Gaussian core, Uniform tails”) but is merely visible as the Gaussian itself is already too heavily tailed.

Finally, the contribution of both PES to the overall error is of interest.

- 24) Click on the **Total Error** block and inspect the results in the information panel sequentially for three requirement sets.

As shown in Figure 8-6, tabular data is present that summarizes the **Time-constant**, **Time-random** and **Total Error contributions**. The values in the table represent the error bounds which are not exceeded for the given level of confidence (**LoC**) of the requirement set. As only time-random contributions are present, these also correspond to the total contributions in this case. The PDF plots indicate the underlying distributions again while different to the signal contributions of “standard” blocks, the PDF of the absolute value of the error is shown.

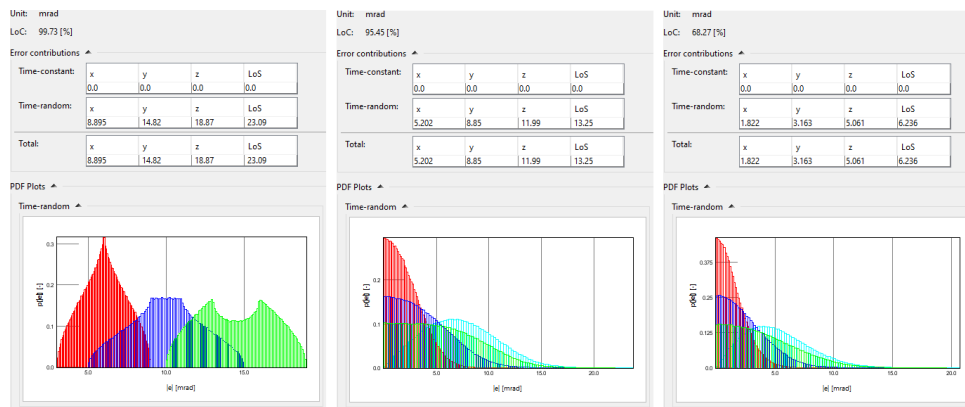


Figure 8-6 Information panel content for Total Error block: Ensemble SI (left), Temporal SI (centre), Mixed SI (right)

8.2 Example 2: Spectral Budgets & Requirement

This example shows how to set up and evaluate a scenario for spectral budgets and how to define spectral requirements.

8.2.1 Description

Assume, a 1D spectral requirement for a relative performance error index is defined as follows (not intended to be a realistic set of requirements!):

- A spectral requirement defined by an analytical function in a given bandwidth for the total error contribution

$$S_1(f) = 0.001 \cdot f / (f + 2) \quad [rad / \sqrt{Hz}] \quad 0.01Hz < f < 100Hz$$

- Spectral requirements defined by an analytical function in a given bandwidth for the “subsystems”

$$S_{2,3}(f) = 0.0005 \cdot f / (f + 2) \quad [rad / \sqrt{Hz}] \quad 0.01Hz < f < 100Hz$$

- A window time of 1s for the RPE metrics

The pointing scenario is fully defined by two band-limited white noise error sources for the (e.g. assume that these sources already represent the collected budgets for two main driving subsystems):

| PES | RP type | Standard deviation [rad] | Bandwidth [Hz] |
|-----|---------|--------------------------|----------------|
| 1 | BLWN | 0.003 | 200 |
| 2 | BLWN | 0.0006 | 200 |

8.2.2 Setup in PEET

- 1) Open PEET by typing `peet` in the MATLAB command window.

- 2) Open the [Database Browser](#), drag one *PES (Time-Random)*, one *PEC* block, a *Summation* and a *Total Error* block in the [System Editor](#) window.
- 3) Double-click on the name of the *PES (Time-Random)* block and rename it to “PES 1”.
- 4) Open the setup dialog of “PES 1”, and set the [Signal dimension](#) to **1D**. Then set the [PES description](#) to **Random Process** and the [Type](#) to **BLWN**.
- 5) Enter the [Noise bandwidth](#) and the [Standard deviation](#) for PES 1. Confirm all settings with the **OK** button.
- 6) Select “PES 1” in the [System Editor](#) and copy it via keyboard commands (CTRL-C, CTRL-V).
- 7) Open the setup dialog of the created “PES 2”, modify the numbers for the parameters and confirm with **OK**.
- 8) As **Radian** is already selected from the [Angular Unit group](#), only the prefix of the unit needs to be adapted. Choose **Milli-** from the [Prefix & unit](#) checkbox, then press the top **Add** button to add the unit “mrad” to the nominator. Confirm the settings with **OK** button.
- 9) Confirm all settings for the PES again with the **OK** button to close the dialog window.
- 10) Open the *PEC* block and the *Total Error* block and explicitly set their [Signal dimension](#) to **1D**. Then copy the *PEC* block.
- 11) Connect the source blocks to the *PEC* blocks, the *PEC* blocks to the *Summation* block and the latter to the *Total Error* block.

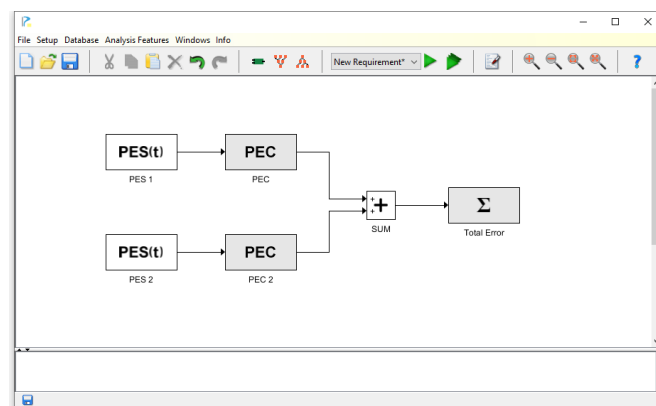


Figure 8-7: System Editor window after step 11

The [System Editor](#) should now look as in Figure 8-7. The next step is to define the spectral requirements.

- 12) Open the [Setup](#) → [Scenario Definition](#) menu.
- 13) Choose “Spectral Requirement” as [Name](#) for the first requirement set and [Relative Performance Error](#) as [Type](#) for the [Error index](#). Then enter the [Window time](#).

- 14) In the **Requirement settings** panel, choose **Spectral** as **Type**. Then specify the **Bandwidth** for the spectral requirements, i.e. 0.01 as **Lower Bound** and 100.0 as **Upper bound**).
- 15) Switch from the **General** tab to the **Requirement Specification** tab to further define the individual requirements. Enter the **Requirement Function** for the different blocks using element-wise operations (e.g. $0.001 * f ./ (f+2)$ for the Total Error) and corresponding ID (e.g. S1 for the Total Error).

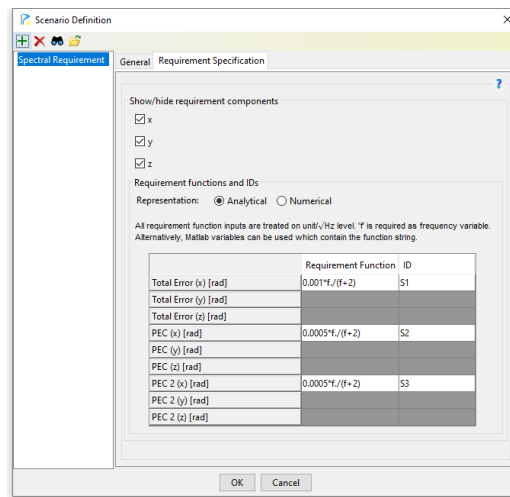




Figure 8-8: Requirement Specification tab after step 15

- 16) Confirm all settings for the with the **OK** button to close the dialog window.
- 17) Save the scenario as “Example2.peet” to an arbitrary location. Then evaluate the scenario using the -icon and wait until the computation is completed.
- 18) Click the -icon to open the **Breakdown Tree View**. Inspect the tree view and sequentially click the blocks to display data in the information panel.

The tree on the left of the window shows all PEC and Total Error in a hierarchical structure. The blocks contain the IDs of any related requirements and a colour-coding for the “state” of the requirement. In this example, the requirement at the first PEC is violated while both the requirements of PEC 2 and for the total error are met. A closer inspection of the plots gives the following conclusion:

- The requirement at the PEC is violated in the range up to about 1Hz.
- The requirement at PEC 2 is safely met over the entire requirement bandwidth

The requirement on the total error can still be met, although there is only few margin between 0.001 Hz and 1Hz (the “margin warning” indicated by yellow colour - see section 6.4.2.3 - is however not yet triggered, as the margin still exceeds the default value of 10% over the entire band).

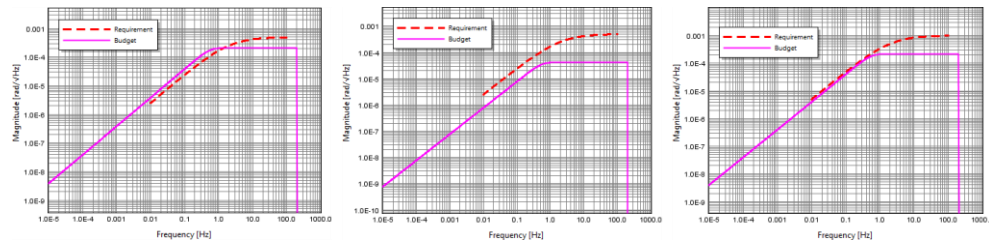


Figure 8-9 Comparison of requirement and budget in Breakdown Tree View: PEC (left), PEC 2 (centre) and Total Error (right)

8.3 Example 3: User-defined Domains

This example shows how to set up and evaluate a scenario for statistical budgets with user-defined ensemble domains and how to use *Container* blocks.

8.3.1 Description

Assume, two sensors are known to be the dominating error sources of an APE mixed required with a 95.45% level of confidence “2-Sigma”. Each sensor basically contributes via a bias induced by misalignments and a temporal noise contribution.

For simplicity, the noise properties of both sensors are considered to be identical in this example.

All alignment errors are further assigned to a domain “Misalignment”, all temporal noise contributions to a domain “Noise” to able to assess also the contributions from different error groups.

| PES | Properties | Parameter values [mrad] |
|--------|--|--|
| Biases | Time-constant, Truncated-Gaussian, Symmetric bound | μ : [0 0 0] σ : [6 6 6] STB: [15 15 15] |
| Noises | Time-random, Random process, BLWN | Bandwidth: 10 Hz σ : [3 3 3] |

8.3.2 Setup in PEET

- 1) Open PEET by typing `peet` in the MATLAB command window.
- 2) Open the [Database Browser](#), drag a *Summation* block, a *Container* and a *Total Error* block in the [System Editor](#) window.
- 3) Double-click on the name of the *Container* and rename it to “Sensor 1”. Then double-click on the block itself to open its internal editor window.
- 4) Drag one *PES (Time-Random)*, one *PES (Time-Constant)* and a *Summation* block to this editor window.

- 5) Delete the *Input Port* block and connect the PES blocks to the *Summation* block and the latter to the *Output Port*.
- 6) Rename the *PES (Time-Constant)* block to “Bias” and define the *Truncated Gaussian* with the mentioned parameters in its setup dialog. Confirm the settings with the *OK* button.
- 7) Rename the *PES (Time-Random)* block to “Noise” and define the *BLWN* with the mentioned parameters in its setup dialog. Confirm the settings with the *OK* button.
- 8) Close the “Sensor 1” editor window.

Now the entire content of the subsystem “Sensor 1” is available as one single block on top-level.

- 9) Copy and paste the “Sensor 1” Container. Connect the two *Container* blocks on top-level to the *Summation* block and the latter to the *Total Error* block.

The error signal flow of the scenario is now complete. The next step is to define the user-defined ensemble domains and to assign to errors sources to each of them.

- 10) Open the *Setup* → *Ensemble Domains...** menu. In the dialog, press the *Add* button and define the first domain called “Misalignment”. Repeat this step and create the second domain “Noise”.
- 11) Assign the bias contributions to the “Misalignment” domain and the noise contributions to the “Noise” domain in the *Error source assignment* panel. Confirm all settings with the *OK* button.

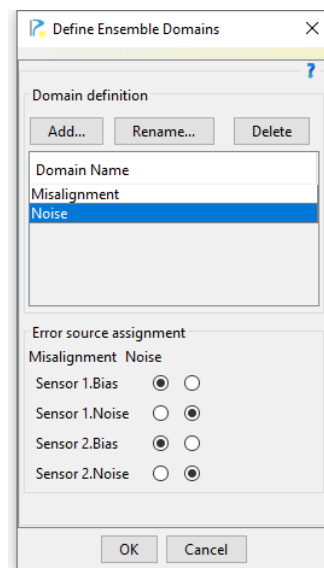




Figure 8-10 Ensemble Domains menu

In the last step before the evaluation, the requirement set needs to be defined using the *Setup* → *Scenario Definition* menu. Different to Example 1, now the *Domain treatment* panel allows selecting the statistical treatment for each user-defined ensemble domain. Furthermore, an option becomes available to specify the *Level of confidence evaluation* (*Common* or *Individual*) in the *Level of confidence* panel. In this example both options are used by setting up two different requirement sets to highlight the impact of this option.

Note: An unambiguously defined requirement set also needs a clear definition how the level of confidence shall be evaluated if multiple ensemble domains are present (in case of a single domain, the distinction between **Common** and **Individual** is not necessary). These options do not represent an additional “degree of freedom”.

- 12) Choose “Common” as *Name* for the first requirement set and **Absolute Performance Error** as *Type* for the **Error index**.
- 13) Set all **Domain treatment** options to **Statistical** to realize a “mixed interpretation” for all domains.
- 14) Set the *Value* for the **Level of confidence** to 95.45% and maintain the **Common** evaluation.
- 15) Create a second identical requirement set named “Individual”. Just change the **Level of confidence evaluation** to **Individual**. Specify a value of 95.45% for both domains.
- 16) Save the scenario as “Example3.peet” to an arbitrary location. Then evaluate the scenario using the -icon and wait until the computation is completed.
- 17) Click the -icon to open the **Budget Tree View**. Make sure that the **Common** requirement set is selected in the first drop-down list on the top-left of the window.

Having defined multiple ensemble domains, a second drop-down list for the domain selection is present in the toolbar.

- 18) Make sure that **Misalignment** is selected in this list. Then click on the “SUM” block and inspect the information panel on the right. Then choose the **Noise** domain and repeat the inspection.

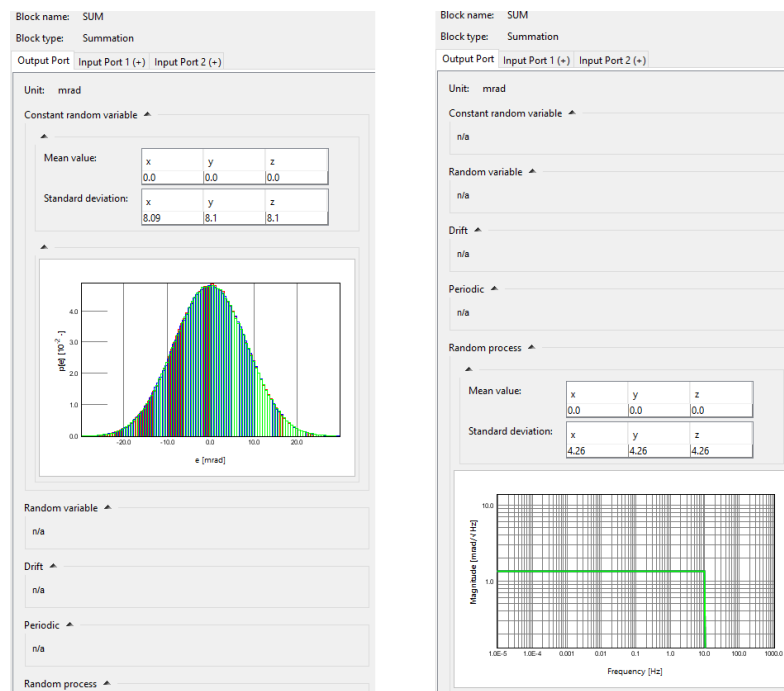


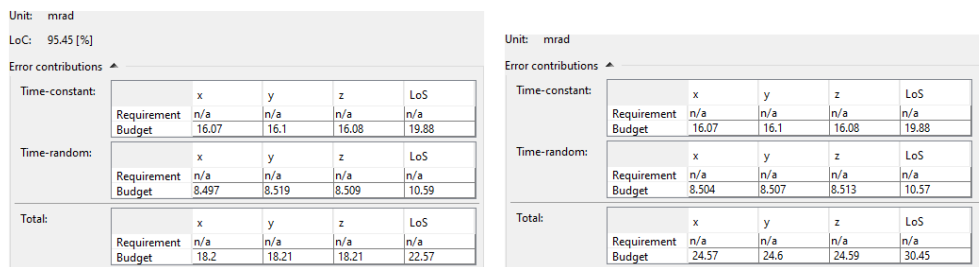
Figure 8-11 Information panel content for SUM: domain “Misalignment” (left) and domain “Noise” (right)

In this specific case the “misalignment” domain only hold constant random variable contributions from the biases and the “noise” domain only holds random process contributions. This clear distinction by error classes is however not mandatory. In more complex systems various signal classes can be present in a domain or split over different domains (e.g. noises and biases dependent on temperature in one domain, noise and biases dependent on inertial attitudes in another).

- 19) Choose the **Individual** requirement from the first drop-down list in the **Budget Tree View**. The inspect the information panel of the “SUM” block again for the **Misalignment** and **Noise** domain.

Note that there is no difference at all concerning the error signals and the statistical properties at that point, as the evaluation option only affects the evaluation of these signals at the **PEC** or **Total Error** blocks.

- 20) Click on the **Total Error** block and inspect its information panel for the **Individual** requirement set and the **Overall** error contribution. Then repeat the inspection for the **Common** requirement set.



| Unit: mrad | | LoC: 95.45 [%] | | | |
|---------------------|-------|----------------|-------|-------|-----|
| Error contributions | | | | | |
| Time-constant: | | | | | |
| | | x | y | z | LoS |
| Requirement | n/a | n/a | n/a | n/a | n/a |
| Budget | 16.07 | 16.1 | 16.08 | 19.88 | |
| Time-random: | | | | | |
| | | x | y | z | LoS |
| Requirement | n/a | n/a | n/a | n/a | n/a |
| Budget | 8.497 | 8.519 | 8.509 | 10.59 | |
| Total: | | | | | |
| | | x | y | z | LoS |
| Requirement | n/a | n/a | n/a | n/a | n/a |
| Budget | 18.2 | 18.21 | 18.21 | 22.57 | |

| Unit: mrad | | LoC: 95.45 [%] | | | |
|---------------------|-------|----------------|-------|-------|-----|
| Error contributions | | | | | |
| Time-constant: | | | | | |
| | | x | y | z | LoS |
| Requirement | n/a | n/a | n/a | n/a | n/a |
| Budget | 16.07 | 16.1 | 16.08 | 19.88 | |
| Time-random: | | | | | |
| | | x | y | z | LoS |
| Requirement | n/a | n/a | n/a | n/a | n/a |
| Budget | 8.504 | 8.507 | 8.513 | 10.57 | |
| Total: | | | | | |
| | | x | y | z | LoS |
| Requirement | n/a | n/a | n/a | n/a | n/a |
| Budget | 24.57 | 24.6 | 24.59 | 30.45 | |

Figure 8-12 Overall contribution at Total Error block for Common (left) and Individual (right) evaluation of the level of confidence

For this example, it is sufficient to focus on the **Overall** contributions as the domain contributions are in this case equivalent to the **Time-constant** and **Time-random** contributions, i.e. all “Misalignment” errors map to **Time-constant** and all “Noise” errors to **Time-random** (in general, this is not the case).

What can be implied from the result is that the **Total** contribution computed with the **Individual** evaluation is more conservative than the one from the **Common** evaluation (which is equivalent to totally neglecting domains – presumed that the same statistical treatment is defined for each domain).

This is the consequence of summing the “2-Sigma” errors from each domain rather than computing the “2-Sigma” errors from the “joint” error information of all domains.

8.4 Example 4: Correlation

This example shows how to set up correlation between different error sources, how correlation affects the results and how correlation information can be “extracted” from a signal. The example is not built up from scratch, but loaded from a template which is then completed step by step.

8.4.1 Description

This example covers the case of two artificial time-random PES which are defined as random variables and whose parameters are each subject to an ensemble distribution.

Two requirement sets are defined (APE, with 95% level of confidence): “R1” where ensemble interpretation applies and “R2” where “temporal” interpretation applies.

The following correlation properties need to be realized before the execution:

| Dependency | Value | Correlation |
|--------------------|-------|----------------------|
| PES1 y with PES2 y | 0.38 | temporal correlation |
| PES1 x with PES2 x | 0.7 | ensemble correlation |

8.4.2 Setup in PEET

- 1) Load `Example4.peet` from the `\examples` subfolder of the PEET installation. Save a copy with the same name at an arbitrary location.
- 2) Open the **Setup** → **Set Correlations...** menu.
- 3) In the **Ensemble** tab, set the entry PES1|x, PES2|x to 0.7 to define the correlation of the ensemble properties between the x-axes of the two sources.
- 4) In the **Temporal** tab, set the entry PES1|y, PES2|y to 0.38 to define the correlation of the temporal properties between the y-axes of the two sources.
- 5) Confirm the settings with the **OK** button and save the scenario again.

Now, the scenario is ready for evaluation.

Tip: The setup of coherences between random process error sources and phase relations relation between periodic sources is realized in a similar manner user the **Setup** → **Set Coherence...** / **Setup** → **Set Phase Relations...** menus. As for the correlations, only “valid” sources are displayed in the respective matrices.

- 6) Select the requirement set **R1** from the drop-down list in the toolbar and evaluate this set only using the ▶-icon.
- 7) After the computation is completed, open the **Budget Tree View**, choose **R1** from the drop-down list in the tool bar and click on the “PES 1” block. Then double-click the plot preview in the **Random variable** section of the information panel.
- 8) In the **Plot Window**, click the **Correlation** tab.
- 9) Choose **Correlation with: PES 2, Output Port, x, RV** and press the refresh button (↻-icon).
- 10) Choose **Component: y, Correlation with: PES 2, Output Port, y, RV** and press the refresh button.

“R1” is a requirement where ensemble interpretation applies, thus the statistics of the error sources are related to the ensemble properties of the sources. In the first case (step 9), the computed correlation coefficient is found to be the one specified for the ensemble correlation between the x-axes of the two sources. For the y-axes, no ensemble correlation has been defined and the specified temporal correlation has no effect.

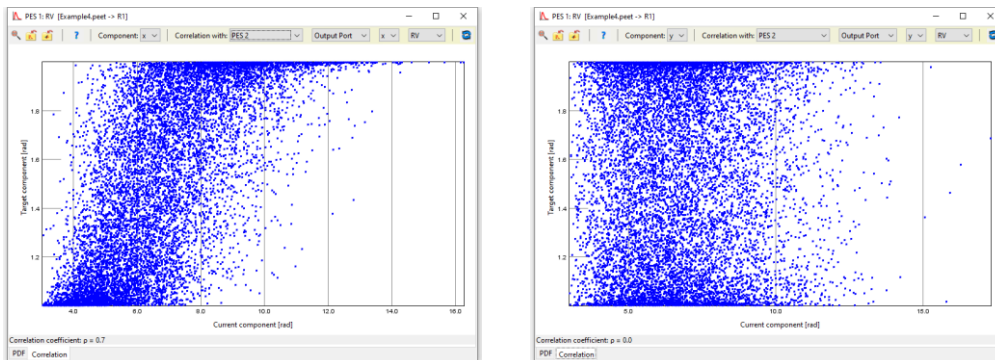


Figure 8-13 Scatter plot indicating ensemble correlation between x-axes (left) and y-axes (right) of the two error sources

- 11) Select the requirement set **R2** from the drop-down list in the toolbar and evaluate the scenario this set (▶-icon).
- 12) After the computation is completed, open the **Budget Tree View**, choose **R2** from the drop-down list and click on the “PES 1” block. Then double-click the plot preview in the **Random variable** section of the information panel and switch to the **Correlation** tab.
- 13) Choose **Component: x**, **Correlation with: PES 2**, **Output Port, x**, **RV** and press the refresh button.
- 14) Choose **Component: y**, **Correlation with: PES 2**, **Output Port, y**, **RV** and press the refresh button.

“R2” is a requirement where temporal interpretation applies, thus the statistics of the error sources are related to the temporal properties of the sources. In the first case (step 13), the computed correlation coefficient is found to be the one specified for the temporal correlation between the y-axes of the two sources. For the x-axes, no temporal correlation has been defined and the specified ensemble correlation has no effect.

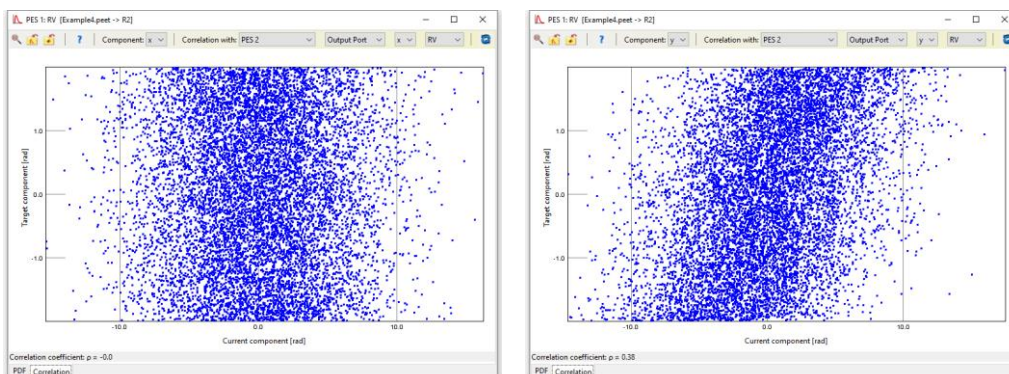


Figure 8-14 Scatter plot indicating temporal correlation between x-axes (left) and y-axes (right) of the two error sources

- 15) In the **Budget Tree View**, choose first **R1** from the drop-down list in the tool bar, click on the “SUM” block and inspect the **Output Port** tab of the information panel. Then repeat the inspection for the results of **R2**.

In both cases the correlation “imprinted” to the error signals affects the statistical moments of the sum. In particular, the contributions on axes where correlation is present are large than those on axes without correlation. The standard deviations of the summation block output follow the well-known relation:

$$\sigma_{1+2}^2 = \sigma_1^2 + \sigma_2^2 + 2Cov_{1,2} = \sigma_1^2 + \sigma_2^2 + 2\rho_{12}\sigma_1\sigma_2$$

where the indices 1,2 are placeholder for the components of the axis under consideration and the “right-hand side” contains the standard deviations of the input signals and the specified correlation coefficients.

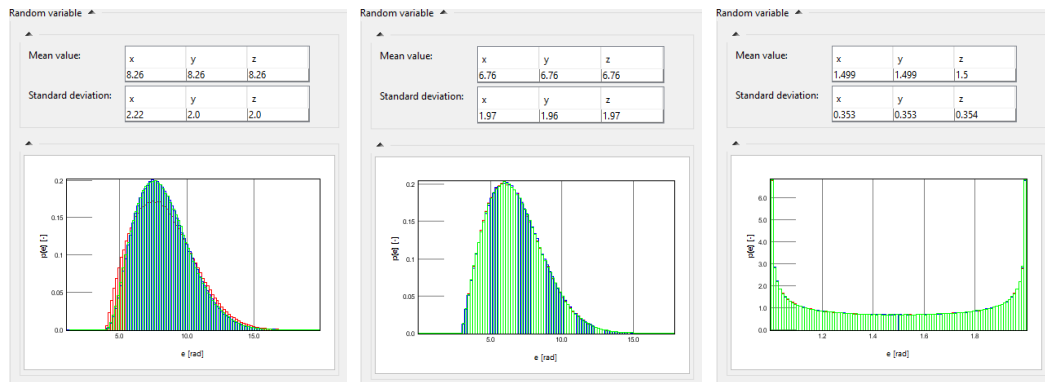


Figure 8-15 Error contribution of the summed signal (left), PES 1 (centre) and PES 2 (right) for requirement set “R1”

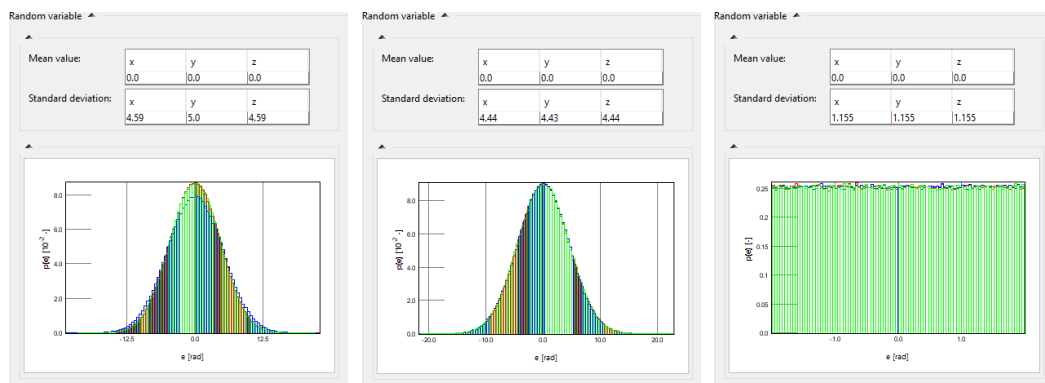


Figure 8-16 Error contribution of the summed signal (left), PES 1 (centre) and PES 2 (right) for requirement set “R2”

8.5 Example 5: Frequency Bandwidth and Refinement

This example covers the application of the optional features that provide an overview of the scenario frequency “content” and allow refining the frequency grid for the evaluation of random process error signals.

8.5.1 Description

By default, PEET uses a log-spaced initial frequency grid for the evaluation of the random process variance. If a spectrum has significant features (e.g. peaks due to poles) which are not covered by the initial grid, the computed variance may not be as exact as required.

In this example, a single random process error source is present which has a spectrum whose magnitude is represented by the transfer function of a second order lag element of the form:

$$P(s) = \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2}$$

where a single “peak” is present and $\omega_0 = 2\pi f_0$ determines the frequency location of the peak. Furthermore, the initial frequency grid evaluation is chosen to be intentionally sparse with only 3 *Grid points per decade* (specified in the [Setup → Evaluation Settings](#)) menu.

With this given setup, the tool “sees” only an almost flat grid as shown in the figure below (the evaluation bandwidth is equivalent to the range shown as well, i.e. 1 Hz to 10 Hz):

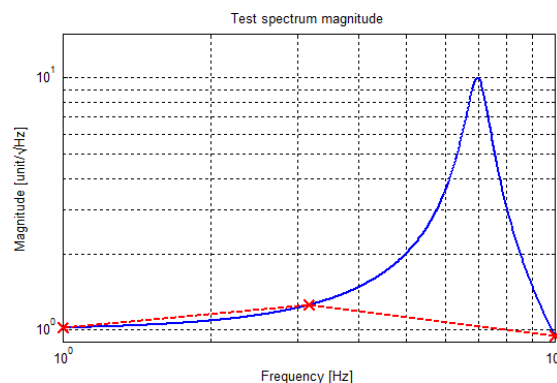


Figure 8-17 Real spectrum magnitude (blue) and as ‘seen’ from the sparse initial frequency grid

To determine the actual value for the random process standard deviation, first the spectrum has to be expressed on power level by squaring the magnitude:

$$G(s) = \left(\frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \right)^2$$

Then the resulting expression needs to be integrated over the evaluation bandwidth:

$$\sigma_{RP} = \sqrt{\int_{f=1\text{Hz}}^{f=10\text{Hz}} |G(f)| df}$$

This gives an exact reference value for the standard deviation of the random process of 10.3533 [mrad].

8.5.2 Setup in PEET

- 1) Load `Example5.peet` from the `\examples` subfolder of the PEET installation. Save a copy with the same name at an arbitrary location.
- 2) Evaluate the scenario.
- 3) When the computation is complete, open the [Budget Tree View](#), click on the *PES (Time-Random)* block and inspect the plot preview in the information panel.

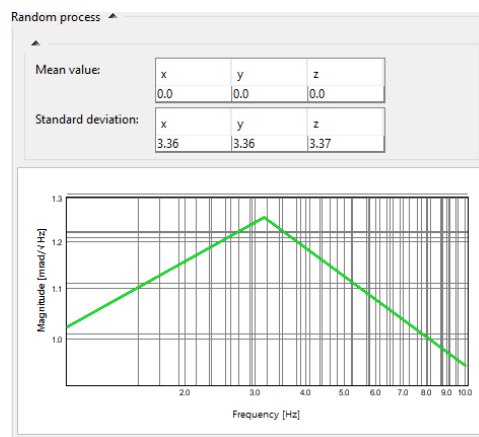


Figure 8-18: Random process contribution with (intentionally) sparse grid

As predicted the tool is not able to compute the “correct” random process standard deviation with the sparse frequency grid provided as the peak at around 7 Hz is not covered. The relative error with respect to the reference value is about 67%.

To first detect any system and source features, the scenario frequency content needs to be analysed.

- 4) Open the [Setup → Check Frequency Bandwidth...*](#) menu and inspect the [Frequency overview](#) panel.

The evaluation bandwidth is shown in Figure 8-19 as red (dashed) vertical lines. Furthermore, the peak in the spectrum of PES (Time-Random) at around 7 Hz is detected as feature (black cross) within this bandwidth. PEET provides the option to refine the grid around such detected features.

- 5) Close the [Setup → Check Frequency Bandwidth...*](#) menu. Then open the [Setup → Refine Frequency Grid...*](#) menu and inspect the [Frequency overview](#) panel.

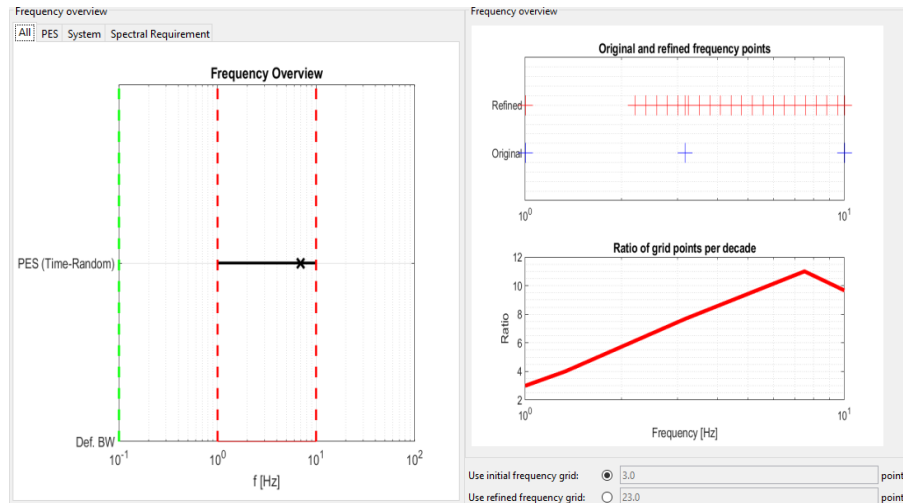


Figure 8-19 Frequency overview in Check Frequency Bandwidth (left) and Refine Frequency Grid (right) menus

The first subplot shows the initial log-spaced frequency grid (blue) and a refined version suggested by the tool that concentrates further points in the frequency decade around identified features. The grid “density” compared to the initial is further displayed in lower subplot. The density of the refined grid in this case has the largest value around the peak at 7 Hz as desired.

The next step is to repeat the evaluation with the refined grid.

- 6) Check **Use refined frequency grid** at the bottom part of the menu dialog. Confirm the settings with the **OK** button.
- 7) Evaluate the scenario again.
- 8) Open the **Budget Tree View**, click on the **PES (Time-Random)** block and inspect the new result in the plot preview in the information panel.

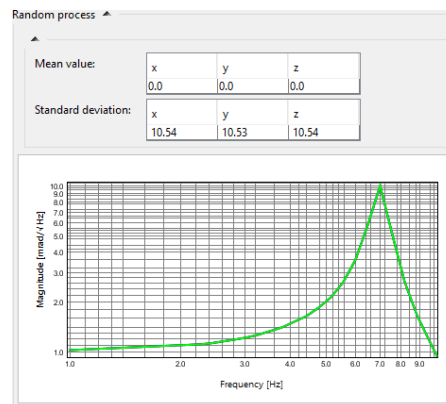


Figure 8-20: Random process contribution with refined grid

With the refined grid, the accuracy of the computed standard deviation is significantly improved and the shape of the spectrum is also properly represented. The relative error with respect to the reference value is reduced from 67% to about 1.7% percent.

Note however that this example represents an extreme case. Usually the default settings for the grid points per decade (100 points) produces an already dense enough grid to cover most features that might be present. The refinement is only necessary, if especially high accuracy is required or if extremely “narrow” features are present. The result obtained with the default grid for this example would even be more accurate than using the refinement based on the initial sparse grid (0.03% relative error, see figure below).

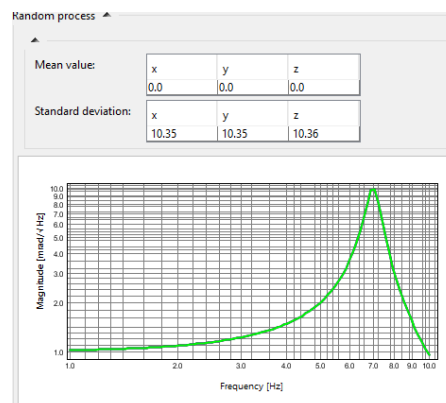


Figure 8-21: Random process contribution with default (“high-res”) grid

8.6 Example 6: Script-Based Execution

This example covers the evaluation of an existing scenario controlled by a MATLAB script and using the functions available for script-based execution (see chapter 7). This is always possible after a `.peet` scenario has been created (configured and saved) once via the GUI. The script in the example is used to:

- introduce the use of MATLAB variables defined as block parameters
- evaluate the scenario for all defined requirement sets with different parameterizations of block settings

- enable a PES-To-PEC contribution analysis (see chapter 6.4.4.3) and add the results to a report
- generates an Excel report for the results of each run with a user-defined suffix for the report file

8.6.1 Description

This scenario for this setup is mainly equivalent to the one for Example 1 (chapter 8.1). It contains the same statistical requirement sets, i.e.:

- "Ensemble" statistical interpretation applies with a 3-Sigma level of confidence
- "Temporal" statistical interpretation applies with a 2-Sigma level of confidence
- "Mixed" statistical interpretation applies with a 1-Sigma level of confidence


In this case however, the scenario shall be evaluated for two different sets of values of each PES' ensemble parameters of their temporal distributions.

| PES | RV distribution | Distributed Parameter | Parameter values [mrad] |
|-----|-------------------------------|--|---|
| 1 | Uniform: $U(-C, C)$ | Range C: Bimodal $BM(C_{min}, C_{max})$ | C_{min} : [0 2 4] C_{max} : [3 6 10] |
| 2 | Gaussian: $G(\mu, \sigma)$ | Standard deviation σ : Uniform $U(\sigma_{min}, \sigma_{max})$ | μ : [0 0 0] σ_{min} : [1 1 2] σ_{max} : [2 3 3] |

Using MATLAB variables for that purpose allows changing these values externally without any changes in the block dialogs in general. Further, evaluating the scenario repeatedly with different values can be controlled solely from a MATLAB script without using the GUI at all.

Either inspect and run (with breakpoints if intermediate results are of interest) the script file `Example6_scriptbased.m` in the `examples\Examples6.peet` folder directly or create it from scratch following the step-by-step description in the next subchapter.

8.6.2 Setup in PEET

- 1) Load `Example1.peet` from the `\examples` subfolder of the PEET installation.
- 2) Double-click on "PES 1" and check the MATLAB icon () in the **Range** panel. The table inputs for the **Minimum** and **Maximum** values are replaced by single input fields. Enter `minAmplitudes` and `maxAmplitudes` in the respective fields. These are the (arbitrary) variable names which are used to assign the values from a MATLAB script which is created in later steps. Confirm the changes with the **OK** button.

- 3) Double-click on "PES 2" and check the MATLAB icon in the **Standard deviation** panel. Enter `minSigma` and `maxSigma` as variable name for the **Minimum** and **Maximum** and confirm the changes with the **OK** button.
- 4) Save the scenario as `Example6.peet` to an arbitrary location.

The scenario is now fully configured and GUI is no longer required for the script-based execution.

- 5) Close the PEET GUI.
- 6) Open or create a new MATLAB `.m`-file and save it to the `Example6.peet` folder as `Example6_scriptbased.m`
- 7) Define the parameters to be modified in the scenario according to the table values in the previous chapter. For instance, add the following code snippet to the `.m`-file:

```
%% Define the parameters to be varied
% Parameters for PES 1
minAmplitudeData = [0 0 0      % 1st set
                   1 2 3];    % 2nd set
maxAmplitudeData = [1 2 3      % 1st set
                   4 5 6];    % 2nd set

% Parameters for PES 2
minSigmaData = [1 1 2          % 1st set
                2 2 3];        % 2nd set
maxSigmaData = [2 2 3          % 1st set
                3 3 4];        % 2nd set
```

- 8) The next step is to load the `.peet` scenario as analysis object into a MATLAB workspace variable (`example6` is used here, but the name is arbitrary in general). Assuming that the script file is located as mentioned in step 5, this is realized using:

```
example6 = engine.Analysis(pwd);
```

- 9) The variation of the parameter values is realized using a loop over with loop index `idxPar`:

```
for idxPar = 1:size(minAmplitudeData,1)
    .
    .
    .
end
```

Note: All following snippets need to be placed within this loop.

- 10) Now the parameter values for the current loop are assigned to the MATLAB variable names which were specified in the block dialogs of the PEET GUI (steps 2 and 3):

```
minAmplitudes = minAmplitudeData(idxPar, :);
maxAmplitudes = maxAmplitudeData(idxPar, :);
minSigma      = minSigmaData(idxPar, :);
maxSigma      = maxSigmaData(idxPar, :);
```

- 11) In a next step, the scenario can be initialized using the following command:

```
disp(['Initializing scenario for parameter set ',
num2str(idxPar), '...'])
example6.initialiseSystem;
```

Tip: An initialization is only necessary when any parameter values are modified. It is not necessary to re-initialize before evaluating different requirement sets (with fixed parameter values).

- 12) To enable the PES-to-PEC contribution analysis (6.4.4.3), the respective setter function is used to set the first flag to true:

```
example6.setPecPercentageFlags([true, false]);
```

- 13) To configure the report such that it includes the result of this analysis, another setter function is used. As the function expects a structure as input, the latter is generated first:

```
flags.PEC_Percentages = true;
example6.setReportSheetFlags(flags);
```

Note: The previous two steps must be executed after the initialization as otherwise the scenario settings are restored.

- 14) Three different requirement sets ("APE Ensemble", "APE Temporal" and "APE Mixed") are defined in the scenario. The data of these sets are extracted and saved in a variable `reqSetData`:

```
reqSetData = example6.getScenarioSettings('reqSet');
```

- 15) To compute all requirement sets sequentially, it is necessary to introduce another loop over these sets (loop index `idxReq` in this example). All further snippets in the next step are placed in this inner loop.:

```
for idxReq = 1:length(reqSetData)
    .
    .
```

```
.  
end
```

- 16) Next, the name of the current requirement needs to be extracted from `reqSetData`. (the second cell entry corresponds to the name of the requirement set). Then this requirement is made active for the evaluation:

```
reqSetName = reqSetData{idxReq}{2};  
  
disp(['Computing budget "', reqSetName, ...  
    '" for parameter set ', num2str(idxPar), '...'])  
example6.setActiveIndex(reqSetName);
```

- 17) Now the actual evaluation can be started:

```
example6.computeBudget;
```

- 18) After the evaluation is completed, a report can be generated. Before doing so, a suffix with a run counter is appended to the nominal report name:

```
suffix = ['_run_', num2str(idxPar)];  
example6.setReportSuffix(suffix);
```

- 19) Finally, the report generation is triggered as last element of the loop:

```
disp(['Creating report of "', reqSetName, ...  
    '" for parameter set ', num2str(idxPar), '...'])  
example6.generateReport(reqSetName);
```

Now the file can be saved and executed. After the batch operation is completed, the report files are located after evaluation in the `\report` folder of the scenario

8.7 Example 7: User-Defined Post-Processing

This example covers the definition and evaluation of a user-defined post-processing analysis (see chapter 6.4.4.1.5) applied to the data of an error contribution block. It uses a preconfigured scenario to show:

- how to link a MATLAB initialization script to a scenario which is always executed before the scenario is initialized
- how to set up a post-processing analysis in the GUI
- how the basic configuration structure of the analysis must be defined
- how the user-defined analysis function can be used to display different types of results in the GUI and to include them in Excel reports

8.7.1 Description

This setup for this scenario is intentionally simple. `Example7.peet` (in the `\examples` subfolder of the PEET installation) contains only one *PES (Time-Random)* block and a *Total Error* block to which the post-processing analysis shall be applied. A single statistical requirement set “APE Temporal” is defined representing a temporal statistical interpretation with a 95% level of confidence.

The PES parameters are the following (such that the resulting distribution at the *Total Error* block is directly given by the one specified for the error source:

| PES | RV distribution | Distributed Parameter | Parameter values [°] |
|-----|-------------------------------|-----------------------|---------------------------------------|
| 1 | Gaussian: $G(\mu, \sigma)$ | None | μ : [1 2 3] σ : [3 2 1] |

Further, the scenario folder contains two preconfigured m-files:

- an initialization script `example7_initScript.m` which contains the configuration structure for the analysis and the definition of the parameter to be used
- a “user-defined” post-processing function `postProcFunctionForExample7.m` which contains the algorithms to be applied to the PEC results and the definition of the outputs to be displayed in the GUI and included in a report

8.7.2 Setup in PEET

The main purpose of this example is to show how the configuration of the analysis outputs defined in the function `postProcFunctionForExample7.m` can be used to display different kinds of information in the GUI and in reports. The scenario `Example7.peet` is ready to run and does not require any further configuration steps before execution.

However, the main steps to define the post-processing analysis are recalled below. First to enable the analysis for the *Total Error* block:

- 1) Open the **Analysis Features** → **Post-Processing** menu.
- 2) As there is only one error contribution block present, the *Total Error* tab is already selected. Use **Add Script** and choose **User-Defined** from the **Select script type** dialog and confirm with **OK**.
- 3) Provide the variable name for the **Configuration Matlab-Struct** (i.e. `example7_config` in this example) and confirm with **OK**.

To ensure that the structure is always present in the MATLAB workspace when required, it is useful to store in a script (`example7_initScript.m` in this example) which is automatically called by PEET. To do so:

- 1) Open the **Setup** → **Evaluation Settings** menu.
- 2) In the **Matlab initialization script** panel, either directly enter the path to `example7_initScript.m` or use the **...** button to browse to the file location

(check **Copy script into local folder** in this case to ensure a file path relative to the scenario is used) and confirm with **OK**.

The scenario is configured such that the post-processing analysis results are included in any generated report and that reports are also automatically generated when evaluating the scenario. To do so:

- 1) Open the **File** → **Create Report...** menu.
- 2) In the **Results** panel, enable the **Post-processing** checkbox.
- 3) In the **Results generation** panel, enable the **Always create report after evaluation** option and confirm with **OK**.

Now the scenario can be evaluated (▶-icon). When completed, inspect the **Breakdown Tree View** (**Post-Processing Example** tab) and the report in the `\report` subfolder of the scenario. Further inspect the files `example7_initScript.m` and `postProcFunctionForExample7.m` itself with the comments present to track how the results are created and how the various options modify the layout.

8.8 PointingSat Scenario

This example describes an artificial precision pointing satellite mission “PointingSat” with typical pointing error sources and system transfers. It serves as application example of a of the methodology described in the ESA Pointing Error Engineering Handbook ([AD1],[RD16]) covering all steps from error source definition, transfer analysis, error index contribution analysis and budget evaluation.

`Example8.peet` in the `\examples` subfolder of the PEET installation represents the study case scenario with a complete description given in [RD12] – which is located in the scenario folder as well.

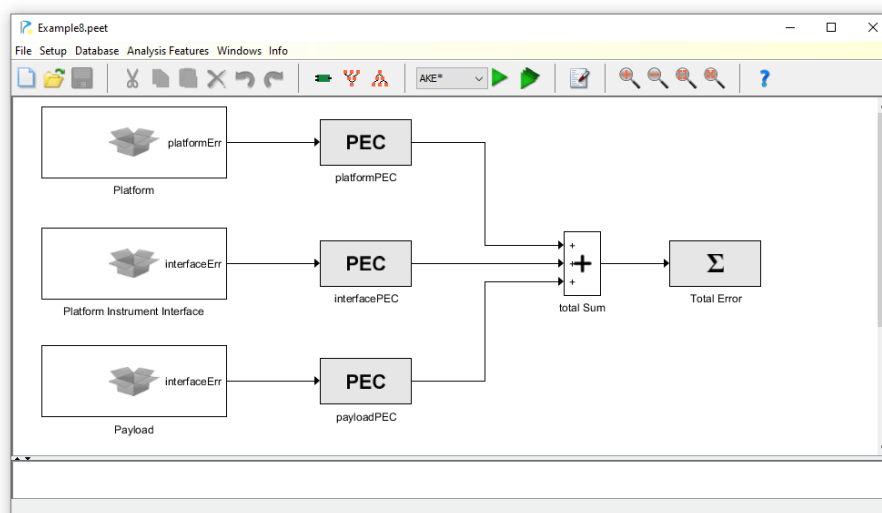


Figure 8-22: PointingSat Scenario (Top-Level)

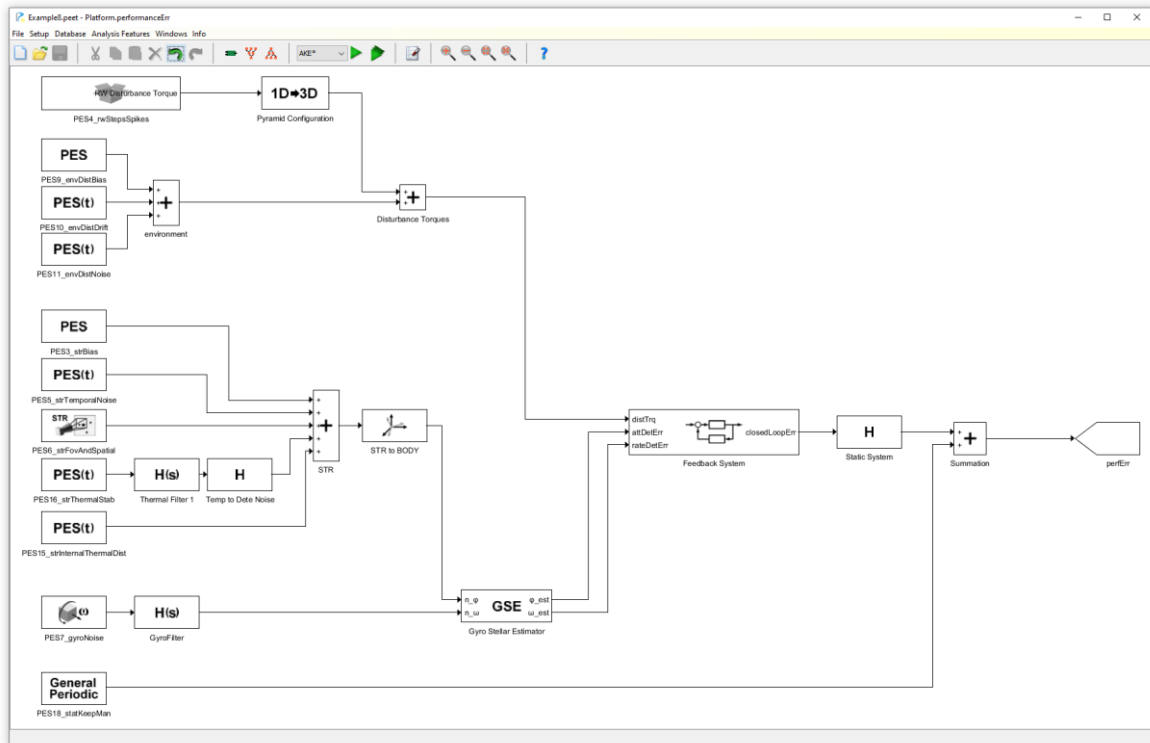


Figure 8-23: PointingSat Scenario (exemplary subsystem)

9 Hints and Guidelines

This chapter provides a collection of advices concerning the use of the tool and the definition and setup of scenarios.

9.1 Software Limitations

PEET needs to be considered as a systems engineering tool with a focus on steady-state solutions rather than a replacement for a non-linear simulator. Functionalities concerning the treatment of non-linearities, transients*, system uncertainties and non-stationary random processes** cannot be explicitly accounted for in the tool. There is a lack of standards for these topics which are partially at the interface of research and tool development where dedicated algorithms and methods are not mature enough.

Workarounds that allow a treatment of such topics in PEET were assessed in case studies (e.g. [RD12], [RD13]) .

Furthermore, although the tool takes over the entire computation chain of an error budget, still a certain amount of user expertise on pointing error engineering is required. In particular, for a proper definition of requirements or modelling of error sources, no universal guidelines or automated rules can be provided. Thus, a dialogue between payload scientists, pointing experts and AOCs simulator experts is usually necessary.

Finally, PEET is not foreseen as a tool for control/estimator design & tuning, as e.g. the stability of closed-loops in a *Feedback System* is not checked. The software only covers the perimeter of the PEEH Handbook and provides interfaces such that each user can plug PEET into his/her own tools for design/tuning.

* The *General Periodic Error* block can be used as a workaround for a selection of 'transient' signals (exponential decay, decaying cosine, rectangular signals) under the assumption that these signals occur periodically.

** The introduction of an ensemble parameter affecting the shape of a PSD (*PES (Time-Random)* block) could be used as a workaround under the assumption that the non-stationary process can be represented by an ensemble of PSDs which are each stationary itself.

9.2 Software Accuracy

Due to the numerical approach used for budgets evaluation (i.e. sample-based PDF determination and numerical integration to obtain the CDF, see [RD7]), all results are expected to have a numerical error in the order of one percent with respect to the exact solution. This holds for level of confidence values up to "3 Sigma" and was verified with simplified examples where analytical solutions for the PDF/CDF exist (maximum relative error for a 1-Sigma value of 1.8% in a single test, otherwise well below 1%). For higher levels of confidence, this accuracy cannot be guaranteed - especially for heavily tailed distributions where only a small number of samples falls into the "far" bins (tests with respective distributions show relative errors <5% for values up to "4 Sigma").

The following settings and models further have an impact on the accuracy of the results, especially when they are related to driving contributions to a budget:

- Choosing the **Low-Resolution** mode in the **Setup** → **Evaluation Settings** leads to a faster evaluation but less accuracy as an order of magnitude less samples are used to represent the PDF for each axis of a signal.
- The frequency grid density specified in the **Setup** → **Evaluation Settings** menu has an impact on the accuracy as it determines the accuracy of the numerical integration of random process spectra to obtain their variance. While the nominal grid settings are expected to be sufficient in most cases, this might not be the case if spectra with large peaks/features have to be modelled. Please refer to chapter 8.5 for an example of this effect.
- The frequency domain models used to represent drift signals and the general periodic error source models generally have larger inherent errors w.r.t. to the corresponding exact signal due the Fourier series approximation used for their representation, see examples below for a drift signal and exponentially decaying cosine:

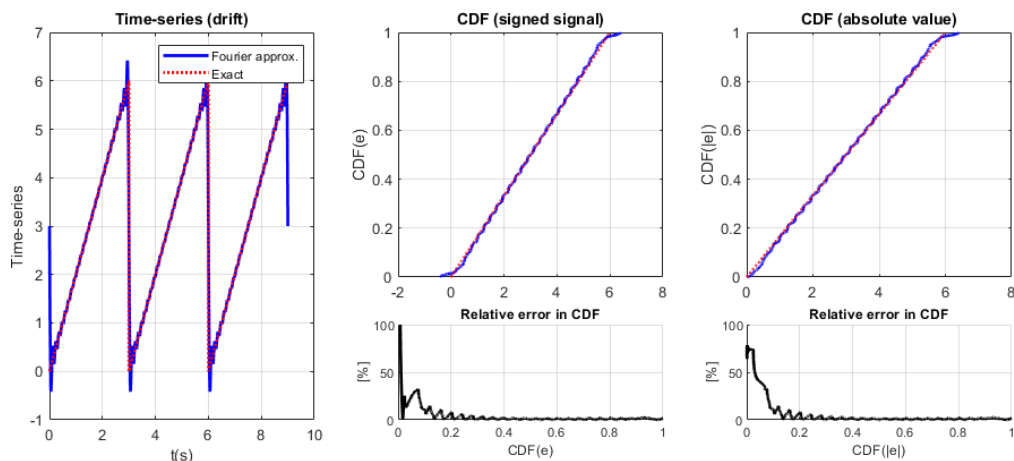


Figure 9-1: Time-series, CDF and relative error for an ideal drift signal and the Fourier series approximation implemented in PEET

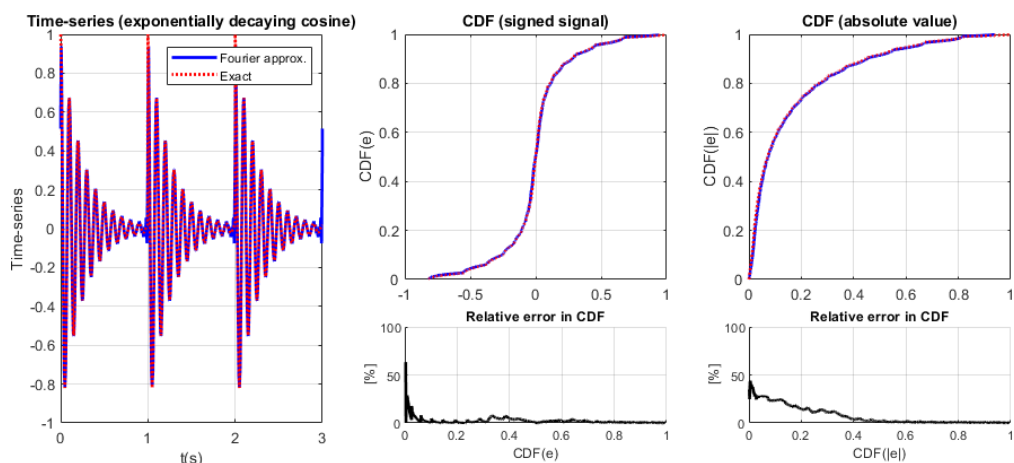


Figure 9-2: Time-series, CDF and relative error for an exponentially decaying cosine signal and the Fourier series approximation implemented in PEET

Tip: The helper function `checkGeneralPeriodicApproximation` can be used to compare the signal approximation with the exact signal (see chapter 7.10).

Although the relative error in the level of confidence region usually of interest (between “1-Sigma” and “3-Sigma”, i.e. CDF values between 0.68 and 0.99) is also in the range of 1%, it largely deviates for smaller level of confidence regions. However, the more crucial point is the worst-case value which is determined for these signals: Fourier series approximations suffer from oscillations at discontinuities (“Gibbs phenomenon”) which range up to 18% of the jump size, thus also worst-case values which result for instance for ensemble statistical interpretation are over-estimated by this amount.

Despite this disadvantage, the Fourier series approach is still considered valuable compared to an even more simplified random variable approach (see e.g. [RD3] for drift signals), as the benefits outweigh – i.e. frequency domain metrics can accurately be applied, the output of a dynamic system transfer can accurately be predicted and signals with different frequency ranges can accurately be summed and evaluated.

9.3 Basic Sensor/Actuator Models

While the PEET model database contains parametric error models for several sensors and actuators, other typical devices are not explicitly covered (e.g. CESS, magnetometers, magnetic torquers, sun sensors) due to a lack of generalized models. However, basic models for bias and noise contribution can always be realized using the standard PES block, e.g.:

- Bias:
PES (Time-Constant) block with or without a distributed value.
- Noise:
PES (Time-Random) block using a **Random Process** description of type **BLWN** which requires a noise standard deviation and sampling time as input.

Such simple models (or more complex, regularly used ones) could then be placed in a *Container* and saved as a user-defined model to the database (see chapter 6.4.3).

9.4 Periodic Signals and Evaluation Time Frame

The treatment of periodic signals in PEET follows the evaluation rules in [AD1] and [AD2]. This implies that periodic signals are always assumed to cover an integer number of cycles to achieve the related statistical moments and distributions.

In general, special care must be taken when specifying periodic error sources which have longer periods than the evaluation time frame relevant for the requirement. This situation is depicted in Figure 9-3, where an error signal is periodic over one day while the observation time frame of interest is much shorter.

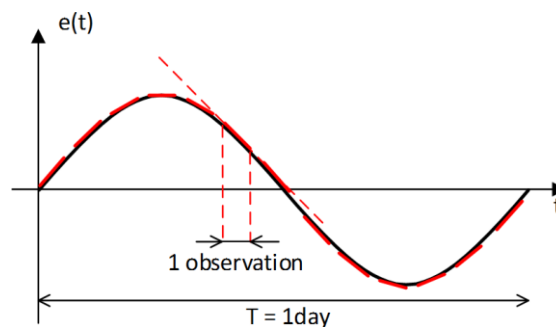


Figure 9-3: Periodic signal appears as drift + bias in observation time frame

In this specific case, the error signal does not appear at all as a periodic disturbance (but as almost as linear) in each observation. Consequently, the result obtained from a periodic error source would not reflect the real situation.

As an alternative description in this case, a drift signal plus a bias could be used. To account for the variation from observation to observation (i.e. the ensemble of all observations), both the drift rate and bias would need to follow an arcsine distribution and need to be defined as uncorrelated (as there is a 90° phase shift between maximum bias and maximum drift rate).

Note that this topic is not a “workaround” due to limitations of the software or the methodology described in the ADs, but a question of properly modelling the error source for a given requirement. Loosely speaking, not the physical periodicity of the error signal itself is crucial, but the periodic nature of an error source within the time frame relevant for the requirement.

To detect such “mismatches” between evaluation time frame and periodic signal frequencies, the [Check Frequency Bandwidth...](#) menu (6.4.2.7) can be used. If frequencies below the specified evaluation period (green dashed line) or outside the evaluation bandwidth are present, a remodelling might be suitable.

9.5 Ensemble Domains

As described in chapters 6.4.2.1 and 6.4.2.2, multiple ensemble domains can be defined (and assigned to the error sources) in a scenario and their statistical treatment can be further configured. The definition of ensemble domains is completely optional, however their presence allows a more flexible definition of requirements and evaluation of a budget.

The following examples shall highlight the different options available when using multiple ensemble domains. These examples use two different domains e_1 (“Misalignments”) and e_2 (“Observations”), but the statements are also valid for a larger number. The involved settings can be configured in the [Scenario Definition](#) menu.

- **Common** level of confidence evaluation

This option can be used for requirements of the type “The pointing error shall be smaller than X for 99.7% of all satellites and observations.”

Concerning the evaluation of the total error, this option is basically equivalent to not using any specific ensemble domain at all (The pointing error shall be smaller than X with 99.7% level of confidence”). The total error is evaluated jointly from the PDF of the

sum of the contributions from all domains, i.e. the contributions from all different domains are still evaluated statistically as depicted in Figure 9-4.

However, it allows assessing a separate budget for the contributions from each domain in a single scenario and ensures that no “meaningless” correlation can be specified between contributors in different domains.

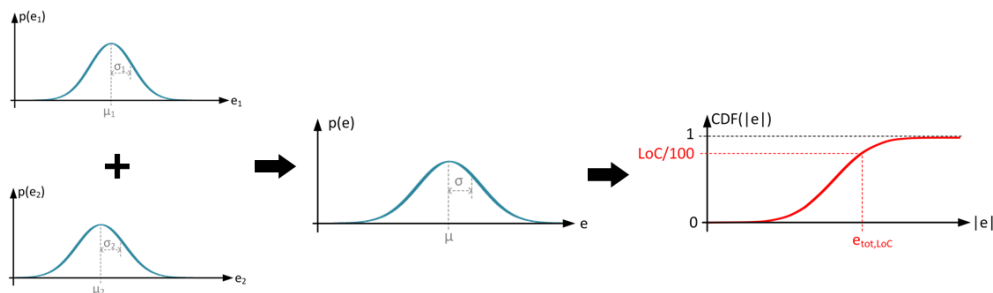


Figure 9-4: Common level of confidence evaluation

Furthermore, the *Statistical treatment* can be selected individually for each domain. This option accounts for requirements where certain contributions should be considered as worst-case (e.g. the satellite with worst-case alignment) while for other contributions a statistical consideration is sufficient (e.g. 90% of all observations).

Without an additional ensemble domain, the entire budget would have to be evaluated either statistically or as worst-case for all error sources and no distinction would be possible.

- **Individual** level of confidence evaluation

This option allows assigning a level of confidence value individually to each domain. It can be used when an explicit requirement is defined for different contributions (e.g. “The worst-case pointing error shall be smaller than X for 99.7% of all satellites (i.e. alignments) and 68% of all observations.” In this case, the total error is computed from two discrete value with the levels of confidence given for the individual domains (see Figure 9-5).

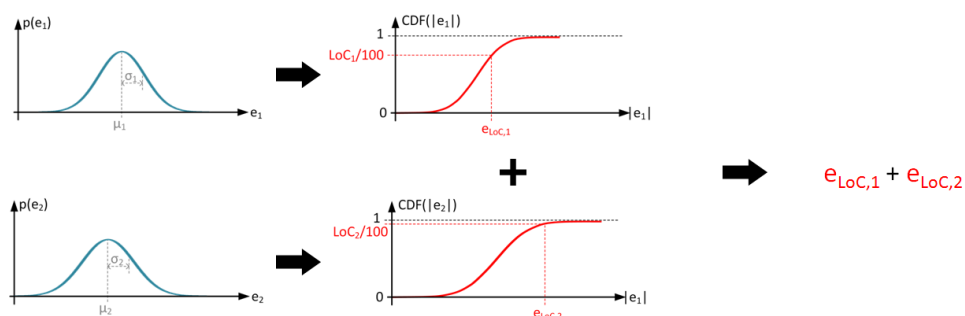


Figure 9-5: Individual level of confidence evaluation

Note that in case of an identical level of confidence value for both domains, the total error is intentionally different compared to the **Common** evaluation (as the acceptable “worst-case” values from the two domains are summed), and provides more

conservative results. Thus, the evaluation to be used needs to be clearly formulated in the related requirement.

9.6 Worst-Case Values of Distributions and Worst-Case Distributions

Dependent on the user-defined statistical treatment, worst-case values of certain PDFs (ensemble interpretation) or worst-case distributions (temporal interpretation) have to be determined which are related to the distribution bounds.

The minimum and maximum values of all available distributions are summarized in the following Table where conventions are introduced for unbounded distributions (any restrictions on the parameter values itself are detailed in the Block Settings paragraph of chapter 10.16).

Table 3: Minimum and maximum values of basic distributions

| Distribution | PDF parameters | e_{min} | e_{max} | Remarks |
|-------------------|---|---------------------------|---------------------------|--|
| Delta | $\delta(\mu_D)$ | μ_D | μ_D | discrete value |
| Uniform | $U(x_{min}, x_{max})$ | x_{min} | x_{max} | |
| Bimodal (Arcsine) | $BM(x_{min}, x_{max})$ | x_{min} | x_{max} | |
| Gaussian* | $G(\mu, \sigma)$ | $\mu - 3\sigma$ | $\mu + 3\sigma$ | equivalent to 0.27% and 99.73% LoC as unbounded |
| Rayleigh | $R(r, \sigma_r)$ | r | $r + 3.4393 \sigma_r$ | equivalent to 99.73% LoC as unbounded |
| Trunc. Gaussian | Both bounds $G_T(\mu, \sigma, LTB, UTB)$ | LTB | UTB | |
| | Sym. bound $G_T(\mu, \sigma, STB)$ | $\mu - STB$ | $\mu + STB$ | |
| | Upper bound $G_T(\mu, \sigma, UTB)$ | $min(\mu - 3\sigma, UTB)$ | UTB | Min. equivalent to unbounded 0.27% LoC (or UTB if smaller) |
| | Lower bound $G_T(\mu, \sigma, LTB)$ | LTB | $max(\mu + 3\sigma, LTB)$ | Max. equivalent to unbounded 99.73% LoC (or LTB if larger) |
| Beta | $Beta(\alpha, \beta, d, s)$ | d | $d + s$ | |
| User-Defined | tabular | $min(e)$ | $max(e)$ | numerical definition |

***Note:** The definition of the min/max value of a Gaussian distribution is a convention selected to be in line with the Table B-4 of [AD1]. If any other bound different from “3 σ ” needs to be applied, then a Truncated Gaussian with a symmetric bound can be used instead.

From a general perspective, the worst-case value of each of these distributions is the bound with the maximum deviation from zero, i.e. $WC = \max(|e_{min}|, |e_{max}|)$ times the corresponding sign of the entry with the largest absolute value.

At PES level, these worst-case values may have (intentionally) different signs for different PES. Further, the signs of such initial worst-case values can be inverted during the transfer analysis steps. Thus, a summation of individual worst-case values (determined a priori) does not necessarily lead to the overall worst-case for the final error.

However, for all PDFs related to ensemble parameters of time-random variables, these worst-case values are already required at PES level as they define the properties of the temporal PDF to be realized with full resolution (see tables in chapter 6 of [RD7]).

Tip: The determination of worst-case values in the context of statistical interpretation also for non-random variable error sources is described in [RD7] which is available in the electronic manual of PEET (chapter 5.1).

Thus, conventions are introduced on how the worst-cases are selected. These conventions give the user a clear indication on how error sources need to be set up and provides control on how they are treated in the evaluation.

For temporal statistical interpretation:

Convention: The worst-case time-random variable PDF corresponds to the ‘widest’ distribution of all possible ensemble realizations.

This convention ensures that at least the extreme cases of the distribution bounds are covered by the evaluation and assumes that the PES are defined such that a more positive value worsens a budget. The particular temporal worst-case PDFs for the available time-random variable options are summarized in Table 4. ‘Workarounds’ to specify a different worst-case distribution are provided as examples below the table.

Table 4: Convention for time-random variable worst-case distributions

| Temporal RV | | Ensemble parameter | Worst-case PDF $p_T(e)$ |
|-------------|-------------|--------------------|------------------------------|
| Uniform | | None | $U(LB, UB)$ |
| | | Lower bound | $U(LB_{min}, UB)$ |
| | | Upper bound | $U(LB, UB_{max})$ |
| | | Symmetric bound | $U(-SB_{max}, SB_{max})$ |
| Gaussian | | None | $G(\mu, \sigma)$ |
| | | Mean value | $G(\mu, \sigma)$ |
| | | Standard deviation | $G(\mu, \sigma_{max})$ |
| | Both bounds | None | $G_T(\mu, \sigma, LTB, UTB)$ |

| | | | |
|--------------------|-------------|--------------------|-------------------------------|
| Truncated Gaussian | Sym. bound | None | $G_T(\mu, \sigma, STB)$ |
| | | Mean value | $G_T(\mu, \sigma, STB)$ |
| | | Standard deviation | $G_T(\mu, \sigma_{max}, STB)$ |
| | Upper bound | None | $G_T(\mu, \sigma, LTB, UTB)$ |
| | Lower bound | None | $G_T(\mu, \sigma, LTB, UTB)$ |

Example:

For a temporal uniform distribution with a uniformly distributed upper bound $p_{T,E}(e) = U(-3, U(-2,1))$, the worst-case (temporal) distribution *by convention* is $p_T(e) = U(-3,1)$.

To consider explicitly the 'narrower' distribution $p_T(e) = U(-3,-2)$ as worst-case distribution, the ensemble distribution of the upper bound needs to be replaced by the explicit discrete value of -2.

$U(-3,-2)$ would actually lead to a larger overall error contribution when evaluating $p_T(|e|) = U(2,3)$ alone. However, this is not necessarily the case if other (positive) distributions are present and summed which is why the convention of the 'widest' distribution is applied by default.

For **ensemble** statistical interpretation:

Convention: The PDF of the time-random variable worst-case values corresponds to the distribution of the maximum values obtained from all possible ensemble realizations (rather than the maximum absolute value).

This convention essentially leads to the PDF of the upper bound values of the distributions and assumes that the PES are defined such that a more positive value worsens a budget. The particular worst-case value PDFs for the available time-random variable options are summarized in Table 5. 'Workarounds' to specify a different worst-case distribution are provided as examples below the table.

Table 5: Convention for worst-case value PDFs of time-random variables

| Temporal RV | Ensemble parameter | Worst-case value PDF $p_E(e)$ | |
|-------------|--------------------|-------------------------------|---------------|
| Uniform | None | $\delta(UB)$ | |
| | Lower bound | $p_E(UB)$ | |
| | Upper bound | $p_E(UB)$ | |
| | Symmetric bound | $p_E(SB)$ | |
| Gaussian | None | $\delta(\mu+3\sigma)$ | |
| | Mean value | $p_E(\mu+3\sigma)$ | |
| | Standard deviation | $p_E(\mu+3\sigma)$ | |
| | Both bounds | None | $\delta(UTB)$ |

| | | | |
|--------------------|-------------|--------------------|------------------------------------|
| Truncated Gaussian | Sym. bound | None | $\delta(\mu + STB)$ |
| | | Mean value | $p_E(\mu + STB)$ |
| | | Standard deviation | $\delta(\mu + STB)$ |
| | Upper bound | None | $\delta(UTB)$ |
| | Lower bound | None | $\delta(\max(\mu + 3\sigma, LTB))$ |

Example:

For a temporal uniform distribution $p_T(e) = U(-3, 1)$, the worst-case value *by convention* is 1.

To obtain explicitly a worst-case value of -3 for ensemble SI, the error source needs to be mirrored, i.e. defined as $p_T(e) = U(-1, 3)$ and fed through a static system with a gain of -1 in the system transfer. If all other sources are negative as well, this indeed gives the overall worst-case.

Example:

For a temporal uniform distribution with a uniformly distributed upper bound $p_{T,E}(e) = U(-3, U(-2, 1))$, the (ensemble) distribution of the worst-case values *by convention* is $p_E(e) = U(-2, 1)$.

To consider explicitly the discrete worst-case value of -3, the distribution can again be mirrored, i.e. $p_{T,E}(e) = U(U(-1, 2), 3)$ and multiplied with a gain of -1 in the system transfer.

9.7 Remark on Temporal Statistical Interpretation

The implemented approach for the temporal statistical interpretation describes the temporal signal of the realization which contains the overall worst-case value (in time) of all ensemble realizations (see plot in the middle of the ECSS figure below for illustration):

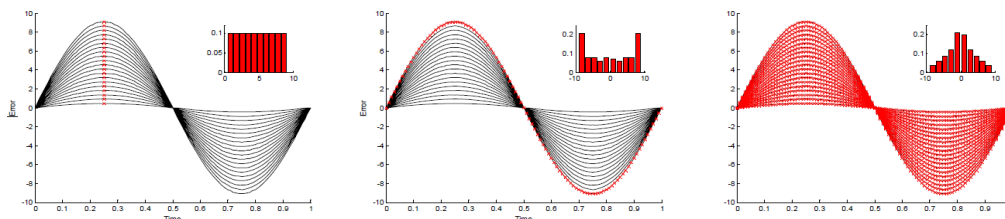


Figure 9-6: Illustration of ensemble (left), temporal (center) and mixed (right) statistical interpretation [AD1]

With this approach, the evaluation steps are as follows:

- Determine the worst-case value of all ensemble parameter realizations (for random variables, this can be done a priori per error source using the analytical worst-case of the ensemble distribution as shown in the previous chapter 9.6).
- Realize the temporal distributions **for the worst-case parameters** numerically via samples.

- Compute the CDF numerically and evaluate the error value for the specified level of confidence as result of the temporal interpretation.

This approach is line with the temporal SI formulation (“*variation over time for the worst-case member of the statistical ensemble. In this case, the mean and variance used in the budget relates to the variation over time, and can be derived analytically if the time variation is known*”) in Appendix B of the ECSS standard [AD1], which is also applied to the budget contribution tables for the random variable description of the different error classes therein.

However, note that the above step “*realize the temporal distributions for the worst-case parameters*” deviates slightly from the control ECSS definition of the worst-case member. In the above step, the worst case is selected for 100% confidence level over time whereas in the ECSS definition refers to the worst-case parameters for the specified confidence level over time (which ensures that the requirement is satisfied for all members of the ensemble). This deviation from the control ECSS definition is in general acceptable, but it has to be confirmed on a case-by-case basis.

Considering the temporal SI formulation in Appendix A of [AD1] - “*I is less than I_{max} for a fraction P_C of the time (i.e. this holds for any member of the ensemble)*” - in particular the term in parentheses provides a more general approach which is foreseen to be included additionally in an update of the ESA Handbook [AD2]. This approach is based on the following steps:

- Realize the temporal distributions **of all ensemble parameter realizations** (numerically via samples or simulation).
- Compute the CDF of each realization numerically and evaluate the error value for the specified level of confidence.
- Determine the worst-case error value of all evaluations as result of the temporal interpretation.

In other words: while the implemented approach first determines a single worst-case time series by evaluating the 100% level of confidence value over time (of all realizations) and then applies the requirement confidence level to this time-series, the general approach directly applies the requirement confidence level to the time-series of each ensemble realization and then determines the effective value from the ensemble of worst-case values.

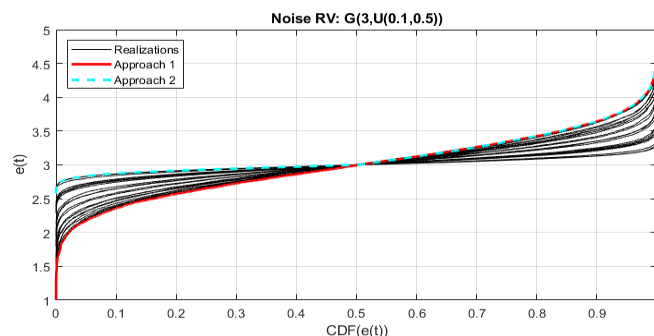


Figure 9-7: Exemplary results obtained with the different Temporal SI approaches

The different results obtained with the implemented approach (red) and the alternative approach (blue) are depicted in the figure above exemplary for a temporal Gaussian

distribution with a standard deviation varying (uniformly) in each realization. The individual realizations of all ensembles are indicated by the set of black lines.

For this specific example, both approaches lead to the same result only for CDF values larger than 0.5 (or 50% level of confidence specified for the temporal statistical interpretation). However, below this value, considering the CDF of the realization with the overall worst-case value (1st approach) results in a smaller budget value than the second approach.

This also holds in general, i.e. the 2nd approach always leads to a more conservative result. While the latter is well suited for Monte-Carlo simulations where all required numerical data for all realizations is generated anyway, it is not applicable for a) a semi-analytical approach as in PEET for reasons of memory consumption ($10^6 \times 10^6$ samples per source/axis to represent the distributions of all realizations) and performance (10^6 CDF determination steps from 10^6 samples each) or b) approximate table-based approaches as in Appendix B of [AD1].

Concluding, for a worst-case value determined for a 100% confidence level over time, the implemented approach is fully aligned with the general ECSS definition. For worst-cases determined from smaller confidence levels, results can deviate slightly – which is general acceptable, but might need to be checked for a specific use case.

9.8 Sensitivity Analysis

The sensitivity of (final or intermediate) budget results to a certain error source or transfer system parameter can easily be assessed exploiting the script-based execution of PEET (see also chapter 7).

The parameter p for which the sensitivity is of interest, needs to be set up as MATLAB variable (🚩-icon in the respective dialog). In the first computation run, the nominal value for p is assigned to the variable, and the “target” results of interest are saved. Then, a “perturbed” value for p is assigned to the variable instead, the computation is run a second time and the “target” results are saved again.

Then the sensitivity s can be computed from:

$$s = \frac{[\text{Result}]_{\text{pert}} - [\text{Result}]_{\text{nominal}}}{P_{\text{pert}} - P_{\text{nominal}}}$$

The perturbed parameter should be chosen according to the following relations:

$$P_{\text{pert}} = \begin{cases} P_{\text{nominal}} + \delta & \text{if } |p| \leq 1 \\ P_{\text{nominal}} \cdot (1 + \delta) & \text{if } |p| > 1 \end{cases}$$

Note: Theoretically, the perturbation δ can be arbitrarily small. However, due to the numerical approach used for the evaluation of statistical requirements, a variation in the range of 5% should be realized, especially when the parameter represents a distribution parameter of an error source.

Example:

Assume the sensitivity analysis is carried out for a generic PES with output unit [m] and the total error is computed in [mrad]. The analysis gives a result of [10.0 0.0 -1.5] for the x-, y- and z-axis. In this case, it can be observed that:

1. The parameter only has an effect on the x and z component of the total error.
2. Comparing all three axes, the x-axis budget is most sensitive to changes of the parameter.
3. The unit of the sensitivity is [mrad/m].
4. Increasing the signal increases the total error on the x-axis.
5. Increasing the signal decreases the total error on the z-axis.

10 Database Blocks

This chapter gives a detailed description of the parameters which are necessary to set up the various blocks defined in the PEET model database. Where applicable, also the mathematical background of the models is provided. After a description of parameters common for most blocks, the blocks are listed in alphabetical order.

10.1 Common Block Settings

Several panels and configuration settings are common to most of the database blocks. These settings are introduced in this section and not explicitly repeated in the description of the individual models.

Signal/System dimension

Various blocks support the selection of the signal or system dimension. In this case either **1D** or **3D** can be selected from the respective drop-down list.


Output unit

Generic blocks (*PES*, *Static System*, *Dynamic System*) allow a free definition of the output unit. Other blocks are restricted to a certain class of units (e.g. time), however an output unit can still be chosen within the respective class. To do so, choose **Custom...** from the unit selection drop-down list (see chapter 6.1.3).


Inherit input unit / output unit from input

Certain blocks manipulate the incoming signal, but do not necessarily change the signals' unit (e.g. a *Coordinate Transformation*). In such case, by default, the output unit is inherited from the input signal. If this is not desired, the respective selection can be unchecked. This enables a unit selection drop-down list for the selection of a different compatible unit.

(User-description)

A left-click on the -icon opens dialog which allows an arbitrary user-defined description of the block. This might become useful in case many blocks are used in a scenario or a scenario is shared among different users. Placing the mouse over the icon shows the current description as a tooltip.

(Quick-Help)

Placing the mouse on the -icon gives a brief description for a certain setting of a block or the entire block itself.

Ensemble domain

In case multiple ensemble domains are defined in a scenario (*Setup* → *Ensemble Domains* menu, see chapter 6.4.2), it is also possible to assign a **pointing error source** to a domain directly in the block dialog. By default, this option is not available.

10.2 Accelerometer Noise

Output

The output of this block represents the noise spectrum of a linear accelerometer (in case the *Signal dimension* is **3D**, an identical spectrum is assumed for all axes). No acceleration bias is included in the model. Any unit compatible to $[\text{Length}/\text{Time}^2]$ can be defined as *Output unit*.

Block Settings

The following **Noise parameters** need to be specified in the respective panel:

Velocity random walk

The scalar coefficient N for the velocity random walk contribution to the power spectrum given in a unit compatible to [Acceleration/ $\sqrt{\text{Hz}}$] or equivalently [Velocity/ $\sqrt{\text{Time}}$].

Acceleration random walk

The scalar coefficient K for the acceleration random walk contribution to the power spectrum given in a unit compatible to [(Acceleration/Time)/ $\sqrt{\text{Hz}}$] or equivalently [Velocity/ $\sqrt{(\text{Time}^3)}$].

Bias instability

The scalar coefficient B for the bias instability to the power spectrum given in a unit compatible to [(Acceleration/ $\sqrt{\text{Time}})/\sqrt{\text{Hz}}$] or equivalently [Velocity/Time].

Quantization noise

The scalar quantization noise coefficient Q for the quantization noise contribution to the power spectrum given in a unit compatible to [(Velocity/ $\sqrt{\text{Time}})/\sqrt{\text{Hz}}$] or equivalently [Velocity].

Sampling period

The scalar sampling period T in a unit compatible to [Time] related to the quantization noise.

Background

The model for the acceleration noise spectrum is based on [RD1] with a more specific mapping of the parameters derived from [RD2]:

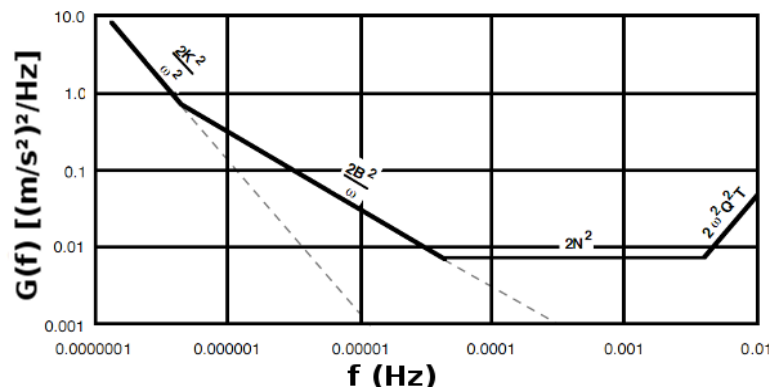


Figure 10-1: Acceleration noise PSD derived from [RD1] and [RD2]

10.3 Camera Range Noise

Output

The output of this block represents 3D bias and band-limited white noise contributions to the range measurement using a camera type sensor. Both bias and noise scale with the overall range to the target. Any unit compatible to [Length] can be defined as **Output unit**.

Block Settings

The following **Range error parameters** need to be specified in the respective panel:

Nominal range to target

Sets the scalar nominal working point distance between the sensor and the target in any [Length]-compatible unit.

Range bias

Sets the time-constant measurement bias(es) in [%] of the nominal range.

Range noise standard deviation

Sets the standard deviation(s) of the band-limited white noise in [%] of the nominal range.

Noise bandwidth

Sets the upper limit of the noise bandwidth in any [1/Time]-compatible unit.

Background

This model simply scales bias and the noise standard deviation with the given nominal range value. The flat noise spectrum is realized such that the given standard deviation is ideally realized within the bandwidth determined by the given sampling time.

10.4 Container

Container blocks are a special block type. They can be used to abstract a complex block structure into a single block symbol similar to a “Subsystem” in Simulink. A *Container* has no block mask, but a double-click open a new system editor window which can be populated with other blocks as on the main system editor level.

Note: No loops are allowed within a *Container* block. The *Feedback System* block is designed for that purpose.

The link to a higher level of the system editor is realized by the *Input Port* and *Output Port* blocks (their number within a *Container* is not restricted).

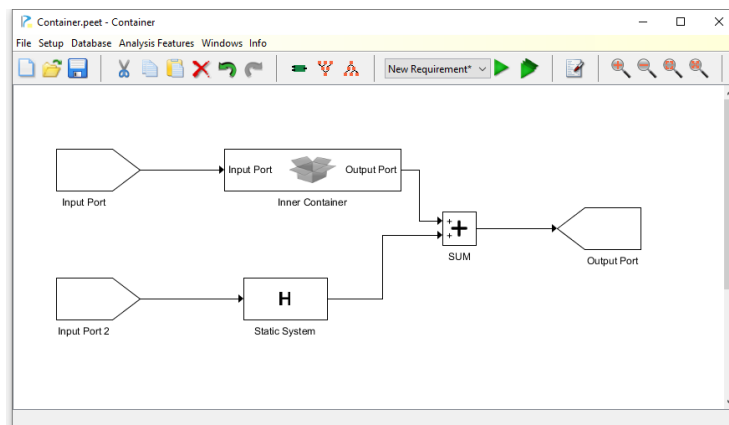


Figure 10-2: Exemplary content of a container block with two input ports and one output port

Figure 10-2 shows an example of a *Container* block with two inputs and one output. Nesting is also possible, i.e. further *Containers* can be placed inside a *Container* to represent lower system levels or to “tidy up” the system editor.

10.5 Coordinate Transformation

Input/Output

The output of this block represents the input signal expressed in a different coordinate frame following a user-defined conversion. By default, the unit of the output signal is inherited from the input. Only 3D input signals are supported.

Block Settings

The following settings need to be specified in the *Transformation settings* panel:

Rotation sequence

Defines the sequence used for the given Euler angles. The possible options cover both classical Euler sequences (1-2-1, 1-3-1, 2-1-2, 2-3-2, 3-1-3, 3-2-3) as well as Tait-Bryan sequences (1-2-3, 1-3-2, 2-1-3, 2-3-1, 3-1-2, 3-2-1). Note that "1", "2" and "3" are equivalent to the x-, y- and z-axis respectively

Euler angle unit

Defines the unit for the given Euler angles, where any [Angle]-compatible unit is supported.

Euler angles

Defines the values for the angles in the given unit for the first (*Phi*), second (*Theta*) and third (*Psi*) rotation of the sequence.

Background

The coordinate transformation is based on three elementary rotations around the x, y, and z-axis:

$$\mathbf{T}_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}, \mathbf{T}_Y = \begin{bmatrix} \cos(\phi) & 0 & -\sin(\phi) \\ 0 & 1 & 0 \\ \sin(\phi) & 0 & \cos(\phi) \end{bmatrix}, \mathbf{T}_Z = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The entire transformation is then realized using three of these elementary rotations dependent on the chosen sequence:

$$\mathbf{T}(\phi, \theta, \psi) = \mathbf{T}_3(\psi) \mathbf{T}_2(\theta) \mathbf{T}_1(\phi)$$

This represents an intrinsic rotation, i.e. the elementary rotation is applied to the "local" axis in the current intermediate frame.

Example:

For a 3-1-2 sequence, the frame is first rotated around the z-axis $\mathbf{T}_1(\phi) = \mathbf{T}_Z(\phi)$. Then the intermediate frame (') is further rotated around the current x'-axis with $\mathbf{T}_2(\theta) = \mathbf{T}_X(\theta)$.

Finally, the resulting frame (") is rotated around the current y" with $\mathbf{T}_3(\psi) = \mathbf{T}_Y(\psi)$. This gives the overall transformation $\mathbf{T}(\phi, \theta, \psi) = \mathbf{T}_Y(\psi) \mathbf{T}_X(\theta) \mathbf{T}_Z(\phi)$.

Tip: In case data is only available for extrinsic rotations (i.e. elementary rotations around axes of a fixed frame), this can easily be converted by "inverting" the rotation sequence and the related angles, i.e. an extrinsic sequence y-x-z with angles a,b,c is equivalent to an intrinsic sequence z-x'-y" with angles c,b,a.

10.6 Dynamic System

Input/Output

The output of this block represents the response of an LTI dynamic system to the given input signal. By default, the units of the input/output are inherited from the input signal, but also any unit compatible to the input signal and an arbitrary output unit can be specified. Generally, only system dimensions matching to the input signal are supported. In case a 1D input signal is present, and the *System dimension* is 3D (or vice versa) an error is thrown.

Note: The unit of the model parameters is derived from the selected units, i.e. if the input unit is [A] and the output unit is [B], it is assumed that the system model parameters are provided in a unit [B/A].

Block Settings

The following settings need to be specified in the *LTI model definition* panel:

Representation

Defines by which means the dynamic system model is provided. Possible options are *Transfer Function*, *Zero-Pole-Gain*, *State-Space* and *Matlab LTI Model*. Related parameters are described below:

Transfer Function


A transfer function can either be *Defined by Coefficients* or by a *Rational function*.

- Case *Coefficients*: Dependent on the signal dimension, a set of real vectors for the *Numerator* and *Denominator* coefficients must be provided for all axis and cross-axis where at least one non-zero coefficient is mandatory for the *Denominator*.
- Case *Rational function*: Alternatively, the rational transfer can also be directly defined as a function of the Laplace variable $s (= i\omega = 2\pi if)$. The given form can be any kind of MATLAB compatible expression, e.g. $1 / (s^2 + 3*s)$ or $\text{sqrt}(2) * s / s + 1$.

Note: s needs to be used as frequency variable in the function. Multiplication (*), division (/) and power (^) operations may **not** be preceded by a dot, i.e. no element-wise operations.

Zero-Pole-Gain

This representation realizes a transfer function for each axis (and cross-axis) by a set of vectors for the transfer function *Poles* and *Zeros* and scalar *Gains*. Poles and zeros can be represented by real or complex numbers. Furthermore, they can also be non-existent. In this case, all rows of the respective table entry can be deleted.

Tip: In case the MATLAB input option () is checked, empty brackets “[]” are used to represent non-existent poles or zeros. Furthermore, no distinction between real and imaginary parts is necessary as MATLAB directly supports complex numbers.

State-Space

This realizes a state space representation of the dynamic system with a user-defined number of *State variables*. The size of the required real matrices *A*, *B*, *C* and *D* is automatically adapted according to this number.

Matlab LTI Model

This option allows the setup of any of the previous options with a single *Matlab workspace variable*. The variable must represent a LTI model which matches the selected system dimension (i.e. dim x dim).

Note: MATLAB frd models are not supported for the *Dynamic System* block.

Tip: Instead of a MATLAB variable name, also an explicit formulation of the system model can be used, e.g. `ones(3,3)*tf(1,[1 1])`.

Background

A **Transfer Function** which is defined by N_n real **Numerator** coefficients n_i and N_d real **Denominator** coefficients d_j results in the following representation for the affected axis (or cross-axis):

$$H(s) = \frac{\sum_{i=0}^{N_n-1} n_i s^i}{\sum_{j=0}^{N_d-1} d_j s^j}$$

The equivalent **Rational function** $H(s)$ can also be directly specified using the respective option. Similarly, the **Zero-Pole-Gain** model with real gain K , N_z **Zeros** z_i and N_p **Poles** p_j results in a function (for each axis and cross-axis):

$$H(s) = K \cdot \frac{\prod_{i=1}^{N_z} (s - z_i)}{\prod_{j=1}^{N_p} (s - p_j)}$$

The **State-Space** model with real matrices A, B, C and D realizes the following dynamics equation with n state variables and input of size *dim*:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$$

$$\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$$

with:

- **A** system matrix (n x n)
- **B** input matrix (n x dim)
- **C** output matrix (dim x n)
- **D** feedthrough matrix (dim x dim)
- **x** state variable vector (n x 1)
- **u** input signal (dim x 1)
- **y** output signal (dim x 1)

10.7 Feedback System

Feedback System blocks are a special block type. They can be used to realize any kind of (feedback) loop structure. Similar to a *Container*, a **Feedback System** has no block mask,

but a double-click opens a new [System Editor](#) (sub-)window which can be populated with other blocks as on the main [System Editor](#) level.

Note: No pointing error source blocks, evaluation blocks (*PEC*, *Total Error*), *Mapping* and *Container* blocks are allowed inside a *Feedback System*.

The link to a higher level of the [System Editor](#) is realized by the *Input Port* and *Output Port* blocks (their number within a *Feedback System* is not restricted).

The entire content of a feedback is internally converted to closed-loop transfer functions from all inputs to all output.

Note: Concerning the error evaluation, only the error signal at the inputs and outputs is externally visible (e.g. in the [Budget Tree View](#)). It is not possible to access or display intermediate results of building blocks within the feedback system.

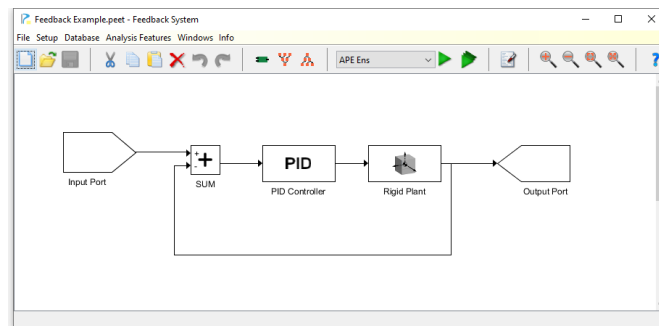


Figure 10-3: Exemplary loop realized in a feedback system block with one input port and one output port

Figure 10-3 shows an example of a simple loop structure with one input and one output. Nesting is also possible, i.e. further *Feedback System* blocks can be placed inside a *Feedback System* to represent implicitly inner loops.

Note: The modelled closed loop transfer function must represent the system behaviour from error signal input (e.g. disturbance error or measurement error) to the desired output which is usually different from the transfer function of the nominal signal input to the desired output.

10.8 Flexible Plant

Input/Output

The output of this block represents the 3D attitude response of a flexible body to a torque input. Thus, the unit of the input signal must be in a torque-compatible unit, i.e. [Force·Length]. For the output signal, any [Angle]-compatible unit can be selected. Only 3D input signals are supported.

Block Settings

The following settings need to be specified in the [Model parameters](#) panel:

[Modes](#)

The number n of flexible modes to be modelled.

Inertia

The 3x3 inertia matrix of the body given in any [Mass · Length²]-compatible unit.

Coupling coefficients

The 3xn matrix of coupling coefficients which describe the coupling of flexible modes into the spacecraft body rotation in any [$\sqrt{\text{Mass} \cdot \text{Length}}$]-compatible unit

Cantilever frequency

The nx1 vector of cantilever frequencies associated to the n modes in any [Frequency]-compatible unit.

Damping ratio

The nx1 vector of dimensionless damping ratios for each of the flexible modes

Background

The flexible body dynamics takes into account n flexible modes defined by the user to extend the rigid body dynamics. The underlying model is given by the following set of differential equations (note that the coupling between the flexure and the spacecraft linear acceleration/force is neglected):

$$\Theta \dot{\omega} - \delta \ddot{a} = \mathbf{N}$$

$$\ddot{a} + 2\zeta\Omega\dot{a} + \Omega^2 a = \delta^T \dot{\Omega}$$

with:

- Θ spacecraft inertia matrix (3x3)
- ω vector of spacecraft angular rates (3x1); integration of this quantity gives the block output
- δ matrix of coupling coefficients (3xn)
- \mathbf{N} vector of torques acting on the spacecraft body (3x1), i.e. the block input
- a vector containing the amplitudes of n flexible modes (nx1)
- ζ diagonal matrix containing the damping ratio of the flexible modes (nxn)
- Ω diagonal matrix containing the cantilever frequencies of the flexible modes (nxn)

Internally, the model is first realized in an equivalent state-space representation, which is given by:

$$\begin{bmatrix} \Theta & -\delta & \mathbf{0} \\ \delta^T & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \dot{\mathbf{x}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 2\zeta\Omega & \Omega^2 \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{N} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

with state vector $\mathbf{x} = [\omega, \dot{a}, a]^T$ and \mathbf{I} and $\mathbf{0}$ unity and zero-matrices of proper size. Furthermore, above system model is extended such that the attitude (rather than the rates) is used as output.

10.9 General Periodic Error

Output

The output of this block represents a temporal periodic (but non-sinusoidal) error source dependent on the selected *Signal type*. It can also be used as a workaround to model certain transient signals under the assumption that they (re)occur periodically.

The *Signal dimension* can be either **1D** or **3D** and an arbitrary *Output unit* can be specified.

Tip: The helper function `checkGeneralPeriodicApproximation` can be used to compare the signal approximation with the exact signal (see chapter 7.10).

Block Settings

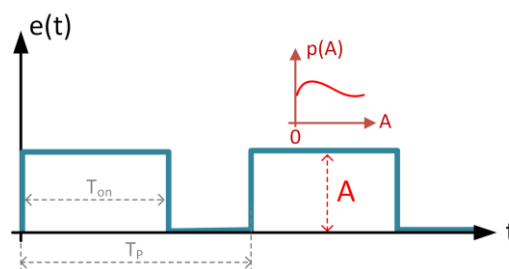
The *General Periodic Error* block has two common settings which are independent of the selected *Signal type*.

First the positive *Value* for the *Fundamental period* T_p of the periodic signal in any [Time]-compatible unit must be specified.

Second, the *Amplitude* A of the signal can be defined using any supported *Distribution type*, namely **Delta (No Distribution)**, **Uniform**, **Bimodal (Arcsine)**, **Gaussian**, **Rayleigh**, **Truncated Gaussian**, **Beta** and **User-Defined** (with *Parameters* as described in chapter 10.16 for the *PES (Time-Constant)* block).

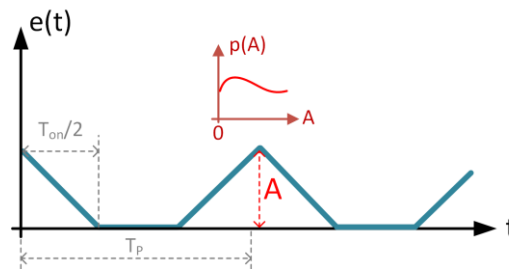
All remaining *Signal parameters* depend on the selected *Signal type* which allows choosing from the following models:

- Rectangular**
 Realizes a step signal defined by its fundamental period T_p , its amplitude A and the *On-Off ratio* (T_{on}/T_p) in [%].



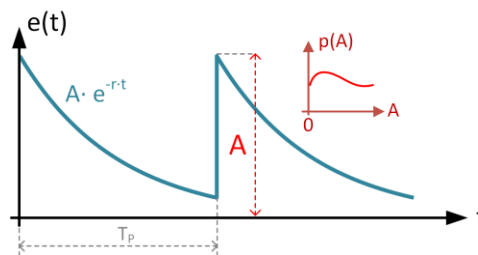
Note: Nominally, the *On-Off ratio* can be defined in the range 0-100%. For good approximations with the model however, it is advised to use range a of 5-95%, i.e. such that both significant on- and off-phases are present.

- Triangular**
 Realizes a triangular signal defined by its fundamental period T_p , its amplitude A and the *On-Off ratio* (T_{on}/T_p) in [%].



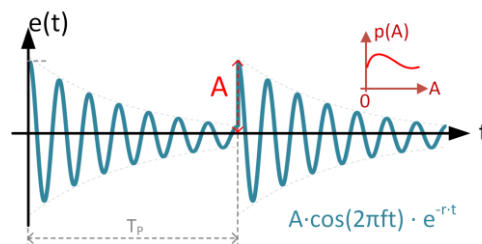
Note: Nominally, the *On-Off ratio* can be defined in the range 0-100%. For good approximations with the model however, it is advised to use values >5%, i.e. such that a significant triangular phase is present.

- Exponential Decay**
 Realizes a signal exponentially decaying from amplitude A towards zero with the dimensionless *Decay rate* r :



Note: Nominally, the *Decay rate* can be any positive scalar value. For good approximations with the model however, it is advised to model signals with a dominant decay phase (i.e. neither a 'too fast' nor a 'too slow' reset). Suggested values are in the range $T_p \cdot r \approx [3, 20]$.

- Decaying Cosine**
 Realizes a cosine signal with *Cosine frequency* f and initial amplitude A which exponentially decays towards zero with the dimensionless *Decay rate* r :



Note: As this model is based on the exponential decay model above, also the same parameter range is suggested concerning *Decay rate* and fundamental period, i.e. $T_p \cdot r \approx [3, 20]$. Further the cosine frequency f must be smaller than the maximum frequency of the Fourier series approximation ($n=50$) with a certain margin. It is suggested to keep $f < 0.85 \cdot n/T_p$.

Background

All models are internally parameterized as a Fourier series approximation with a specific coefficient set for each *Signal type* (see [RD7], chapter 6.6 for the precise model coefficients). The error signal is then realized as standard sinusoidal signal with components at different frequencies (according to the Fourier frequencies of the series approximation and amplitudes corresponding to the series coefficients).

10.10 GPS

Output

The output of this block represents a 3D bias and a simplified noise spectrum of a GPS sensor. Dependent on the *Output selection*, it is possible to model *Position* or *Velocity* errors. It is also possible to define any [Length]- or [Velocity]-related unit for the output signal.

Block Settings

Noise bandwidth

Sets the upper limit of the noise bandwidth time in any [1/Time]-compatible unit.

If selected, the following settings need to be specified in the *Position errors* panel:

Position bias

Sets the bias vector in any [Length]-compatible unit.

Position noise

Sets the standard deviations of the position noise (in any [Length]-compatible unit) which determines the flat part of the noise spectrum.

Position random walk

Sets the standard deviations of the position random walk (in any [Length/Time]-compatible unit) which determines the 1/f part of the noise spectrum.

Analogously, if selected, the following settings need to be specified in the *Velocity errors* panel:

Velocity bias

Sets the bias vector in any [Velocity]-compatible unit.

Velocity noise

Sets the standard deviations of the velocity noise (in any [Velocity]-compatible unit) which determines the flat part of the noise spectrum.

Velocity random walk

Sets the standard deviations of the velocity random walk (in any [Velocity/Time]-compatible unit) which determines the 1/f part of the noise spectrum.

Background

For the simplified noise spectra, the random walk contributions basically need to be integrated over time. This is internally realized, by feeding a white noise through an integrator which essentially gives a 1/f contribution to the spectrum which is added to the flat ("white") background of the spectrum.

The specified noise bandwidth and the standard deviations determine the magnitudes of the spectra for the flat and random walk contributions.

10.11 Gyro Rate Noise

Output

The output of this block represents the noise spectrum of a gyroscopic sensor (in case the *Signal Dimension* is 3D, an identical spectrum is assumed for all axes). No rate bias is included in the model. Any unit compatible to [Angle/Time] can be defined as *Output unit*.

Block Settings

The following *Noise parameters* need to be specified in the respective panel:

Angle random walk

The scalar coefficient N for the angle random walk contribution to the power spectrum given in a unit compatible to [(Angle/Time)/√Hz] or equivalently [Angle/√Time].

Rate random walk

The scalar coefficient K for the rate random walk contribution to the power spectrum given in a unit compatible to [(Angle/Time²)/√Hz] or equivalently [Angle/√(Time³)].

Bias instability

The scalar coefficient B for the bias instability to the power spectrum given in a unit compatible to [(Angle/√(Time³))/√Hz] or equivalently [Angle/Time].

Quantization noise

The scalar quantization noise coefficient Q for the quantization noise contribution to the power spectrum given in a unit compatible to [(Angle/√Time)/√Hz] or equivalently [Angle].

Sampling period

The scalar sampling period in a unit compatible to [Time] related to the quantization noise.

Background

The model for the gyro noise spectrum with the given parameters is based on [RD2] and realizes the following shape:

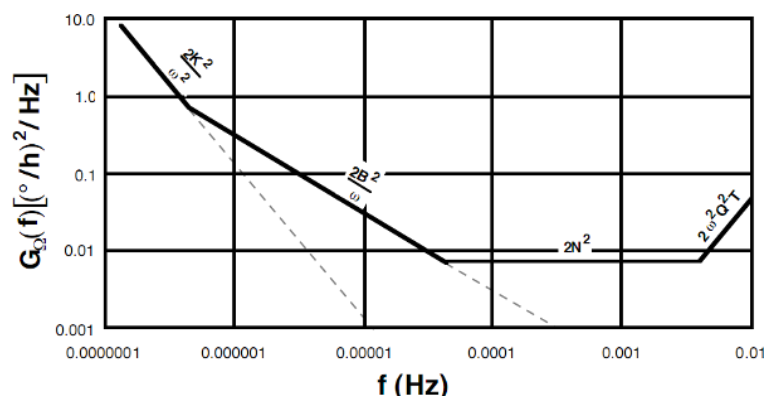


Figure 10-4: Gyro rate noise PSD derived from [RD2]

10.12 Gyro-Stellar Estimator

Input/Output

The outputs of this block represent the 3D attitude estimation errors (1st output) and rate estimation errors (2nd output) after filtering the measurement inputs with a fixed-gain Kalman filter.

Both attitude measurement errors (1st input) and rate measurement errors (2nd input) can contain time-constant and time-random contributions and need to be provided in any [Angle]- and [Angle/Time]-compatible unit. For the output, it is possible to *Inherit output units from inputs* (default) or to define any other compatible unit individually.

Block Settings

The only settings which needs to be specified in the *Kalman gains* panel are the respective (positive) gains K_d and K_p individually for each axis.

Background

The gyro-stellar estimator is a model-replacement Kalman filter used to fuse data from gyro and star-tracker to get an estimate of the spacecraft attitude and of the gyro bias. The measurement update is then performed using star-tracker data.

The following fixed-gain model is realized independently for each axis based on the derivation given in [RD5] (implementation #3):

$$\begin{bmatrix} \hat{e}_\phi \\ \hat{e}_\omega \end{bmatrix} = \frac{1}{s^2 + K_p s + K_d} \begin{bmatrix} sK_p + K_d & s \\ sK_d & s^2 + sK_p \end{bmatrix} \begin{bmatrix} n_\phi \\ n_\omega \end{bmatrix}$$

with

- \hat{e}_ϕ attitude estimation error (first output signal)
- \hat{e}_ω (gyro) rate bias estimation error (second output signal)
- K_p, K_d Kalman gains > 0
- n_ϕ (star tracker) attitude measurement errors (first input signal)
- n_ω (gyro) rate measurement error (second input signal)
- s Laplace domain frequency variable ($s = i\omega = 2\pi if$)

Note that K_d is equivalent to $-k_2$ in [RD5] to allow both gains to be specified as positive numbers.

10.13 Input PEC

Output

The output of this block represents an 'evaluated' error signal, i.e. a signal where both the respective error index (metric) and the statistical interpretation are assumed to be already applied. It can be used, for instance, to include results from external analyses which have already covered above-mentioned steps. Consequently, the output can also be individually specified for each requirement set defined in a scenario, as the metrics and statistical interpretation usually differ for each requirement set.

The *Signal dimension* can be either 1D or 3D and an arbitrary *Output unit* can be specified.

Block Settings

The *General* panel consists of a number of tabs, where each tab represents the error source contribution to a specific requirement set present in the scenario. By default, the error source *Type* is set to *None*. This ensures that the block can also be used in cases where no externally analyzed data is available for all requirement sets. For all other cases, the following models can be used:

- **Time-Constant PDF**
A numerically defined *User PDF* which represents the effective contribution of a time-constant error source
- **Time-Random PDF**
A numerically defined *User PDF* which represents the effective contribution of a time-random error source
- **PSD**
The magnitude of the effective power spectral density of a time-random error for which (optionally) either an *Explicit Cross-spectrum* or one defined *By Coherence* can be specified.

Background

For PDF inputs, the *Input PEC* internally generates samples according to the input PDF and maps these to either the Constant random variable (for *Time-Constant PDF* input) or the Random variable component (for *Time-Random PDF* input). Different to any standard random variable input, no further modification is applied to the samples (i.e. concerning error metrics and statistical interpretation).

For PSD inputs, no frequency domain metric related to the requirement set is applied. In case of spectral requirements, the input thus directly reflects the contribution of the source. For statistical requirements, the statistical interpretation is applied as for any other PSD contribution, just skipping the step of previously applying the frequency domain metric.

Note: No consistency checks can be applied to identify if an input is 'reasonable' for a given requirement set, i.e. it is fully up to the user to ensure that the provided input corresponds to the result for a given requirement set in terms of metric and statistical interpretation (e.g. that the metrics are correctly applied or that a PDF actually represents the ensemble or temporal behaviour according to the statistical interpretation chosen for a requirement set).

Tip: The only reasonable input for a *Time-Constant PDF* in case of temporal interpretation is a discrete value which represents the worst-case.

10.14 Mapping

Input/Output

The output of this block represents a 3D spatial distribution of a one-dimensional signal, e.g. for mapping the force along a thrust axis of one thruster (or several thrusters) to a force or torque noise in the reference frame of the pointing error. This block thus accepts only 1D input signals, otherwise an error message is displayed when connecting the block. The

default option for the units is *Inherit units*, but also any compatible *Input unit* and an arbitrary *Output unit* can be specified.

Note: The unit of the model parameters is derived from the selected units, i.e. if the input unit is [A] and the output unit is [B], it is assumed that the system model parameters are provided in a unit [B/A].

Block Settings

The following settings need to be specified in the *Mapping parameters* panel:

Number of devices

This defines the number N of identical devices which should be used for the mapping. The total number is internally limited to 99 devices.

Mapping matrix

This defines the Nx3 matrix with the conversion factors for the 3D signal. Each row defines the mapping vector of one device (e.g. its orientation in space, a lever arm mapping a scalar thrust to a torque vector).

Background

The *Mapping* block internally duplicates the 1D input signal N times (according to the number of devices), extends this copy for each device to a 3D signal scaled with the conversion factors in the mapping matrix and sums the contributions from all devices. For both random variable type inputs and noise spectra, this approach implicitly assumes mutual correlation (or coherence) between the contributions of different devices.

Tip: This block only serves as a quick-helper to realize multiple devices with the same properties. In case no correlation or coherence between different devices needs to be realized, the different devices can alternatively be modelled by copying the 1D source blocks and feeding their outputs to “single device” mapping blocks each (or by directly setting them up as 3D components).

10.15 PEC

Input/Output

The *PEC* block supports 1D and 3D input signals and the output dimension is always identical to the input dimension. By default, the *Inherit unit from input* option is enabled, but it is also possible to specify any other evaluation unit compatible with the input unit.

Block Settings

The *PEC* block has no specific block parameters. Requirement values which shall be associated with this block are completely defined via the menu *Setup* → *Scenario Definition* in the *Requirement Specification* tabs (see chapter 6.4.2.2).

Background

The *PEC* block has no effect on the input signal itself and just routes the input signal to the output. It is only used to evaluate the current error signal with respect to the given requirement (and compare it to specified requirement value(s) if provided).

10.16 PES (Time-Constant)

Output

The output of this generic block represents the contribution of a time-constant error source. The time-constant error can either be represented by a discrete value or as an ensemble-random quantity defined by a statistical distribution, i.e. a PDF p_E (see Figure 10-5).

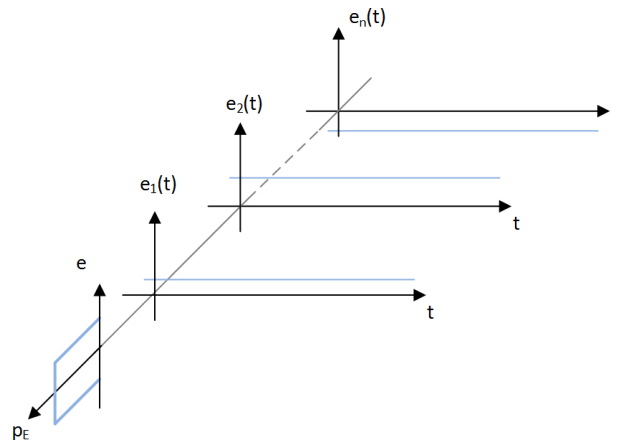


Figure 10-5: PES (Time-Constant): Temporal and ensemble behaviour of the error e for different realizations

The *Signal dimension* can be either 1D or 3D and an arbitrary *Output unit* can be specified.

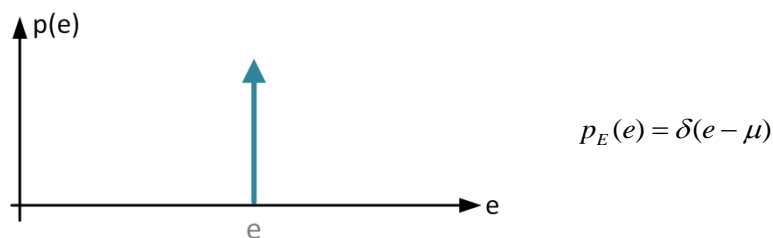
Block Settings

The following setting needs to be specified in the *Ensemble settings* panel:

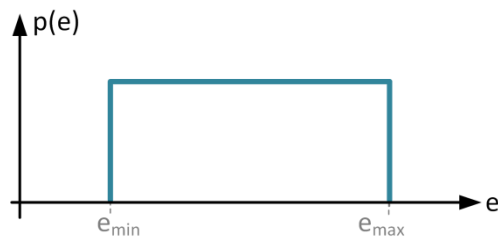
Distribution Type

Defines the statistical distribution for the ensemble behaviour of the error source determines the *Parameters* to be specified in the respective panel below. The following options are available:

- Delta (No Distribution)**
 Defines a constant bias for each axis fully determined by an arbitrary deterministic *Value* μ . The PDF of such an error is given by (δ denotes the Dirac-Delta function):



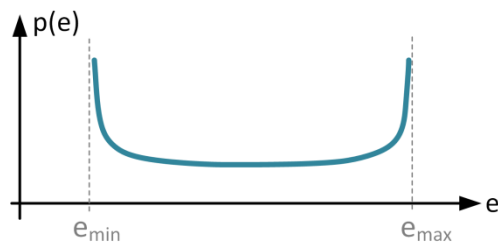
- Uniform**
 Defines a uniformly distributed error between a *Minimum* value e_{min} and a *Maximum* value e_{max} ($\geq e_{min}$) for each axis. The PDF of such an error is given by:



$$p_E(e) = \frac{1}{e_{\max} - e_{\min}}$$

- Bimodal (Arcsine)**

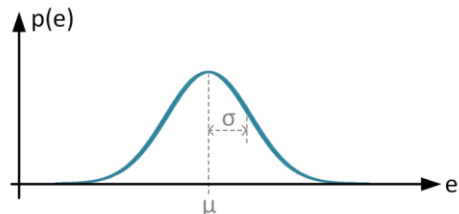
Defines a “bimodal”, i.e. arcsine distributed error between a *Minimum* value e_{\min} and a *Maximum* value $e_{\max} (\geq e_{\min})$ for each axis. The PDF of such an error is given by:



$$p_E(e) = \frac{\pi^{-1}}{\sqrt{(e_{\max} - e)(e - e_{\min})}}$$

- Gaussian**

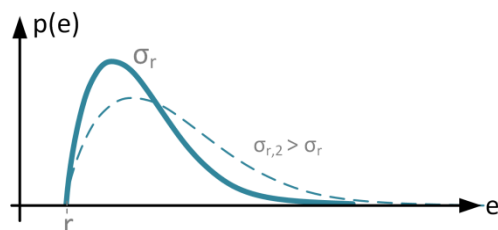
Defines a normal distributed error individually for each axis by specifying its *Mean value* μ and its *Standard deviation* $\sigma > 0$. The PDF of such an error is given by:



$$p_E(e) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{e-\mu}{\sigma}\right)^2\right]$$

- Rayleigh**

Defines a Rayleigh distributed error individually for each axis by its *Scale parameter* $\sigma_r > 0$. Furthermore, an additional *Shift parameter* r is introduced which removes the restriction of a zero minimum value ($r=0$ represents the “standard” Rayleigh distribution). The PDF of such an error is given by:



$$p_E(e) = \frac{e-r}{\sigma_r^2} \exp\left[-\frac{(e-r)^2}{2\sigma_r^2}\right]$$

(for $e \geq r$, zero else)

- Truncated Gaussian**

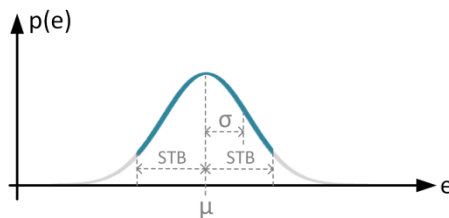
Defines an error which follows a normal distribution which is truncated at one or two given bounds dependent on the selected *Truncation type*. In all cases, first the *Mean value* μ and the *Standard deviation* $\sigma > 0$ need to be specified.

Note: These values refer to the “original” unbounded distribution. Mean and standard deviation of the truncated PDF are thus different.

The *Truncation type* can be one of the following (with resulting PDFs as shown below):

Symmetric Truncation

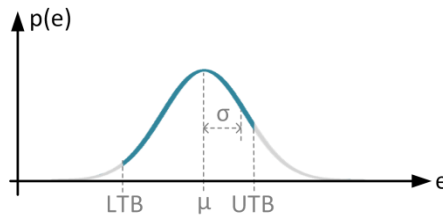
The Gaussian distribution is truncated at a given (non-negative) *Symmetric bound* (*STB*) relative to the provided mean value.



$$p_E(e) = \frac{\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{e-\mu}{\sigma}\right)^2\right]}{CDF_G(\mu + STB) - CDF_G(\mu - STB)}$$

Two Sided truncation

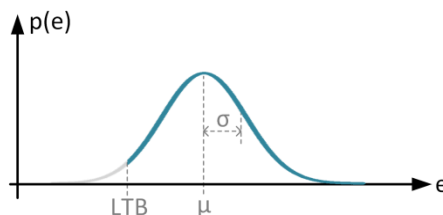
The Gaussian distribution is truncated at an arbitrary *Lower bound* (*LTB*) and *Upper bound* ($UTB \geq LTB$), i.e. not with respect to the mean.



$$p_E(e) = \frac{\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{e-\mu}{\sigma}\right)^2\right]}{CDF_G(UTB) - CDF_G(LTB)}$$

Lower Bound truncation

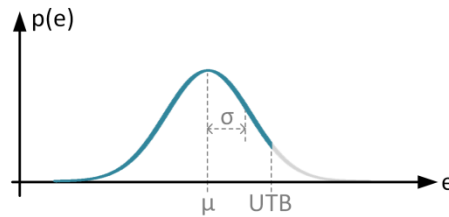
The Gaussian distribution is truncated at an arbitrary *Lower bound* (*LTB*).



$$p_E(e) = \frac{\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{e-\mu}{\sigma}\right)^2\right]}{1 - CDF_G(LTB)}$$

Upper Bound truncation

The Gaussian distribution is truncated at an arbitrary *Upper bound* (*UTB*).



$$p_E(e) = \frac{\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{e-\mu}{\sigma}\right)^2\right]}{CDF_G(UTB)}$$

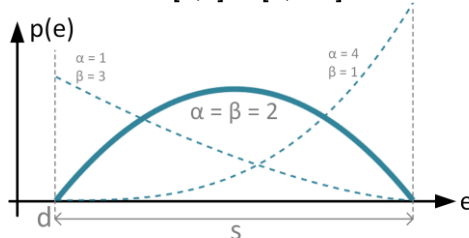
In all cases above, CDF_G denotes the cumulative distribution function (CDF) of a Gaussian distribution:

$$CDF_G(e) = \frac{1}{2} \left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right) \right]$$

where erf denotes the error function. The correction with the CDF is necessary to “normalize” the PDF after truncation.

- Beta**

Defines a Beta distributed error individually for each axis by the *Shape parameter* $\alpha \geq 0$ and the *Shape parameter* $\beta \geq 0$. Further, an additional *Scale parameter* $s \geq 0$ and an additional *Offset parameter* d have been introduced to extend the domain of definition from $[0,1]$ to $[d, s+d]$. The PDF of such an error is given by:



$$p_E(e) = \frac{\left(\frac{e-d}{s}\right)^{\alpha-1} \left(1 - \frac{e-d}{s}\right)^{\beta-1}}{|s| \operatorname{Beta} \alpha, \beta}$$

(for $d \leq e \leq d+s$, zero else)

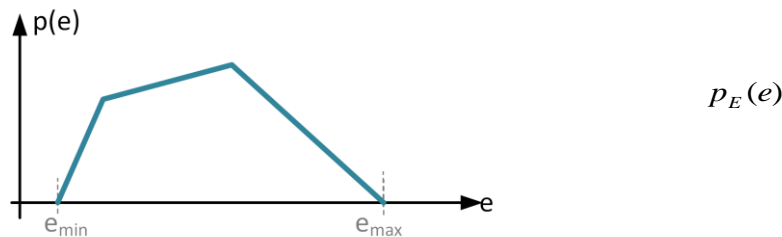
where the standard Beta distribution itself is defined by the Gamma function Γ :

$$\operatorname{Beta} \alpha, \beta = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)}$$

- User-Defined**

Defines an arbitrary bounded *User PDF* individually for each axis. The input data is provided in tabular form by specifying the error value and a related “density” point by point. If the specified data does not represent a valid PDF (i.e. no unity area in the given range, it is automatically converted into a valid PDF by proper normalization.

A valid setup further requires at least two points per axis to be specified. The first and last point provided represent the explicit bounds of the distribution.



10.17 PES (Time-Random)

Output and Top-Level Block Settings

The output of this generic block represents the contribution of a time-random error source. Independent from the chosen error model, the *Signal dimension* can be either **1D** or **3D** and an arbitrary *Output unit* can be specified.

According to AST-1 in [AD2], time-random errors can either be described as (Time-) **Random Variable** or as **Random Process** in case sufficient information on the frequency spectrum of the error source is available. The various options and settings dependent on the selected *PES description* are introduced in the following subsections.

10.17.1 Random Variable

Generally, a time-random error can be described by statistical distributions (see Figure 10-6). The PDF $p_T(e, \beta_{1...k})$ describes the temporal behaviour of the error source, where $\beta_{1...k}$ denotes the parameters of this distribution (e.g. mean value and standard deviation in case of a Gaussian). Furthermore - and completely optional - in most cases it is also possible to define a statistical distribution $p_E(\beta_i)$ for one of the parameters of the temporal behaviour. This should be understood as an ensemble-randomness, i.e. the parameter is considered as constant over time, but can vary over different "conditions" (e.g. observations, satellites, etc.).

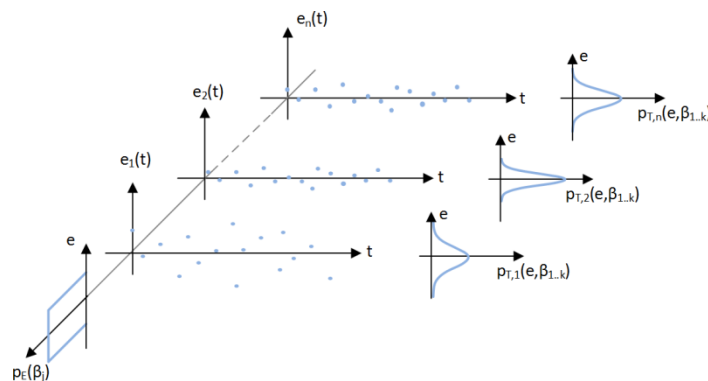


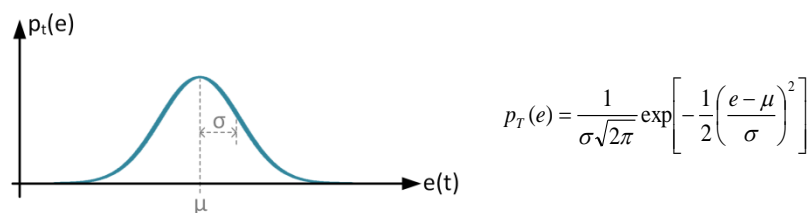
Figure 10-6: PES (Time-Random): General temporal behaviour of the error e with ensemble-random parameter (optional)

PEET provides three different *Signal classes* (**Gaussian**, **Uniform** and **Drift**) for random variable in accordance with [AD2].

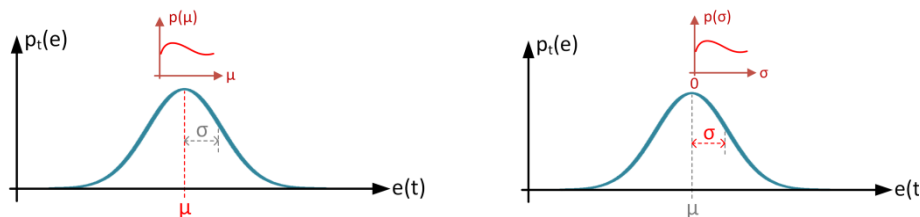
Note: The transfer (AST-2 in [AD2]) of random variable error sources through a dynamic system can only be conservatively approximated using certain assumptions (lack of information about their frequency spectrum, see [RD7]). Therefore, it is recommended to use a random process description whenever possible when dynamic system transfer of a source is required.

10.17.1.1 Gaussian Random Variable

The **Gaussian Signal class** models a normal distributed temporal error individually for each axis by specifying its **Mean value** μ and its **Standard deviation** $\sigma > 0$ in the respective panels, i.e. a PDF $p_T(e, \mu, \sigma)$ in terms of Figure 10-6.



Optionally, as illustrated below, one of these parameters can be chosen as **Distributed parameter** (None by default) in the **Ensemble settings** panel. In terms of Figure 10-6, this is equivalent to specifying the ensemble PDF $p_E(\beta_j)$ with either $\beta_j = \mu$ or $\beta_j = \sigma$.



Available options required for the selected **Distribution** are equivalent to those for the **PES (Time-Constant)** block described in chapter 10.16, namely **Uniform**, **Bimodal (Arcsine)**, **Gaussian**, **Rayleigh**, **Truncated Gaussian**, **Beta** and **User-Defined**.

In case the **Mean Value** is chosen as **Distributed Parameter**, no further restrictions apply. This is different in the case of **Standard Deviation**, as the latter needs to be non-negative. Consequently, in case the lower bound of any **Distribution** is < 0 , PEET automatically truncates the distribution at 0.

Tip: The Quick-Help (?) for the **Distribution** gives further hints on these restrictions.

10.17.1.2 Truncated Gaussian Random Variable

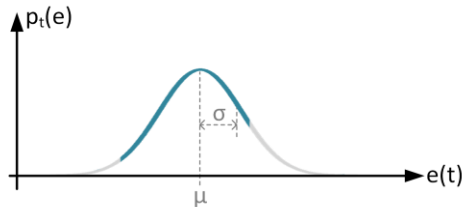
The **Truncated Gaussian Signal class** models a normal distributed temporal error similar to the one described in the previous subchapter, but explicit bounds can be specified for the error values dependent on the selected **Truncation type**. In all cases, first the **Mean value** μ and the **Standard deviation** $\sigma > 0$ need to be specified.

Note: These values refer to the “original” unbounded distribution. Mean and standard deviation of the truncated PDF are thus different.

The **Truncation type** can be one of the following (with resulting PDFs as shown below):

- **Symmetric** truncation

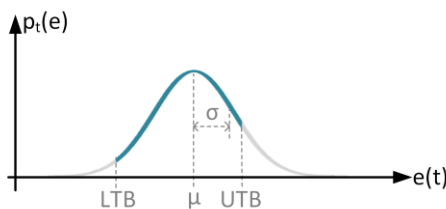
The Gaussian distribution is truncated at a given (non-negative) *Symmetric bound (STB)* relative to the provided mean value.



$$p_T(e) = \frac{\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{e-\mu}{\sigma}\right)^2\right]}{CDF_G(\mu + STB) - CDF_G(\mu - STB)}$$

- **Two Sided** truncation

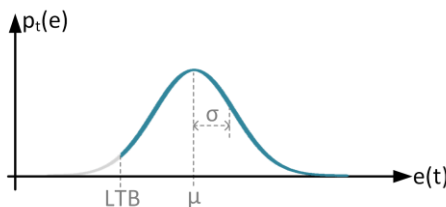
The Gaussian distribution is truncated at an arbitrary *Lower truncation bound (LTB)* and *Upper truncation bound (UTB)*, i.e. not with respect to the mean.



$$p_T(e) = \frac{\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{e-\mu}{\sigma}\right)^2\right]}{CDF_G(UTB) - CDF_G(LTB)}$$

- **Lower Bound** truncation

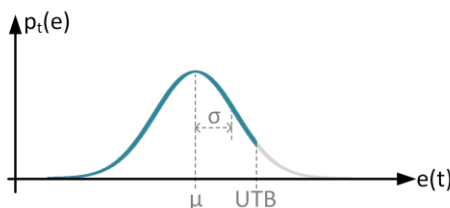
The Gaussian distribution is truncated at an arbitrary *Lower bound (LTB)*.



$$p_T(e) = \frac{\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{e-\mu}{\sigma}\right)^2\right]}{1 - CDF_G(LTB)}$$

- **Upper Bound** truncation

The Gaussian distribution is truncated at an arbitrary *Upper bound (UTB)*.



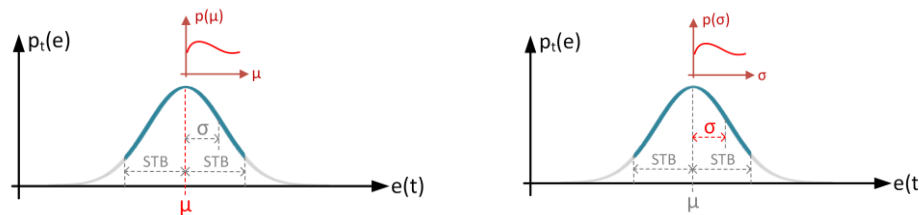
$$p_T(e) = \frac{\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{e-\mu}{\sigma}\right)^2\right]}{CDF_G(UTB)}$$

In all PDF expressions above, CDF_G denotes the cumulative distribution function (CDF) of a Gaussian distribution:

$$CDF_G(e) = \frac{1}{2} \left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right) \right]$$

where erf denotes the error function. The correction with the CDF is necessary to “normalize” the PDF after truncation.

Optionally, as for the non-truncated Gaussian case, the **Mean Value** or **Standard Deviation** can be chosen as **Distributed parameter** (**None** by default) in the **Ensemble settings** panel. In terms of Figure 10-6, this is equivalent to specifying the ensemble PDF $p_{\epsilon}(\beta_j)$ with either $\beta_j = \mu$ or $\beta_j = \sigma$.



Note: Having selected a **Distributed Parameter**, only a **Symmetric** truncation can be specified.

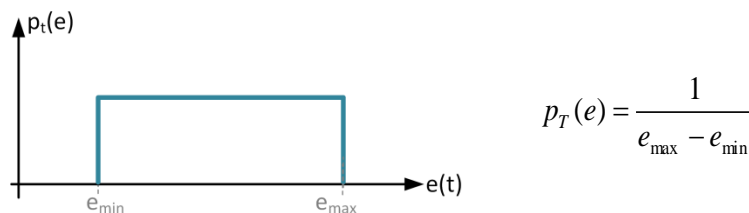
Available options required for the selected **Distribution** are equivalent to those for the **PES (Time-Constant)** block described in chapter 10.16, namely **Uniform**, **Bimodal (Arcsine)**, **Gaussian**, **Rayleigh**, **Truncated Gaussian**, **Beta** and **User-Defined**.

In case the **Mean Value** is chosen as **Distributed Parameter**, no further restrictions apply. This is different in the case of **Standard Deviation**, as the latter needs to be non-negative. Consequently, in case the lower bound of any **Distribution** is < 0 , PEET automatically truncates the distribution at 0.

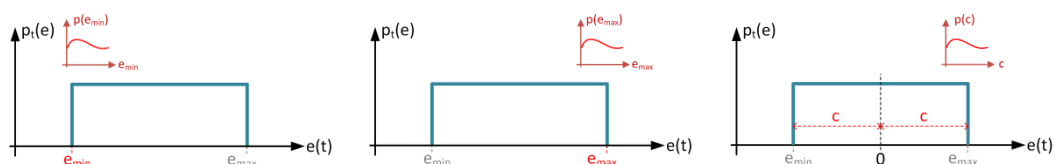
Tip: The Quick-Help (?) for the **Distribution** gives further hints on these restrictions.

10.17.1.3 Uniform Random Variable

The **Uniform Signal class** models an equally distributed temporal error individually for each axis by specifying its **Lower bound** e_{min} and its **Upper bound** e_{max} in the respective panels, i.e. a PDF $p_T(e, e_{min}, e_{max})$ in terms of Figure 10-6.



Optionally, as illustrated below, one of these distribution parameters can also be chosen as **Distributed parameter** (**None** by default) in the **Ensemble settings** panel. As additional option, also a distributed **Range** > 0 of a zero-mean uniform distribution can be specified.



In terms of Figure 10-6, this equivalent to specifying the PDF $p_E(\beta_j)$ with either $\beta_j = e_{\min}$, $\beta_j = e_{\max}$ or $\beta_j = c$.

Available options required for the selected *Distribution* are equivalent to those for *PES (Time-Constant)* block described in chapter 10.16, namely *Uniform*, *Bimodal (Arcsine)*, *Gaussian*, *Rayleigh*, *Truncated Gaussian*, *Beta* and *User-Defined*.

As a uniform distribution has a limited support, further restrictions apply on the possible values for the *Distributed Parameter*:

- In case the maximum value of a distributed *Lower Bound* exceeds the specified *Upper bound* e_{\max} , PEET automatically truncates the distribution at this upper bound.
- In case the minimum value of a distributed *Upper Bound* falls below the specified *Lower bound* e_{\min} , PEET automatically truncates the distribution at this lower bound.
- In case the minimum value of a distributed *Range* falls below 0, PEET automatically truncates the distribution at zero.

Tip: The Quick-Help (?) for the *Distribution* gives further hints on these restrictions.

10.17.1.4 Drift signal

The *Drift Signal class* models a special kind of error source. It realizes a linear drift in time (with individual *Drift rate* D for each axis) which is repeatedly corrected to zero after a certain *Reset time* Δt_D (common to all axes). Although the temporal PDF $p_T(e, D, \Delta t_D)$ is always uniform with lower bound zero and upper bound $D \cdot \Delta t_D$, this signal is in fact not random, but deterministic.

Internally, always full cycles (i.e. an integer “sawtooth” number) are assumed, to be compliant with the statistics given in [RD3].

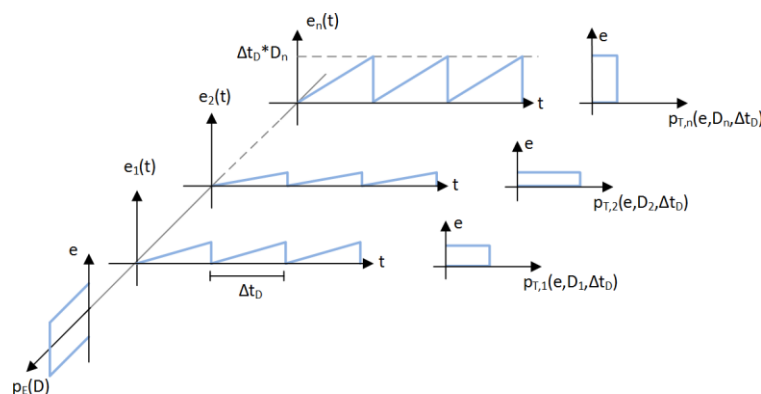


Figure 10-7: PES (Time-Random) - Drift: Temporal behaviour of the error e with ensemble-random drift rate (optional)

Note: Different to previous PEET versions (V1.0.1 and earlier), a frequency domain approach (Fourier series approximation) is implemented to represent the temporal drift error signal.

This enables a routing of drift signal through any dynamic system block model, taking into account resets in the time-windowed analysis and temporal "correlation" of drift signals with different reset times and other periodic signals (at the cost of a slightly less accurate representation of the temporal PDF itself). More information about the implementation is provided in the "Drift Error" chapters of [RD7].

Optionally, as indicated in Figure 10-7, a *Distribution* (Delta (No Distribution)) by default, i.e. a fixed value) can be assigned for the drift rate in the *Ensemble settings* panel. The available options are equivalent to those for *PES (Time-Constant)* block described in chapter 10.16, namely *Uniform*, *Bimodal (Arcsine)*, *Gaussian*, *Rayleigh*, *Truncated Gaussian*, *Beta* and *User-Defined*. There is no restriction on the range of parameter values.

10.17.2 Random Process

If – different to the random variable description – also information about the frequency spectrum of an error source is available, it can be modelled as a random process. This approach has two major advantages:

- The error after the transfer analysis (AST-2 in [AD2]) can exactly be predicted, also in the case that signals are fed through dynamic system blocks.
- Metric filters can exactly be applied to the random process signals for the pointing error index contribution (AST-3 in [AD2]) and no approximations are necessary as partially applied in the Tables in [AD1].

PEET provides three different *Types* of random processes (*Periodic*, *PSD* and *BLWN*) in accordance with [AD2], where the last option is a derived simplified setup for a PSD.

10.17.2.1 Periodic Signal

The *Periodic* random process *Type* is a special kind of random process. It realizes a zero-mean sinusoidal signal in time which is defined by an amplitude $A > 0$.

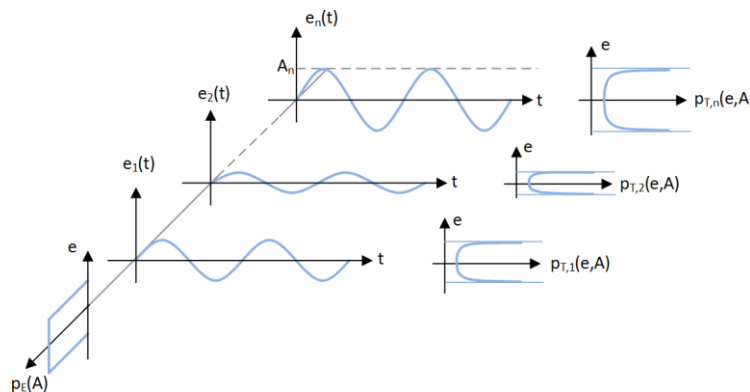


Figure 10-8: PES (Time-Random) - Periodic: Temporal behaviour of the error e with ensemble-random amplitude (optional)

Block Settings

Frequency and corresponding amplitude *Values* are jointly specified in a table (matrix) in the *Amplitude settings* panel. It is also possible to account for multiple frequencies (e.g. to model harmonics) and related amplitudes in one error source by adding additional rows to the table for additional frequencies.

In total, thus an $N_f \times (1+\text{dim})$ array of parameters needs to be provided, where N_f represents the number of different frequencies and dim the dimension of the signal.

Note: With the advanced statistical method, also the phases between periodic signals have an effect on the resulting PDF. By default, all periodic signals have zero-phase difference. Signal phases can be individually specified in the **Setup → Set Phase Relations** menu.

Optionally, as indicated in Figure 10-8, an *Amplitude distribution* (Delta (No Distribution) by default, i.e. a fixed value) can be assigned in the **Ensemble settings** panel. The available options are equivalent to those for the *PES (Time-Constant)* block described in chapter 10.16, namely **Uniform**, **Bimodal (Arcsine)**, **Gaussian**, **Rayleigh**, **Truncated Gaussian**, **Beta** and **User-Defined**. As the amplitude is restricted to be positive, consequently in case the lower bound of any *Amplitude distribution* is < 0 , PEET automatically truncates the distribution at 0.

Tip: The Quick-Help (?) for the *Amplitude distribution* gives further hints on this restriction.

In case a non-discrete *Amplitude distribution* is selected, the size of the data input table in the **Amplitude settings** panel is automatically adjusted to an $N_f \times (1 + N_p \cdot \text{dim})$ array, where N_p denotes the number of parameters required to describe the distribution (e.g. $N_p=2$ for a uniform distribution defined by its lower and upper bound).

Background

Although the temporal PDF $p_T(e,A)$ of a periodic signal is always a bimodal (arcsine) distribution with bounds $[-A, A]$, this signal is in fact not random, but deterministic. However, its power spectral density can explicitly be expressed as:

$$G(f) = \frac{A^2}{2} \delta(f - f_p)$$

where δ denotes the Dirac-Delta function and f_p is the frequency of the sinusoid. Internally, always full cycles (i.e. integer number of periods) are assumed, to be compliant with the statistics given in [AD1] and [AD2].

10.17.2.2 Power Spectral Density (PSD)

The **PSD Type** models a zero-mean stationary, ergodic Gaussian random process represented by an arbitrary user-defined power spectral density $G(f)$. The spectrum can also be limited to a certain bandwidth as indicated in Figure 10-9:

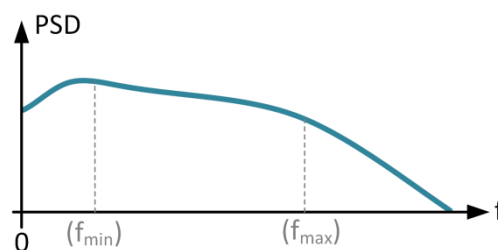


Figure 10-9: PES (Time-Random) - PSD: Spectrum of the error e with (optionally) limited bandwidth

Note: All spectral density models are defined as single-sided amplitude spectra (i.e. $P(f)=\sqrt{G(f)}$ on unit/ $\sqrt{\text{Hz}}$ level). The conversion to a power spectral density is carried out internally.

Block Settings

In the **General** settings panel, first the **Representation** and the method for the **Cross-spectrum** setup need to be specified.

The following options are available for the PSD **Representation**:

- **Analytical**

The amplitude spectra are defined by an analytical function of frequency in the **PSD components** panel. In the 1D case, this is one single function (P). In the 3D case, at least 3 functions for the autospectra (P_{xx} , P_{yy} , P_{zz}). Dependent on the selection for the **Cross-spectrum**, also functions for the latter (i.e. P_{xy} , P_{xz} and P_{yz}) are required.

In addition, the **Ensemble parameter** panel allows defining the **Distribution Type** of a 1D parameter (default: **None**) which can be used to describe a varying shape of any PSD component in each realization of an ensemble. The available options are equivalent to those for the **PES (Time-Constant)** block described in chapter 10.16, namely **Delta (No Distribution)**, **Uniform**, **Bimodal (Arcsine)**, **Gaussian**, **Rayleigh**, **Truncated Gaussian**, **Beta** and **User-Defined**.

Tip: The function definition accepts any MATLAB compatible notation, e.g.:

$$(f+p).^2./\text{sqrt}(f.^5+2.*f+\sin(p)) \quad \text{for} \quad P(f)=\sqrt{G(f)}=\frac{(f+p)^2}{\sqrt{f^5+2f+\sin(p)}}$$

Note: f needs to be used as frequency variable in the function, p as variable for the ensemble parameter. Multiplication ($.*$), division ($./$) and power ($.^$) operations need to be defined as element-wise, i.e. preceded by a dot.

- **Numerical**

The amplitude spectra are defined by a vector of frequencies and corresponding magnitudes of the spectrum in the **Model definition** panel. **Frequency** and **Magnitude** data is jointly specified in a table (matrix) where each frequency is represented by a row.

Consequently, for N_f frequency points and error source dimension dim , in total an $N_f \times (1+\text{dim})$ matrix of data needs to be provided to define the auto-spectra (P_{xx} , P_{yy} , P_{zz}). Dependent on the selection for the **Cross-spectrum**, also magnitudes for the latter (i.e. P_{xy} , P_{xz} and P_{yz}) are required which leads to total data matrix size of $N_f \times 7$.

- **LTI Model**

This option represents another auxiliary way to realize the magnitude of a spectrum. It is introduced as often noise shaping filters are used to create noise time series with a predefined spectrum in time-domain simulations. With this option, typical representations of these filters can directly be reused. The following options for the **Model Type** are available:

- **Transfer Function**
- **State Space**
- **Zero-Pole-Gain**


- **Matlab LTI Model**
- **Frequency Response Data**

The parameters and settings for the first four options in the **Model definition** panel are basically identical to the ones for the **Dynamic System** block (see chapter 10.6) and are not repeated here. The only difference - dependent on the selection for the **Cross-spectrum** - is that no cross-axis information needs can be specified for “transposed” entries (e.g. x-y and y-x) as it is fully determined by one of the two entries.

- The **Transfer Function** defined as **Rational function** is the LTI-equivalent for the **Analytical PSD Representation**. Thus, it equivalently provides the option to define a 1D parameter with a given **Distribution Type** to be used in any I/O of the entire transfer function. The given form can be any kind of MATLAB compatible expression, e.g. $1 / (s^2 + 3 * (s + p) + p) \cdot O$.

Note: s needs to be used as frequency variable in the function, p as variable for the ensemble parameter. Multiplication (*), division (/) and power (^) operations may **not** be preceded by a dot, i.e. no element-wise operations.

Frequency Response Data is the LTI-equivalent for the **Numerical PSD Representation**. It is defined by a vector of frequencies and corresponding (complex) frequency responses in the **Model definition** panel. **Frequency** and **Response** data is jointly specified in a table (matrix) where each frequency is represented by a row. Consequently, for N_f frequency points and error source dimension dim , in total an $N_f \times (1 + 2 \cdot dim)$ matrix of data needs to be provided to define the “auto-spectra” (real and imaginary part of responses for x, y, z). Dependent on the selection for the **Cross-spectrum**, also responses for the latter (i.e. xy, xz and yz) are required which leads to total data matrix size of $N_f \times 13$.

Note: In case the MATLAB input option () is checked, no distinction between real and imaginary parts is necessary as MATLAB directly supports complex numbers. This reduces the size of the data matrix to $N_f \times (1 + dim)$ or $N_f \times (1 + 2 \cdot dim)$ respectively.

The following options are available for the **Cross-Spectrum**:

- **By Coherence**
The cross-power spectra are automatically determined from a user-defined coherence factor, i.e. a constant coherence over all frequencies is realized. The definition of the coherence factors is managed globally for all error sources using the **Setup** → **Set Coherence** menu.

Tip: By default, all coherence factors are initialized with a zero value. That means without any modification in the **Set Coherence** menu, cross-power spectra are neglected (i.e. have zero magnitude).

- **Explicit**
The data for the cross-power spectra is explicitly defined by the user directly in the block dialog.

The following settings are available in the **Bandwidth** panel:

- **Global**
The defined spectrum is considered valid over the entire frequency range.

- **User-defined**

The defined spectrum is only valid within a certain bandwidth.

Tip: The **Bandwidth** panel is not available for numerical definitions of the spectra, (i.e. **Numerical** and **Frequency Response Data**). For these cases, the bandwidth is directly given by the provided frequency vector and zero magnitude outside this bandwidth is assumed.

Note: For the same reason, the information in the **Bandwidth** panel is ignored when the variable for the **Matlab LTI Model** represents a frequency response data (frd) object.

Background

The **PSD Type** models a zero-mean stationary, ergodic random process, i.e. its statistical properties do neither vary over time, nor over the ensemble of realizations. Furthermore, the underlying PDF is assumed to be Gaussian. Thus, the error source PDF is fully characterized through the knowledge of the PSD $G(f)$, since it alone determines the variance (or standard deviation) of the signal [RD4], at least when the process is zero-mean. For a single-sided spectrum, the relation between variance and PSD is then given by:

$$\sigma_e^2 = \int_0^{\infty} G(f) df$$

In case the spectrum has a limited bandwidth (as indicated in Figure 10-9), the bounds for the integration change to f_{\min} and f_{\max} respectively.

Another important property of this setup is that if a Gaussian process undergoes a linear transformation (e.g. a transfer through a linear time-invariant dynamic system in AST-2 of [AD2]), the output is still a Gaussian, i.e. the output properties can exactly be determined.

The **LTI Model Representation** of the spectrum first realizes a dynamic system model $H(s)$ (with $s = 2\pi if$). The respective magnitude of the amplitude spectrum is then computed by taking its absolute value, i.e.

$$P(f) = \sqrt{G(f)} = |H(2\pi if)|$$

10.17.2.3 Band-Limited White Noise (BLWN)

The **BLWN Type** models a special case of a power spectral density $G(f)=const$, i.e. it realizes a flat (“white”) spectrum with a certain variance and bandwidth and zero magnitude outside (see Figure 10-10).

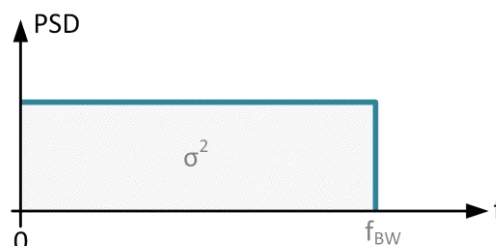


Figure 10-10: PES (Time-Random) – BLWN: Power spectral density of the error e

Block Settings

The only two parameters which need to be specified in the **Band-limited white noise** panel are the desired **Standard deviation** (for each axis) of the common noise **Noise bandwidth** (upper frequency bound).

Tip: To specify non-zero cross-spectra, use the **Setup → Set Coherence** menu.

Background

The magnitudes of the realized power and amplitude spectral density within the given bandwidth is given by:

$$G(f) = \frac{\sigma^2}{f_{BW}}$$

in [unit²/Hz] or

$$P(f) = \sigma / \sqrt{f_{BW}}$$

in [unit/√Hz] respectively.

10.18 PID Controller

Input/Output

The output of this block represents the control signal of an ideal Proportional-Integral-Derivative (PID) controller. By default, the **Inherit units** option is enabled, but also an any unit compatible to the input signal unit and an arbitrary output unit can be specified.

The block accepts both 1D and 3D input signals dependent on the selected **System Dimension**. In case the latter is 3D and a 1D input signal is present (or vice versa), an error is thrown.

Note: The unit of the model parameters is derived from the selected units, i.e. if the input unit is [A] and the output unit is [B], it is assumed that the system model parameters are provided in a unit [B/A].

Block Settings

The only settings which need to be specified in the **Controller gains** panel are a set of **Proportional**, **Integral** and **Differential** gains individually for each axis.

Background

The ideal controller is realized as a single-input single-output transfer function of the following form for each axis:

$$K(s) = K_P + \frac{K_I}{s} + K_D s$$

with:

- K total controller transfer function
- K_P proportional gain of the controller
- K_I integral gain of the controller
- K_D differential gain of the controller

- s Laplace domain frequency variable ($s = i\omega = 2\pi if$)

10.19 Reaction Wheel

Output

PEET provides two special pointing error source blocks for setting up disturbance forces and torques on the spacecraft interface which are generated by a single reaction wheel. The 3D output disturbance is always provided with respect to the wheel frame (defined by the wheel spin around the z-axis).

The model covers periodic (harmonic) error contributions as well as noise contributions dependent on the user input. The *Output unit* can be any [Force]- or [Torque]- compatible unit for the *Reaction Wheel (Force)* and *Reaction Wheel (Torque)* block respectively.

Tip: The orientation of the wheel with respect to the spacecraft/reference frame can be realized with the *Coordinate Transformation* block, multiple wheels by repeated use of the blocks.

The implemented models are based on [RD8] (which are further based on [RD9] and [RD10]) and briefly explained in the following subsections.

10.19.1 Reaction Wheel (Force)

The disturbance force model includes models for the radial and axial translation mode of the wheel and covers different kinds of parameter sets for the excitation force inputs. The definition of axial force parameters is optional.

Block Settings

The following settings need to be specified in the *Wheel properties* panel:

Wheel mass

The mass of the flywheel only in any [Mass]-compatible unit.

Wheel speed distribution

The statistical distribution of the (positive) wheel speed. The options are identical to the ones for the *PES (Time-Constant)* block (see chapter 10.16 for more details), i.e. *Delta (No Distribution)*, *Uniform*, *Gaussian*, *Bimodal (Arcsine)*, *Rayleigh*, *Truncated Gaussian*.

Note: This description of the wheel speed represents an ensemble of “working points”, i.e. the wheel speed is considered constant within each working point and varies only between different working points (e.g. operational conditions, observations, etc.)

Wheel speed

Specifies the parameters for the distribution dependent on the selected *Wheel speed distribution*. Any [1/Time]-compatible *Unit* can be selected. As only positive wheel speeds are allowed, any distribution will be automatically truncated at zero.

Broadband force noise

Determines the modelling approach for the force noise contribution (common for the *Radial mode* and *Axial mode*):

- **Standard deviation:** The noise is modelled as a zero-mean Gaussian time-random variable defined by its standard deviation (similar to a Gaussian time-random variable in the *PES (Time-Random)* block, see chapter 10.17.1.1).
- **Band-limited:** The noise is modelled as flat spectrum up to a given frequency similar to the band-limited white noise random process in the *PES (Time-Random)* block, see chapter 10.17.2.3).
- **PSD:** The noise is modelled following an arbitrary user-defined spectrum to the band-PSD option for a random process in the *PES (Time-Random)* block, see chapter 10.17.2.2).

The following settings need to be specified in the **Radial mode** panel:

Translation mode frequency

The fundamental frequency of the periodic disturbance in any [1/Time]-compatible unit.

Translation mode damping

The dimensionless damping coefficient for the periodic disturbance of the radial translation mode.

Noise standard deviation

The standard deviation of the noise on the radial axis (available for *Broadband force noise* options **Standard deviation** and **Band-limited**).

Noise bandwidth

The bandwidth (largest frequency) of the noise on the radial axis (available for *Broadband force noise* option **Band-limited**).

PSD representation

The desired representation of the user-defined spectrum (available for *Broadband force noise* option **PSD**). All related parameters correspond to the settings in chapter 10.17.2.2.

Tonal disturbance

- **By Imbalance:**
The tonal disturbance is modelled with only one harmonic whose magnitude is defined by the *Static imbalance coefficient* (U_s) in any [Mass·Length]-compatible unit.
- **By Harmonics:**
The tonal disturbance is modelled with a user-defined *Number of harmonics* N. For each harmonic, the following inputs need to be specified:
 - An Nx1 vector of *Amplitude coefficients* (C_r) in any [Mass·Length]-compatible unit
 - An Nx1 vector of dimensionless *Harmonic numbers* (h_r)

The specification of the *Axial mode* is completely optional and be enabled by a left-click on the adjacent checkbox. If enabled, the settings in the **Axial mode** panel are equivalent to the ones for the **Radial mode**. The tonal disturbance model configuration is inherited from the radial mode settings (i.e. **By Imbalance** or **By Harmonics**), the total number of harmonics M can however differ from the number N specified for the radial mode.

Background

Radial force model:

The radial (wheel x-y plane) disturbance forces acting on the spacecraft interface are modelled using the set of equations described below:

$$\begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} + \begin{bmatrix} c_r & 0 \\ 0 & c_r \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} k_r & 0 \\ 0 & k_r \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{F}_r$$

$$c_r = 4\pi\xi_r f_r m$$

$$k_r = m(2\pi f_r)^2$$

$$\mathbf{F}_{r,SC} = \begin{bmatrix} k_r & 0 \\ 0 & k_r \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

with:

- m flywheel mass
- \mathbf{F}_r the (x,y) excitation forces for the radial translation mode
- ξ_r damping coefficient of the radial translation mode
- f_r frequency of the radial translation mode
- $\mathbf{F}_{r,SC}$ resulting (x,y) disturbance forces at the spacecraft interface

Axial force model:

The axial (wheel z-axis) disturbance forces acting on the spacecraft interface are modelled using the set of equations described below:

$$m\ddot{z} + c_a \dot{z} + k_a z = F_a$$

$$c_a = 4\pi\xi_a f_a m$$

$$k_a = m(2\pi f_a)^2$$

$$F_{a,SC} = k_a z$$

with:

- m flywheel mass
- F_a the excitation forces for the axial translation mode
- ξ_a damping coefficient of the axial translation mode
- f_a frequency of the axial translation mode
- $F_{a,SC}$ resulting (z) disturbance force at the spacecraft interface

Excitation force model:

The overall excitation force comprises both (broadband) noise and tonal disturbances:

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_r \\ F_a \end{bmatrix} = \begin{bmatrix} F_r \\ F_r \\ F_a \end{bmatrix} = \begin{bmatrix} F_{r,tonal} + F_{r,noise} \\ F_{r,tonal} + F_{r,noise} \\ F_{a,tonal} + F_{a,noise} \end{bmatrix}$$

Tonal disturbance model:

The tonal force contributions to both the axial and translational force are realized as periodic 3D signal with amplitudes at frequencies of the corresponding harmonics. The amplitude A_k of the k-th harmonic ($k=1\dots N$, index for radial and axial mode omitted) and the corresponding frequency f_k are obtained from:

$$A_k = C_k \Omega^2$$

$$f_k = h_k \Omega$$

where Ω is the spin speed of the wheel, C_k is the amplitude coefficient of the k-th harmonic and h_k the harmonic number (i.e. the ratio of frequency of k-th harmonic to spin frequency of the wheel).

Alternatively, the radial disturbance can also be defined by the static imbalance coefficient U_s (i.e. considering only the first harmonic) resulting in an amplitude/frequency set:

$$A_1 = U_s \Omega^2$$

$$f_1 = \Omega$$

Different to the model realized in the PEET prototype, with the advanced statistical method also the phase shift between periodic signals plays a role and needs to be accounted for.

The phases of all periodic signals are entirely set up in the menu **Setup** → **Set Phase Relations**. As in particular for this model the phase shift between the radial axes x and y is exactly 90°, the phase for the y-axis cannot be set individually as it is uniquely determined by the phase of the x-axis signal. The arbitrary phase angle between different harmonics however, can be specified by the user.

10.19.2 Reaction Wheel (Torque)

The disturbance torque model includes a model for the rocking mode (in the x-y plane) only as axial disturbance are negligible according to [RD8].

Block Settings

The following settings need to be specified in the **Wheel properties** panel:

Inertia about spin axis / Inertia perpendicular to spin axis

The inertia about and perpendicular to the spin axis of the flywheel in any [Mass·Length²]-compatible unit.

Wheel speed distribution

The statistical distribution of the (positive) wheel speed. The options are identical to the ones for the *PES (Time-Constant)* block (see chapter 10.16 for more details), i.e. **Delta (No Distribution)**, **Uniform**, **Gaussian**, **Bimodal (Arcsine)**, **Rayleigh**, **Truncated Gaussian**.

Note: This description of the wheel speed represents an ensemble of “working points”, i.e. the wheel speed is considered constant within each working point and varies only between different working points (e.g. operational conditions, observations, etc.)

Wheel speed

Specifies the parameters for the distribution dependent on the selected *Wheel speed distribution*. Any [1/Time]-compatible *Unit* can be selected. As only positive wheel speeds are allowed, any distribution will be automatically truncated at zero.

The following settings need to be specified in the *Rotational mode* panel:

Rocking mode frequency

The fundamental frequency of the periodic disturbance in any [1/Time]-compatible unit.

Rocking mode damping

The dimensionless damping coefficient for the periodic disturbance of the rotational rocking mode.

Broadband torque noise

Determines the modelling approach for the torque noise contribution:

- **Standard deviation:** The noise is modelled as a zero-mean Gaussian time-random variable defined by its standard deviation (similar to a Gaussian time-random variable in the *PES (Time-Random)* block, see chapter 10.17.1.1).
- **Band-limited:** The noise is modelled as flat spectrum up to a given frequency similar to the band-limited white noise random process in the *PES (Time-Random)* block, see chapter 10.17.2.3).
- **PSD:** The noise is modelled following an arbitrary user-defined spectrum to the band-PSD option for a random process in the *PES (Time-Random)* block, see chapter 10.17.2.2).

Noise standard deviation

The standard deviation of the noise on the rotational axis (available for *Broadband torque noise* options **Standard deviation** and **Band-limited**).

Noise bandwidth

The bandwidth (largest frequency) of the noise on the rotational axis (available for *Broadband torque noise* option **Band-limited**).

PSD representation

The desired representation of the user-defined spectrum (available for *Broadband torque noise* option **PSD**). All related parameters correspond to the settings in chapter 10.17.2.2.

Tonal disturbance

- **By Imbalance:**
The tonal disturbance is modelled with only one harmonic whose magnitude is defined by the *Dynamic imbalance coefficient* (U_d) in any [Mass·Length²]-compatible unit.
- **By Harmonics:**
The tonal disturbance is modelled with a user-defined *Number of harmonics* N. For each harmonic, the following inputs need to be specified:
 - An Nx1 vector of *Amplitude coefficients* (C_i) in any [Mass·Length²]-compatible unit

- An Nx1 vector of dimensionless *Harmonic numbers* (h_i)

Background

Rocking mode model:

The disturbance torques due to the rocking mode (wheel x-y plane) which act on the spacecraft interface are modelled using the set of equations described below [RD8]:

$$\begin{bmatrix} I_{rr} & 0 \\ 0 & I_{rr} \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} c_{rock} & \Omega I_{zz} \\ -\Omega I_{zz} & c_{rock} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} k_{rock} & 0 \\ 0 & k_{rock} \end{bmatrix} \begin{bmatrix} \phi \\ \theta \end{bmatrix} = \mathbf{T}_{rock}$$

$$c_{rock} = 4\pi\xi_{rock}f_{rock}I_{rr}$$

$$k_{rock} = I_{rr}(2\pi f_{rock})^2$$

$$\mathbf{T}_{rock,SC} = \begin{bmatrix} k_{rock} & 0 \\ 0 & k_{rock} \end{bmatrix} \begin{bmatrix} \phi \\ \theta \end{bmatrix}$$

with:

- I_{rr} flywheel inertia perpendicular to spin axis
- I_{zz} flywheel inertia about spin axis
- \mathbf{T}_{rock} (x,y) excitation torques for the rocking mode
- ξ_{rock} damping coefficient of the rocking translation mode
- f_{rock} frequency of the rocking mode
- $\mathbf{T}_{rock,SC}$ resulting (x,y) disturbance torques at the spacecraft interface

Excitation torque model:

According to [RD8], the overall excitation torque comprises (broadband) noise and tonal disturbances for the rocking mode and negligible disturbance torques around the z-axis.

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_{rock} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{rock} \\ \mathbf{T}_{rock} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{rock,tonal} + \mathbf{T}_{rock,noise} \\ \mathbf{T}_{rock,tonal} + \mathbf{T}_{rock,noise} \\ 0 \end{bmatrix}$$

Tonal disturbance:

The tonal torque contribution from the rocking mode is realized as a periodic 3D signal with amplitudes at frequencies of the corresponding harmonics. The amplitude A_k of the k-th harmonic ($k=1\dots N$, index for rocking mode omitted) and the corresponding frequency f_k are obtained from:

$$A_k = C_k \Omega^2$$

$$f_k = h_k \Omega$$

where Ω is the spin speed of the wheel, C_k is the amplitude coefficient of the k-th harmonic and h_k the harmonic number (i.e. the ratio of frequency of k-th harmonic to spin frequency of the wheel).

Alternatively, the rocking mode can also be defined by the dynamic imbalance coefficient U_d (i.e. considering only the first harmonic) resulting in an amplitude/frequency set:

$$A_1 = U_d \Omega^2$$

$$f_1 = \Omega$$

Different to the model realized in the PEET prototype, with the advanced statistical method also the phase shift between periodic signals plays a role and needs to be accounted for.

The phases of all periodic signals are entirely set up in the menu **Setup** → **Set Phase Relations**. As in particular for this model the phase shift between the radial axes x and y is exactly 90°, the phase for the y-axis cannot be set individually as it is uniquely determined by the phase of the x-axis signal. The arbitrary phase angle between different harmonics however, can be specified by the user.

10.20 Rigid Plant

Input/Output

The output of this block represents the 3D attitude response of a rigid body to a torque input. Thus, the unit of the input signal must be in a torque compatible unit, i.e. [Force·Length]. For the output signal, any [Angle]-compatible unit can be selected. Only 3D input signals are supported.

Block Settings

The only setting which needs to be specified in the **Plant parameters** panel is the 3x3 inertia of the rigid body given in any [Mass ·Length²]-compatible unit.

Background

The block realizes an ideal plant model following the equation

$$\Theta \dot{\omega} = \mathbf{N}$$

with:

- Θ body inertia matrix (3x3)
- ω vector of body angular rates (3x1), integration of this quantity gives the Block output
- \mathbf{N} vector of torques acting on the body (3x1) as block input

10.21 Star Tracker Noise

Output

The output of this block represents the bias-free 3D noise spectrum of a star tracker considering field of view and pixel noise contributions. Any [Angle]-compatible *Output unit* can be defined. The z-axis is considered as boresight axis, x- and y-axes correspond the cross-axes with equal noise contributions.

Block Settings

The following settings need to be specified in the *General parameters* panel:

Detector size (pixels)

The overall size in pixels of the active pixel sensor (APS) matrix/detector.

Sensor field of view

The overall field of view (i.e. not the half-width w.r.t. the boresight-axis) of the sensor camera head in any [Angle]-compatible unit.

Spacecraft angular velocity

The magnitude of the average spacecraft angular velocity in any [Angle/Time]-compatible unit.

Note: If the average angular velocity is exactly zero, the model output is also zero. In case of e.g. an inertial pointing scenario this value should be set according to the expected rate control performance to a small, but non-zero value.

Average number of tracked stars

The dimensionless average number of stars tracked by sensor within its field of view.

Alpha

The angle between the star image direction of motion on the APS matrix and the APS reference axis in any [Angle]-compatible unit.

Note: If $\cos(\text{Alpha})$ is exactly zero, the model output is also zero.

Beta

The angle between the sensor boresight and the spacecraft rotation axis in any [Angle]-compatible unit.

Note: If $\sin(\text{Beta})$ is exactly zero, the model output is also zero.

The following settings need to be specified in the *Field of view noise* panel:

Noise standard deviation

The standard deviation of the FOV noise for all axes given in any [Angle]-compatible unit.

The following settings need to be specified in the *Pixel noise* panel:

Size of centroiding window

The scalar size of the centroiding window which is typically around 3 pixels.

Noise standard deviation (boresight)

The standard deviation of the pixel noise around the boresight axis (z) given in any [Angle]-compatible unit.

Noise standard deviation (cross-axes)

The standard deviation of the pixel noise around the cross-axes (x,y) given in any [Angle]-compatible unit.

2nd-order filter damping coefficient

The dimensionless scalar damping coefficient used for the second order noise shaping filter transfer function for the pixel noise.

Background

The underlying model is based on [RD11]. The PSD of the field of view noise spectrum with standard deviation n_{FOV} is represented by the following first order transfer function:

$$P_{FOV} = \frac{\sqrt{T_{FOV}}}{1 + s \frac{T_{FOV}}{2}} n_{FOV}$$

The correlation time T_{FOV} is assumed to be proportional to the inverse of the velocity v_{star} (pixels/sec) with which the star image moves on the sensor pixel matrix with N pixels:

$$T_{FOV} = \frac{N}{v_{star} \sqrt{N_{stars}}}$$

The star velocity itself can be linked to the average spacecraft angular velocity ω_{SC} :

$$v_{star} = \omega_{SC} \frac{N}{FOV} \sin \beta \cos \alpha$$

where FOV is the sensor field of view, β is the angle between the sensor boresight and the spacecraft rotation axis and α is the angle between the star image direction of motion on the detector matrix and the detector reference axis.

Note: If any of the factors above becomes zero, the complete model output is zero as well.

The PSD of the pixel noise with standard deviation n_{pixel} is modelled using a 2nd-order filter as (again, the additional factor of $\sqrt{2}$ compared to [RD11] is necessary to realize for conversion to a one-sided spectrum):

$$P_{pixel} = \frac{\omega_0^2 \sqrt{T_{pixel}}}{s^2 + 2\xi\omega_0 s + \omega_0^2} n_{pixel}$$

where ξ is the filter damping coefficient and the characteristic frequency ω_0 is given by:

$$\omega_0 = \frac{4\xi}{T_{pixel}}$$

The correlation time T_{pixel} is again assumed to be proportional to the inverse of the velocity v_{star} :

$$T_{pixel} = \frac{N_{pixels}}{v_{star}}$$

where N_{pixels} is the size of the centroiding window.

The overall noise spectrum is then obtained by summation of the two contributions. This implies a summation on the level of power spectra, i.e. both expressions are squared before the summation (* denotes the complex conjugate transpose):

$$G_{STR} = G_{FOV} + G_{pixel} = (P_{FOV}P_{FOV}^*) + (P_{pixel}P_{pixel}^*)$$

10.22 Static System

Input/Output

The output of this block represents the input signal multiplied with a constant system matrix. By default, the units of the input/output are inherited from the input signal, but also any unit compatible to the input signal and an arbitrary output unit can be specified. Generally, only system dimensions matching to the input signal are supported. In case a 1D input signal is present, and the *System dimension* is 3D (or vice versa), an error is thrown.

Note: The unit of the model parameters is derived from the selected units, i.e. if the input unit is [A] and the output unit is [B], it is assumed that the system model parameters are provided in a unit [B/A].

Block Settings

The only setting which needs to be specified in the *Model parameters* panel are the coefficients of 3x3 system matrix (or only one scalar value in the 1D case).

Background

The system matrix is used to pre-multiply the input signal, i.e. Output = Matrix · Input and can be used to scale the input signal or to introduce a coupling between axes of the input.

10.23 Summation

Input/Output

The output of this block represents the sum or difference of the input signals dependent on the selected convention. All input signals must have a compatible unit. By default, the *Inherit output unit from first input port* option is enabled, but also any other compatible unit to the input signals can be specified for the output. Only identical input signal dimensions are supported.

Block Settings

The only setting which needs to be specified is the *List of signs* which determines the rule to be applied for the input signal, e.g.

- “+++”: sum all three input signals
- “+-+”: subtract input 2 from the sum of inputs 1 and 2
- “-+” : subtract the first input from the second

The number of signs also determines the number of inputs to the block.

Background

For all error signal parts which are represented by samples (i.e. all but PSD), the summation/subtraction is directly applied to the samples. For the PSD contributions, also the cross-spectra are taken into account.

Note: Subtracting two PSDs with equal magnitude does not necessarily result in a zero output spectrum. This is only the case if the two signals are fully coherent (refer e.g. to [RD7]).

10.24 Total Error

Input

The *Total Error* block supports 1D and 3D input signals. It has no output port and serves as “endpoint” of the budget tree, i.e. its highest level. The specified unit is only relevant for display purposes. By default, the *Inherit unit from input* option is enabled, but it also possible to specify any other evaluation unit compatible to the input unit.

Note: Only one *Total Error* block is allowed in a scenario.

Tip: To evaluate signals on lower level than the final error, an arbitrary number of *PEC* blocks can be used at any stage of the budget tree.

Block Settings

The *Total Error* block has no specific block parameters. Requirement values which shall be associated with this block are completely defined via the menu *Setup* → *Scenario Definition* in the *Requirement Specification* tab (see chapter 6.4.2).

Background

The *Total Error* block has no effect on the input signal itself. It is only used to evaluate the input error signal with respect to the given requirement (and compare it to the specified requirement value(s) if provided).

The content of the different parts of the input error signal (CRV, RV, drift, periodic signal and random process part) is summed according to AST-4 of [AD2]. In case a random process error contribution is present, first its variance is computed within the user-defined evaluation bandwidth.

Tip: The overall error is computed per axis (x,y,z) and with respect to the user defined LoS axis. Note that the latter is the only special feature that links the block really to pointing. Disregarding the LoS error, this block (and PEET) could be used to compute any kind of 3-axis budget (i.e. PEET could generally be understood as "Performance Error Engineering Tool" rather than a "Pointing Error Engineering Tool" only).

10.25 Total Error (Position)

Input

The *Total Error (Position)* supports only 3D input signals. It has one position error and at least one attitude error input while the number of further attitude error inputs depends on the block settings. The specified units are only relevant for display purposes. By default, the *Inherit units from input* option is enabled, but it is also possible to specify any other [Angle]- and [Length]-compatible units for evaluation.

As the standard *Total Error* block, this block serves and "endpoint" of the budget tree (i.e. its highest level). However, it allows the computation of a position/displacement error budget which is the result of "pure" 3-axis position errors and 3-axis attitude errors which couple into equivalent position errors due to dedicated "lever arms" (e.g. as it is the case for formation flying missions).

Block Settings

The following settings need to be specified in the *Attitude coupling* panel:

Number of attitude contributors

Defines the number N of different attitude contributors that couple into a position error. This also determines the number of inputs to the block (1+ N_{att} in total).

Attitude coupling vector(s)

Defines the N_{att} x3 matrix that contains the related coupling vectors for the attitude inputs in any [Length]-compatible unit. The i-th row of the matrix represents the coupling vector for the i-th attitude input, which is equivalent to the (i+1)-th block input.

Background

The implemented model in the PEET prototype for an "exact" position budget was based on Eq.5 in [RD6]:

$$\begin{aligned} \mu_{\text{tot},x} &= \mu_{\text{pos},x} - \sum_{i=1}^{N_{\text{att}}} (y_i \mu_{\text{att},z,i} - z_i \mu_{\text{att},y,i}) & \sigma_{\text{tot},x}^2 &= \sigma_{\text{pos},x}^2 + \sum_{i=1}^{N_{\text{att}}} (y_i^2 \sigma_{\text{att},z,i}^2 + z_i^2 \sigma_{\text{att},y,i}^2) \\ \mu_{\text{tot},y} &= \mu_{\text{pos},y} - \sum_{i=1}^{N_{\text{att}}} (z_i \mu_{\text{att},x,i} - x_i \mu_{\text{att},z,i}) & \sigma_{\text{tot},y}^2 &= \sigma_{\text{pos},y}^2 + \sum_{i=1}^{N_{\text{att}}} (z_i^2 \sigma_{\text{att},x,i}^2 + x_i^2 \sigma_{\text{att},z,i}^2) \\ \mu_{\text{tot},z} &= \mu_{\text{pos},z} - \sum_{i=1}^{N_{\text{att}}} (x_i \mu_{\text{att},y,i} - y_i \mu_{\text{att},x,i}) & \sigma_{\text{tot},z}^2 &= \sigma_{\text{pos},z}^2 + \sum_{i=1}^{N_{\text{att}}} (x_i^2 \sigma_{\text{att},y,i}^2 + y_i^2 \sigma_{\text{att},x,i}^2) \end{aligned}$$

with (axis index omitted):

- μ_{tot} total mean of resulting displacement error
- σ_{tot} total standard deviation of resulting displacement error
- μ_{pos} overall mean of "pure" position error contributors
- σ_{pos} overall standard deviation of "pure" position error contributors
- N_{att} number of attitude error couplings to position
- $\mu_{\text{att},i}$ mean of i-th attitude error
- $\sigma_{\text{att},i}$ standard deviation of i-th attitude error

- x_i, y_i, z_i components of coupling vector of i-th attitude error

Different to Eq. 5 in [RD6], there is no summation over different position error contributors. The summation of these contributors has to be realized using standard *Summation* blocks in the PEET scenario.

As the direction of individual contributors (sign relations) are not exactly known using means and (always positive) standard deviations only, the PEET prototype alternatively offered of a more conservative approach using a “worst case” option. Here, the total mean is computed using absolute values according to the following equation:

$$\begin{aligned}\mu_{tot,x} &= \left| \mu_{pos,x} + \sum_{i=1}^{N_{att}} (y_i \mu_{att,z,i} - z_i \mu_{att,y,i}) \right| \\ \mu_{tot,y} &= \left| \mu_{pos,y} + \sum_{i=1}^{N_{att}} (z_i \mu_{att,x,i} - x_i \mu_{att,z,i}) \right| \\ \mu_{tot,z} &= \left| \mu_{pos,z} + \sum_{i=1}^{N_{att}} (x_i \mu_{att,y,i} - y_i \mu_{att,x,i}) \right|\end{aligned}$$

Above equations rely on the simplified statistical method. With the introduction of the advanced statistical method, the decomposition in mean and variance values is no longer necessary and the position error can directly be expressed precisely without the need for a further distinction (“Exact” or “Worst case”):

$$\begin{aligned}e_{tot,x} &= e_{pos,x} - \sum_{i=1}^{N_{att}} (y_i e_{att,z,i} - z_i e_{att,y,i}) \\ e_{tot,y} &= e_{pos,y} - \sum_{i=1}^{N_{att}} (z_i e_{att,x,i} - x_i e_{att,z,i}) \\ e_{tot,z} &= e_{pos,z} - \sum_{i=1}^{N_{att}} (x_i e_{att,y,i} - y_i e_{att,x,i})\end{aligned}$$

10.26 Block Overview

Table 6: Block overview with supported I/O unit (groups)

| Block | Category | Dimensions | Input unit | Output unit |
|---------------------------|----------|------------|------------|-----------------------------|
| Accelerometer Noise | PES | 1D and 3D | - | [Length/Time ²] |
| Camera Range Noise | PES | 3D | - | [Length] |
| Container | Basic | - | - | - |
| Coordinate Transformation | Static | 3D | Any | Input unit |
| Dynamic System | Dynamic | 1D or 3D | Any | Any |

| | | | | |
|-------------------------|------------|-----------|--------------------------------|---------------------------------|
| Feedback System | Basic | - | - | |
| Flexible Plant | Dynamic | 3D | [Force · Length] | [Angle] |
| General Periodic Error | PES | 1D or 3D | - | Any |
| GPS | PES | 3D | - | [Length] or [Length/Time] |
| Gyro Rate Noise | PES | 1D or 3D | - | [Angle] |
| Gyro-Stellar Estimator | Dynamic | 3D | [Angle] and [Angle/Time] | Input units |
| Input PEC | PES | 1D or 3D | Any | Any |
| Input Port | Basic | Inherited | - | Inherited |
| Mapping | Basic | 1D → 3D | - | Any |
| Output Port | Basic | Inherited | Inherited | - |
| PEC | Evaluation | 1D or 3D | Inherited | Input unit |
| PES (Time-Constant) | PES | 1D or 3D | - | Any |
| PES (Time-Random) | PES | 1D or 3D | - | Any |
| PID Controller | Dynamic | 1D or 3D | Any | Any |
| Reaction Wheel (Force) | PES | 3D | - | [Force] |
| Reaction Wheel (Torque) | PES | 3D | - | [Force · Length] |
| Rigid Plant | Dynamic | 3D | [Force · Length] | [Angle] |
| Star Tracker Noise | PES | 3D | - | [Angle] |
| Static System | Static | 1D or 3D | Any | Any |
| Summation | Basic | Inherited | Inherited | Input unit |
| Total Error | Evaluation | 1D or 3D | Any | (Input unit) |
| Total Error (Position) | Evaluation | 3D | [Length] and [Angle] | [Length] |

11 Troubleshooting

If you encounter problems installing PEET or a license server, which cannot be solved using this troubleshooting section, please do not hesitate to contact user-feedback@astos.de.

11.1 Incompatibilities

Currently, there is only one known incompatibility known for PEET.

It is related to the MATLAB Text Analytics Toolbox (introduced with MATLAB 2017b) which prevents a correct generation of Excel reports with the tool. Both the toolbox and PEET use `apache.poi` while the version used for the Text Analytics Toolbox seems to be incomplete [RD14]. By default, this internal version is preferred by MATLAB (causing the issue when called from the PEET report generation algorithms).

The following workarounds exist and were tested successfully:

1) Uninstall the Text Analytics Toolbox if not required:

- In the MATLAB main window, navigate to **Add-Ons** → **Manage Add-Ons**
- Click the options button on the right in the **Text Analytics Toolbox** panel and choose **Uninstall**.
- Execute the script `installPEET.m` again (as the de-installation overwrites the `javaclasspath.txt` in MATLAB which contains the links to the JAVA classes required by PEET).

2) Force MATLAB to use PEET java classes first (though not reported, this could cause issues with other MATLAB toolboxes in general, so please use with care):

- Navigate to the folder `[MatlabInstallDir]\toolbox\local\`
- Open the file `classpath.txt` (MATLAB should be closed and administrator rights are required).
- Search for the line with the comment `# PEET lib`
- Enter the command `<before>` in the line above this comment.
- Save the file.

11.2 License Activation

Unsuccessful license activation:

In case the **Activation Process** is not successful, please check the following possible causes:

- Make sure to insert the proper license activation data in the proper place and not to mix up the following pieces of license activation data: "**Entitlement ID**", "**User ID**", "**Access Code**" and "**License SN**". For the license activation using either an **Internet connection** or an **Offline Activation**, an Entitlement ID and an Access Code

are required to start the activation process. To complete an Offline Activation process also the User ID is required. (see chapter 4.3).

- Make sure that when copying and pasting the license activation data, no additional white spaces (empty space, tabs, etc.) or other characters are entered in the input fields of the license activation wizard or on the command line. If the license activation process fails and you are sure not to have mixed up the items of license activation data, then try to enter the data by hand. In this case make sure not to mix up "0" and "O" as well as upper and lower cases.

FlexNet Licensing Service not installed correctly in case of a node-locked license:

In some rare cases it might happen, that the FlexNet Licensing Service is not correctly installed during the installation process. In this case node-locked licenses are not accessible. If node-locked licenses are active, but PEET is generating a license error, this service needs to be reinstalled manually. Please follow these steps:

- As administrator, open a *Command Prompt* window.
- Navigate to the `bin` folder of the installation.
- Run `installanchorservice.exe astoslic PEET`. This will reinstall the *FlexNet Licensing Service*.

Afterwards, run PEET again. This time it should be possible to access the node-locked license.

Flexlm activation progress fails:

In case an error appears after starting `svractutil.exe -a` (see step 5 in chapter 4.3.2) with a message similar to

```
"The program can't start because MSVCR120.dll
is missing from your computer.
```

```
Try reinstalling the program to fix this problem."
```

then most probably the Visual C++ *Redistributable Packages for Visual Studio 2013* are not installed on the system. These packages can be directly downloaded from the Microsoft web page: <https://www.microsoft.com/en-us/download/details.aspx?id=40784>.

Flexlm license server launch fails

In case an error appears after starting `lmadmin.exe` (see step 5 in chapter 4.3.2) with a message similar to

```
"The program can't start because MSVCR90.dll
is missing from your computer.
```

```
Try reinstalling the program to fix this problem."
```

then most probably the Visual C++ *Redistributable Packages for Visual Studio 2010* are not installed on the system. To install these packages, either navigate to the `#flexlm_folder#\lmadmin` directory and run `vcredist_x86.exe`. or directly download from the Microsoft web page: <https://www.microsoft.com/en-US/download/details.aspx?id=5555>

The 'Deactivate' button in the License Management tab is inactive

In case there is need to deactivate an active license which is listed in the [License Management](#) tab of the [License Manager](#), please click on an entry below the header row of the table ([Fulfillment ID](#), [Features](#), etc) in order to select the fulfillment ID in the table. Once a fulfillment ID is selected, the color of the selected table entry changes and then the [Deactivate](#) button gets available.

11.3 Starting PEET in MATLAB

Note: In case of a stand-alone license based on trusted storage (no USB dongle), the PEET software has to be used directly on the computer where it is installed. Using PEET via Microsoft Windows Remote Desktop (RDP) connection is not supported. In this case, an error message "No feature PEET version X.Y found!" is shown.

GUI does not open:

Undefined variable error

In case an error message `Undefined function or variable 'peet'.` is displayed, then most likely cause is that the installation script `installPEET.m` in the PEET root folder has not been executed in this MATLAB instance.

Please rerun the script and follow the on-screen instructions.

Error dialog pops up

In case an error dialog with message "No feature PEET version 1.X found" shows up, this indicates that a valid license for the tool is missing.

After confirming the message with the [OK](#) button, the GUI opens with all important menu and toolbar items disabled. Please use the [Info → License Manager...](#) menu to check your license status and activate a license as described in chapter 4.3 if necessary. After having activated a license, please restart PEET to make the activation take effect (i.e. to unlock all menu actions and toolbar items).

GUI opens:

Undefined variable error


In case an error message `Undefined variable "engine" or class "engine.Analysis"` is displayed in the MATLAB command window when the GUI opens, the most likely cause is that the path to the required MATLAB classes has not been properly set in this MATLAB instance. To solve this issue, please

- rerun the script `installPEET.m` and follow the on-screen instructions and/or
- make sure that the paths `#peetroot#\bin` and `#peetroot#\lib\class` are set in MATLAB ([Home](#) tab, [Set Path](#) in the MATLAB main window) and these directories also exist.

Error dialog pops up

In case the GUI opens together with a `Database Read Error`, the most likely cause is that a PEET prototype version (<V0.6) was already installed for this MATLAB instance which could not be detected by the installation script.

- 1) Identify prototype 'artefacts'

The easiest way to check this condition is to determine if the PEET logo () is missing on the top left of the GUI main window. If this is the case, then MATLAB's JAVA class paths still point to JAVA classes of the software prototype.

The MATLAB command `javaclasspath` explicitly reveals any classes which are related to a prototype installation in any `#prototyperoot#` folder.

- `#prototyperoot#\lib\peet.jar`
- `#prototyperoot#\lib\piccolo2d-core-1.3.1.jar`
- `#prototyperoot#\lib\piccolo2d-extras-1.3.1.jar`
- `#prototyperoot#\lib\jxl.jar`

2) Remove prototype 'artefacts'

Such installation "artefacts" need be removed manually from the class paths. The location of the file to be modified depends on the type of MATLAB and prototype installation and on the user privileges on the current machine. It is most likely found in one of the following locations:

- `#matlabroot#\toolbox\local\classpath.txt`
- `#userdir#\javaclasspath.txt`
- `#userdir#\classpath.txt`
- `#prefdir#\javaclasspath.txt`
- `#prefdir#\classpath.txt`

where `#matlabroot#` is the root folder of the current MATLAB instance (can be identified via MATLAB command `matlabroot`), `#userdir#` is the location of the default MATLAB user work folder (can be identified via MATLAB command `userpath`) and `#pref#` is the location of the MATLAB preferences folder (can be identified via MATLAB command `prefdir`).

3) Verify Java class paths

To make sure that all V1.0 classes are properly added to MATLAB's static class path, please make sure that the following class path are set (using the `javaclasspath` command) which should always be the case after having executed the `installPEET.m`.

- `#peetroot#\lib\peet.jar`
- `#peetroot#\lib\piccolo2d-core-1.3.1.jar`
- `#peetroot#\lib\piccolo2d-extras-1.3.1.jar`
- `#peetroot#\lib\jxl.jar`
- `#peetroot#\lib\poi-3.13-20150929.jar`
- `#peetroot#\lib\poi-ooxml-3.13-20150929.jar`
- `#peetroot#\lib\poi-ooxml-schemas-3.13-20150929.jar`
- `#peetroot#\lib\xmlbeans-2.6.0.jar`

Therefore, the absolute paths to all these jar files need be added to MATLAB's static class path:

Alternatively, the absolute paths can also be set manually by modifying the `classpath.txt` file located in the in the toolbox\local folder of your MATLAB installation (doing this, PEET gets installed for all MATLAB users). If this file cannot be edited or should be installed for the current user only, it is also possible to create a local `javaclasspath.txt` file. Simply add the absolute paths to the jar files in this text file located in the MATLAB preferences directory (type "prefdir" in the MATLAB command window to identify it). In case this file does not yet exist, simply create a new text with the given name. MATLAB needs to be restarted if running to make these changes applied.

11.4 Other Support

There are two official contact email addresses available to PEET users:

- service@astos.de: this address should be used for general support questions related to an efficient way of working with PEET.
- user-feedback@astos.de: this address should be used to report bugs, suggest improvements and provide generic feedback.

Please do not hesitate to contact us.

The level of support that we can provide depends on the input that is provided to us. Ideally, please always include the following information in a support request:

- a short textual description of the issue
- the operating system on which PEET is installed
- the MATLAB version and properties (e.g. by providing the output of the `ver` command in the MATLAB Command Window)
- the PEET version (e.g. by providing the output of `engine.Analysis.version` in the MATLAB Command Window) copy (or screenshot) of the [Execution Log](#) window output in the PEET [System Editor](#)
- a copy of the `.peet` scenario folder (ideally zipped) or the scenario XML file (`PointingSystemDefinition.xml`) in the `\input` folder of the test case. This allows us to directly execute the scenario and reproduce a potential bug or issue.

Note: The last bullet may be related to confidential information. Please note, that any of our support is treated as confidential. Users obtain a non-disclosure agreement any time requested.