

## PEET V1.0: THE STATE-OF-THE-ART POINTING AND PERFORMANCE ERROR ENGINEERING TOOL FOR SPACE MISSIONS

Marc Hirth<sup>(1)</sup>, Haifeng Su<sup>(1)</sup>, Thomas Ott<sup>(2)</sup>, Massimo Casasco<sup>(3)</sup>, Guillermo Ortega<sup>(3)</sup>

<sup>(1)</sup> Astos Solutions GmbH, Meitnerstr. 8, 70563 Stuttgart, Germany, Email: marc.hirth@astos.de

<sup>(1)</sup> Astos Solutions GmbH, Meitnerstr. 8, 70563 Stuttgart, Germany, Email: haifeng.su@astos.de

<sup>(2)</sup> Airbus Defence and Space, 88090 Immenstaad, Germany, Email: thomas.ott@airbus.com

<sup>(3)</sup> ESA, ESTEC, Keplerlaan 1, 2201 AZ Noordwijk, The Netherlands, Massimo.Casasco@esa.int

<sup>(3)</sup> ESA, ESTEC, Keplerlaan 1, 2201 AZ Noordwijk, The Netherlands, Guillermo.Ortega@esa.int

### ABSTRACT

Future space missions tend to rely on increasingly demanding pointing performance and/or are driven by the need for a cost-efficient design process.

To avoid potential mission- or cost-critical changes late in the design, having at hand a clear pointing error engineering methodology to systematically assess achievable performances for various requirement categories already in early project phases is crucial.

The Pointing Error Engineering Tool (PEET) is a software tool which is intended to support systems and AOCS engineers in the setup and calculation of such performance error budgets with a special focus on spacecraft pointing and relative-positioning. Its computational core is based on standardized rules established in the ECSS Control Performance Standard and on the methodology described in the ESA Pointing Error Engineering Handbook.

This paper wraps up the translation of this methodology into a software tool and provides an overview of the realized features. Finally, the paper comments on the benefits of PEET and the developed concepts from an ESA and industrial user point of view.

## 1 INTRODUCTION

The ECSS Control Performance Standard E-ST-60-10C [1] and the ESA Pointing Error Engineering Handbook (PEEH) [2] form the basis of the pointing error engineering process in ESA projects. These documents provide both the mathematical elements and a systematic engineering methodology to set up pointing requirements, evaluate budgets and to assess pointing performances. In 2013, a software prototype called Pointing Error Engineering Tool (PEET) was developed and released under an ESA contract with the aim to complement and to ease the application of this methodology in daily work.

With the feedback received from users and the lessons learned during the prototyping phase, a second development phase was initiated under a contract of and funded by the European Space Agency to turn the prototype into a consolidated software tool which is made available free of charge within ESA member states.

The basic information about the methodology and tool implementation presented in this paper has already been outlined in [3], but is updated herein to represent the final state of the tool. Further, the paper covers aspects of tool verification and validation and its current application in projects.

## 2 PEEH METHODOLOGY IN THE TOOL

The methodology described in [2] provides explicit guidelines on how to define pointing requirements and on how to define and evaluate pointing errors from source to the final error contribution using dedicated analysis steps (ASTs). The reflection of these steps in the software tool is described in the following.

### 2.1 Requirement Specification Parameters (“AST-0”)

Dependent on its nature (statistical or spectral), [2] provides sets of specification parameters to unambiguously define pointing requirements which are completely reflected in the tool. For statistical requirements, the following parameters are used:

- Required maximum error value (defined as deviation per axis or half-cone error)
- A related level of confidence that above-mentioned error values are not exceeded
- Metrics for time-windowed errors in terms of performance/knowledge indices as defined in [1] and related parameters (e.g. window time)
- The statistical interpretation that applies to the evaluation

The statistical interpretation concept (as defined in [1]) describes how resulting errors shall be treated in the evaluation concerning the domains “time” and “ensemble” (e.g. different satellites, observations, etc.). As depicted in Figure 1, “ensemble” interpretation accounts for the statistics of the worst-case errors in each realization no matter when they occur in time. “Temporal” interpretation accounts for the temporal statistics of the realization that contains the worst-case error value and “mixed” interpretation finally accounts for the entire statistics over both time and ensemble realizations.

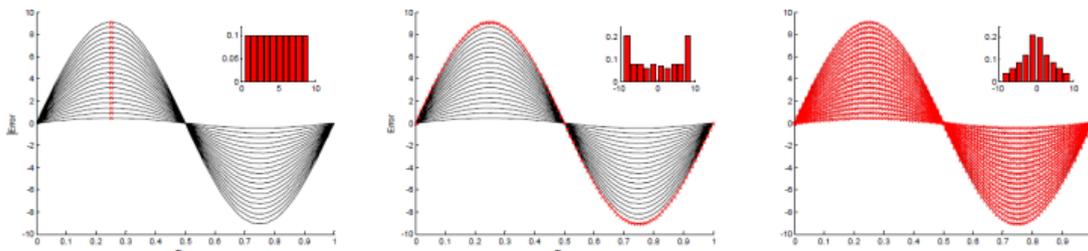


Figure 1: Statistical interpretations [1]: “ensemble” (left), “temporal”(centre), “mixed” (right)

This concept has been further generalised during the PEET development phase to obtain a more flexible definition of requirements (see also section 4.2).

For spectral requirements, only the metric for time-windowed errors is used. All other specification parameters above are replaced by a spectral requirement function that describes the upper bound of the power spectral density of the error over frequency (within a given bandwidth).

### 2.2 Error Source Characterization (AST-1)

This analysis step aims for identifying all potential error sources that affect the budget and classifying these sources on the basis of their characteristic properties (e.g. time-constant vs. time-random, random vs. deterministic, random variable vs. random process).

The first part obviously remains an engineering task as it differs in each application case and neither [2] nor the tool can provide explicit support. For the classification however, PEET covers all the

error signal classes listed in [2] which can be modeled as:

- Time-constant random variables, i.e. biases which can be further subject to an ensemble-randomness (e.g. alignments within a certain bound)
- Time-random variables, i.e. errors that are random over time; the parameters of such sources can further be subject to an ensemble-randomness (e.g. a noise whose upper bound is dependent on temperature conditions that are considered constant within an observation but may differ between observations)
- Random processes, i.e. (stationary, ergodic) noise sources that are completely defined by their power spectral density or by a given variance and bandwidth
- Periodic errors, in particular sinusoidal disturbances whose amplitudes may again be subject to an ensemble-randomness.
- Drift errors, i.e. linearly increasing errors which can be reset after a certain amount of time (e.g. due to calibration); the linear slope may again be distributed over the ensemble of realizations

All ensemble-random properties of the error source classes can either be set as discrete or be selected from common statistical distributions: uniform, (truncated) Gaussian, arcsine, Rayleigh. Furthermore, also arbitrary probability density functions (PDFs) can be defined by the user.

For time-random variables, the temporal distributions are restricted to be uniform and Gaussian, as only for these distributions standardized rules are provided in [1] on how the pointing error metrics in AST-3 need to be applied.

### 2.3 Transfer Analysis (AST-2)

The transfer analysis determines how initial error sources (e.g. a noise or misalignment of a sensor in its mechanical frame) affect the figure of merit defined by the requirement (e.g. the pointing error of a payload in the pointing reference frame). The “route” to this final pointing error contribution may underlie various transfers such as coordinate transformations or closed-loop transfers. The situation for a general transfer analysis setup over multiple subsystem levels is sketched in Figure 2.

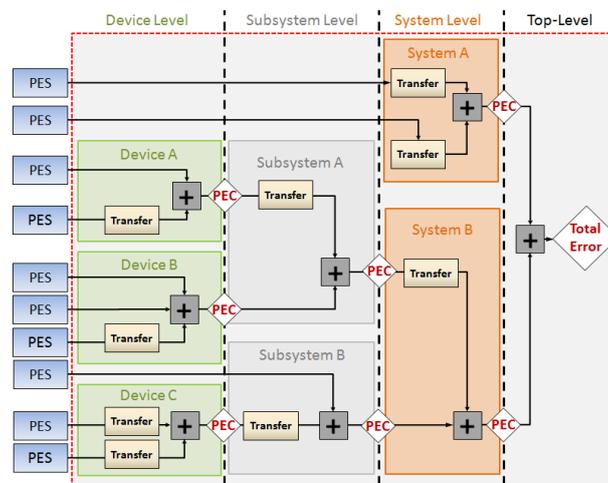


Figure 2: Transfer analysis from source to total error

As for the error source classification, no distinct rules can be provided for the identification of

relevant transfer models for a certain application case. However, the tool provides a selection of generic (i.e. not even restricted to pointing) static and dynamic system models as well as dedicated parametric models for certain sensor and actuator error characteristics.

The dynamic system models available are based on linear time-invariant models with multiple inputs/outputs to account for cross-couplings between different axes. Feedback loops can be realized by either directly specifying a closed-loop transfer function or by setting up the loop explicitly with a dedicated model block.

The transfer rules for the dynamic system transfer depend on the class of the error signal. Periodic and random process errors are accurately transferred using the system gain at the relevant frequency or the entire frequency response respectively [4]. Time-constant random variables are scaled using the system's DC gain. For time-random variables, where no frequency domain information is present by definition, the system's  $H_\infty$  norm is used as conservative upper bound.

## 2.4 Error Index Contribution (AST-3)

This step analyses the impact of the time-windowed pointing error metric as one part of the requirement specification parameters. The definition of all available metrics is depicted in Figure 3 with the typical abbreviations as defined in [1] and [2].

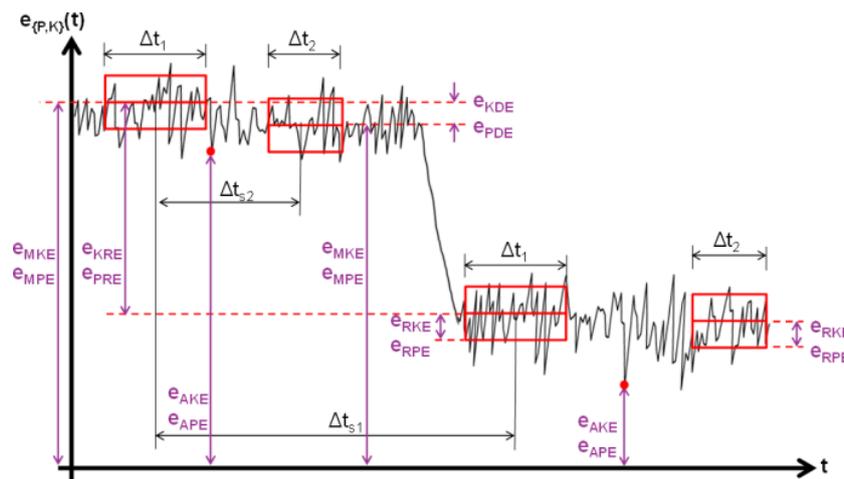


Figure 3: Time-windowed error contributions for different pointing metrics

As an example: the relative performance error (RPE) describes the variation of an error with respect to the mean error in a given time window  $\Delta t$ . If this metric is applied to a periodic signal with a very large period compared to the window time, the variation within the window is much smaller than the signal amplitude. Conversely, for a high frequency periodic signal, the entire amplitude variation shows up within one time window.

For random variables, [1] provides explicit tables with rules on how the various metrics affect the effective errors which are completely integrated in the tool. For random process and periodic error signals, an equivalent representation of the metrics in terms of rational transfer functions exists [2], [3]. This allows an accurate frequency domain contribution analysis with the tool.

## 2.5 Error Evaluation (AST-4)

The final analysis step accounts for the evaluation of the total pointing error or the current error at any stage where a (sublevel-) requirement is defined. According to [2], three contributions are assessed for statistical requirements: the time-constant, the time-random and the total error

contribution where in the end the absolute value of the error for a given level of confidence is of interest.

The summation of the error contribution depends on the statistical method applied [2]. For the “simplified statistical method” (realized in the PEET prototype), the error contributions are fully described by their equivalent mean and standard deviation (i.e. the underlying distribution is discarded at this point). Mean values are summed linearly. Standard deviations are summed either in an RSS sense (assuming no correlation) or linearly (assuming full correlation). Then, a confidence factor  $n_p = 1, 2, 3$  is used to scale the overall standard deviation (i.e. to represent a  $1\sigma$ ,  $2\sigma$  or  $3\sigma$  confidence level).

The “advanced statistical method” relies on a characterisation of the error properties in terms of probability density functions (PDFs). The total error is generally obtained via convolution of the joint PDF of all error contributions. For the resulting PDF, the respective cumulative distribution function (CDF) is computed and evaluated for the level of confidence specified in the requirement.

The major restrictions of the simplified statistical approach is that it assumes applicability of the central limit theorem. For that reason, one of the key extensions of the PEET release version is an update to the advanced statistical method (see section 4.1).

### 3 TOOL OVERVIEW

#### 3.1 Platforms and Requirements

PEET can be used on multiple platforms (Windows, Linux) on a standard desktop PC or laptop. The tool is designed as an extension to MATLAB and completely runs inside the MATLAB environment. Apart from a plain MATLAB installation (2011b and later), only the Control System Toolbox is required.

#### 3.2 Architecture and External Interfaces

PEET mainly consists of two components: a dedicated graphical user interface (GUI) based on Java and the core computational routines implemented as MATLAB classes.

The GUI is used to define requirement specification parameters, values and identifiers and to set up the pointing system from error sources to the final error using blocks from a database. This can include system transfer models (as in Figure 2) or simply comprise a summation of errors on different requirement levels.

Input data can directly be specified in tables and input fields or imported from MS Excel spreadsheets. In addition to numerical inputs, also MATLAB variable names and notation can be used to specify parameters. All relevant scenario data is stored in an XML file which serves as interface for the MATLAB core classes for the initialization and evaluation of the budget.

Once a pointing system is created and saved using the GUI, two operational modes are possible. First, the GUI can be used directly to start an evaluation and to inspect the results. The communication between the GUI and the MATLAB classes is realized in this case via the Java MATLAB Interface (JMI). Second, a script-based execution of the tool via user-defined Matlab scripts is possible. Together with assigning MATLAB workspace variables to system parameters, this allows batch-mode operations without further use of the graphical user interface and an integration of PEET in a tool-chain with other analysis modules.

The results obtained with PEET can further be exported to MS Excel using a configurable report.

### 3.3 Graphical User Interface

This section gives a brief overview of the PEET GUI which provides different windows for the setup and analysis of budgets.

#### System Editor

The editor panel in the System Editor (Figure 4) is the main tool to design the architecture of a pointing scenario. It can first be populated with a selection of model blocks from the Block Database which are then connected to represent the error signal flow. The workflow for moving and connecting blocks is intentionally similar to the workflow with MATLAB Simulink. Also different levels (subsystems) are supported to maintain a good overview in case of more complex systems. Block dialogs (supported by tooltips) are used to specify error source and system related parameters including signal & parameter units. The latter can be chosen from a predefined set of SI and non-SI units and also custom units can be created. When connecting blocks, the compliance of units is automatically checked by the tool.

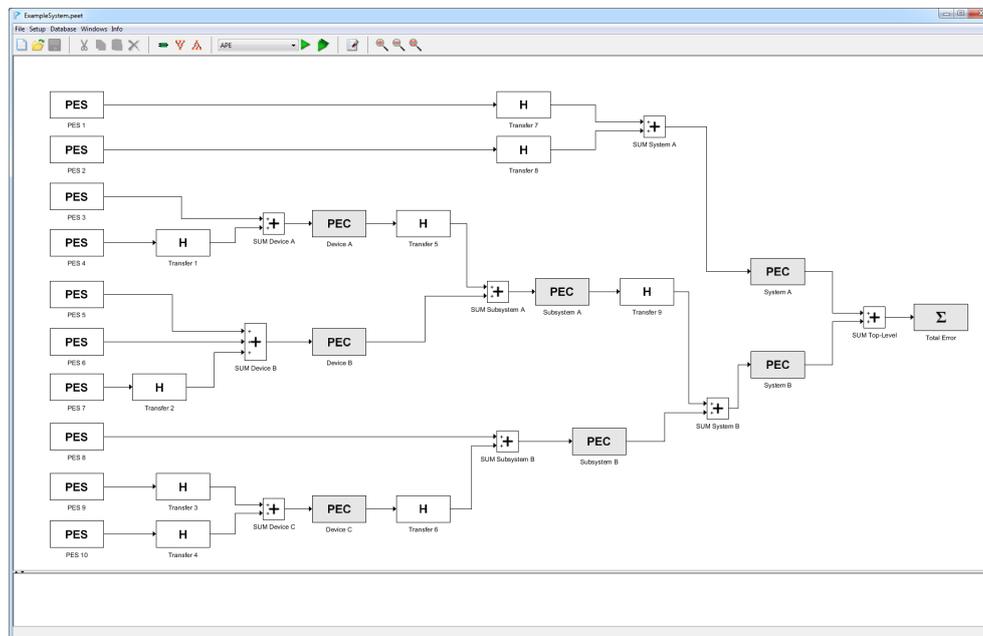


Figure 4: The System Editor

The menus present in the System Editor window serve for file management, requirement definition (multiple requirements sets can be specified in a single scenario) and definition of dependencies (correlation, coherence and phase relations) between error sources.

An execution log serves as scope to track the progress of the evaluation and issues occurring meanwhile (e.g. invalid user parameter inputs).

#### Block Database

The Block Database (Figure 5) - similar to the Library Browser in MATLAB Simulink – contains all building blocks which can be used to populate a pointing scenario. The blocks are categorized in groups (errors sources, static/dynamic systems, etc.).

It contains both generic blocks and parametric models of sensors, actuators and transfer systems which are based on standardized models wherever possible (e.g. [5], [6]). For each model, the block database shows a “quick-view” help with important information about the model (purpose,

input/output dimensions, background). A detailed description of all models is further present in the software user manual.

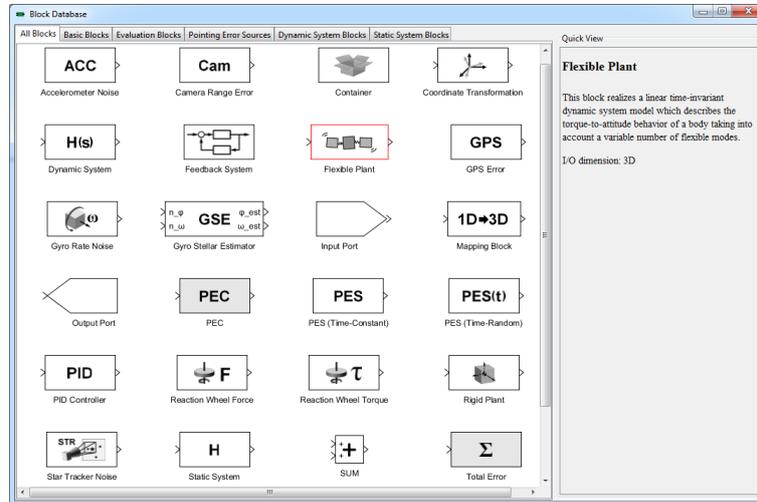


Figure 5: The Block Database

### Budget Tree View

The Budget Tree View (Figure 6) serves to analyze error contributions and the signal content (from different error signal classes) of the entire pointing system. Selecting a block in the tree-like representation of the pointing system shows the related signal content in the information panel on the right. For each signal class, statistical information is provided in terms of mean and standard deviation together with a plot preview of the PDF (or PSD respectively for a random process error signal). In case of spectral requirements, only random process contributions are displayed.

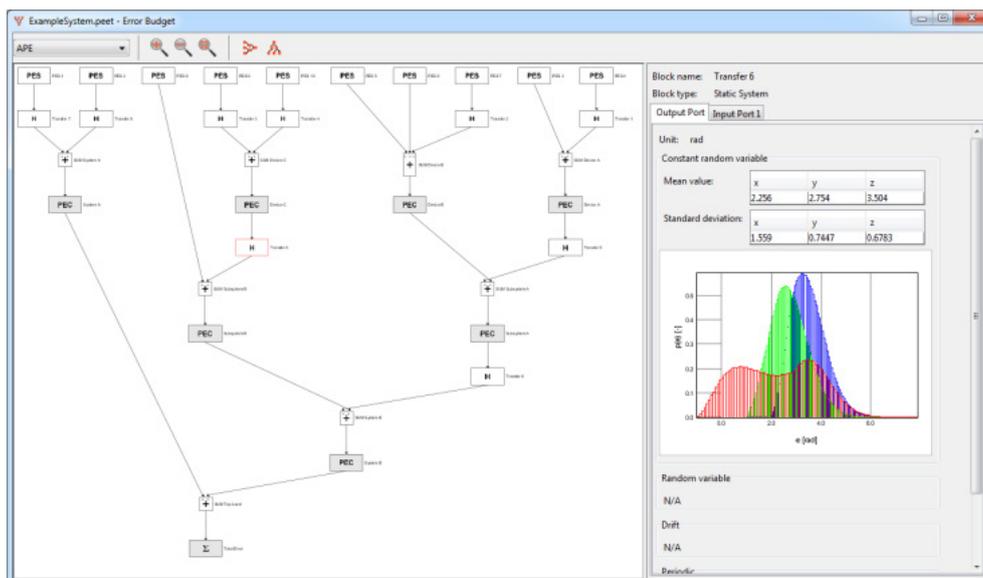


Figure 6: The Budget Tree View

### Breakdown Tree View

The Breakdown Tree View (Figure 7) is used to check the compliance of the budget. It shows only

those blocks where requirements have been associated with (by value and optionally an ID). For statistical requirements, the information panel displays a comparison of budget and requirement values for the time-constant, time-random and total error contributions on all axes together with a plot preview of the underlying CDF. For spectral requirements, the budget spectrum is plotted versus the specified requirement function. In addition, color-coding is used in the tree view for an easy determination of budget violations or proximity to margins.

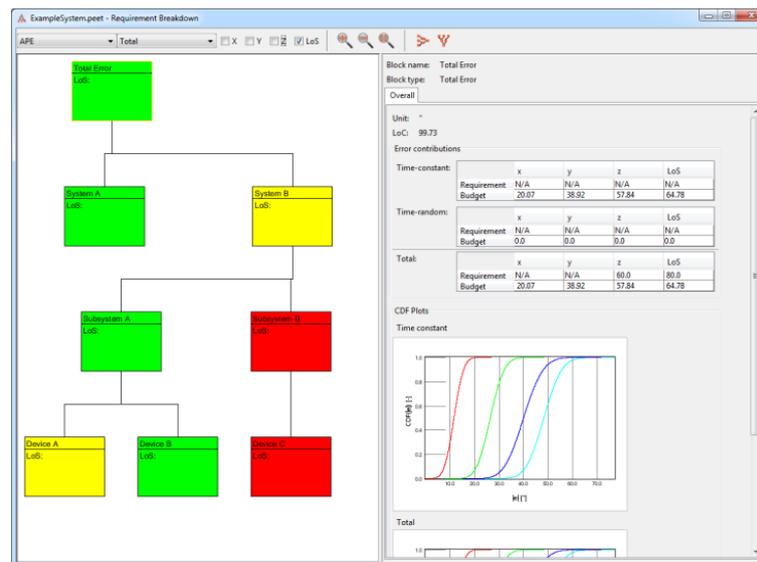


Figure 7: The Breakdown Tree View

### Visualization

Various plots (PDF, CDF, PSD, cumulated variances) are available from the Budget or the Breakdown Tree Views to support the assessment of results. Typical user interactions such as data picking, axes selection, or zooming are available. Dependent on the error signal type, further information such as scatter plots indicating the correlation or cross-spectral density plots are available on separate tabs. All plot data can be exported in a MATLAB compatible format.

### 3.4 Software Limitations

It is important to note that PEET needs to be considered as a systems engineering tool for budgeting 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.

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. Finally, PEET is not intended to be a tool for control/estimator design & tuning, as e.g. the no stability checks of closed-loops in defined feedback systems are carried out. The software intentionally covers only the perimeter of [2] and provides interfaces such that users can plug PEET into own tools for design/tuning.

## 4 KEY SOFTWARE UPDATES

This section presents the major functional updates of the release version of the tool compared to its prototype (apart from improvements on user-friendliness aspects which have been identified from user feedback and industry inputs during working group meetings at ESA). These updates are related to a more accurate approach for evaluating confidence levels, handling initial dependencies between error sources and providing a more flexible definition for requirements.

### 4.1 Advanced Statistical Method

#### Background

The PEET prototype was restricted to the simplified statistical method described in [1] and [2], i.e. the error evaluation is based on the assumption that the central limit theorem is applicable and the total error follows a (nearly) Gaussian distribution (on all axes). While this assumption holds for a “sufficient” number of contributors which have approximately the same magnitude, a significant systematic error is present when dominating non-Gaussian contributions exist.

Figure 8 depicts such situation for two zero-mean distributions (for simplicity & without loss of generality): Assume that the absolute value of the error shall remain below a given value with 99.7% probability ( $p_c = 0.997$ ), i.e. with a confidence factor  $n_p = 3$  as mentioned in section 2.5. For an ideal Gaussian, this obviously results in the exact error value of  $e = 3\sigma$  “by definition”. For a uniform distribution with bounds  $\pm b$ , the standard deviation is given by  $\sigma_u = b/\sqrt{3}$ . Applying the same confidence factor under the same assumptions would then result in error value  $e = \sqrt{3}b$  while the actual result is  $e = 0.997b$ . Thus, a systematic error of more than 80% is present which (already for a “ $2\sigma$  error”) exceeds the explicit bounds of the underlying distribution.

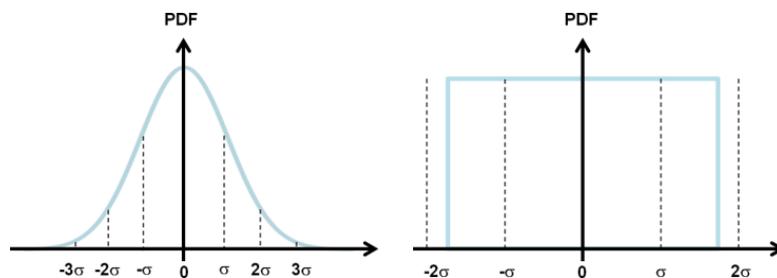


Figure 8: Location of typical confidence levels for a Gaussian distribution (left) and for a uniform distribution (right) “assuming” a Gaussian shape

Another issue with the simplified statistical method is related to the evaluation of line of sight errors (LoS) - even if the errors on individual axes are ideally Gaussian. Assume the z-axis is the desired LoS axis and rotational errors around the 3 target axes are given by  $[e_x, e_y, e_z]$ . Then the proper statistical evaluation of the error is represented by:

$$e_{LOS} = \int_0^{LoC} p \left( \sqrt{e_x^2 + e_y^2} \right) de \quad (1)$$

Reference [2] provides a derived expression which is explicitly valid for instantaneous (or deterministic) LoS errors, i.e. it is not applicable to “statistical” axes errors evaluated from  $|\mu_i| + n_p\sigma_i$ .

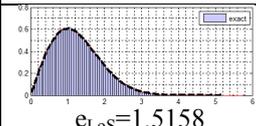
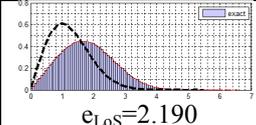
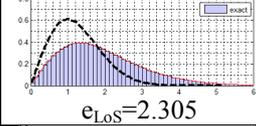
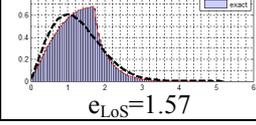
$$e_{LOS} = \sqrt{e_x^2 + e_y^2} \quad (2)$$

For “statistical” errors, [1] provides an approximate solution of the form

$$e_{LOS} = \max(\sigma_x, \sigma_y) \sqrt{-2 \log(1 - p_c)} \quad (3)$$

which is valid for zero-mean Gaussian distributions with closely equal standard deviations on the relevant axis. A “careless” application of equations (2) or (3) to non-matching conditions (e.g. non-Gaussian, non-zero means, significantly different standard deviations) may generally lead to significant systematic errors as depicted in Table 1.

Table 1. Exemplary 68.3% line-of-sight errors

Case	$\approx$ Error Eq.(2)	$\approx$ Error Eq.(3)	PDF and exact value Eq.(3)
X: G(0,1) Y: G(0,1)	-7%	0%	 e <sub>LoS</sub> =1.5158
X: G(1,1) Y: G(1,1)	30%	-30%	 e <sub>LoS</sub> =2.190
X: G(0,1) Y: G(0,2)	-22%	+30%	 e <sub>LoS</sub> =2.305
X: U(-√3, √3) Y: G(0,1)	-10%	-3%	 e <sub>LoS</sub> =1.57

One way to treat these issues is to investigate if the central limit theorem or above-mentioned LoS conditions are applicable and assess the impact case-by-case. The alternative is to fully discard the evaluation based on mean on standard deviation only and systematically account for the underlying probabilities using the advanced statistical method.

### Implementation Details

An analytical treatment of the advanced statistical method implies that the “summation” of moments is replaced by the “summation” of distributions (i.e. convolution). This step as well the subsequent evaluation of errors with a given level of confidence from a CDF requires integration of PDF. If also correlation between different contributions needs to be considered, this further implies to knowledge of the joint PDF all error sources for the convolution.

These preconditions are however in conflict with several constraints:

- Apart from the Control System Toolbox, no further MATLAB toolboxes shall be used (e.g.

- the Symbolic Toolbox) to ensure an easier access for users
- Even with a toolbox for symbolic integration/ convolution, closed-form solutions for an arbitrarily complex system cannot be guaranteed
  - Even with a numerical approach for the convolution, the required joint PDF is usually not known by the user and cannot simply be determined; as a maximum, the marginal distributions of the sources and correlation in terms of a correlation coefficients are expected to be available to users

For these reasons, an entirely numerical (sample-based) approach is realized in PEET. The intrinsic drawback of this approach is a loss of accuracy with respect to the analytical computation. The goal for the project was not to exceed 1% relative error in the overall computation chain using a sufficiently large sample number and a sufficiently large bin number for retrieving the PDF and CDF by numerical integration with adequate resolution. Then, the cost of this numerical error can safely be neglected compared to the potentially large systematic errors related to the simplified statistical method.

The numerical approach for random variable error sources and distributed parameters requires the generation of random samples that represent both the random variable with the desired distribution and the desired correlation between the axes and error sources.

Several methods exist to create samples of correlated standard normal distributed random variables. The method used in PEET is based on [7]. It first requires the covariance matrix  $\Sigma$  of size  $m \times m$  for the  $m$  error signals to be realized. For a standard normal distribution (i.e. with unity variance), the covariance matrix is equal to the matrix of correlation coefficients. Then a Cholesky decomposition is applied to the matrix to obtain an upper triangular matrix  $\mathbf{A}$ :

$$\Sigma = \mathbf{A}^T \mathbf{A} \quad (4)$$

In a next step, a vector  $\mathbf{n}$  of length  $n_s$  is drawn from a standard normal distribution. Both operations are easily feasible using standard MATLAB functions. Then a matrix  $\mathbf{N}$  of size  $n_s \times m$  is computed:

$$\mathbf{N} = [\mathbf{n}_1 \quad \dots \quad \mathbf{n}_m] = \mathbf{A} \mathbf{n} \quad (5)$$

This matrix contains  $m$  vectors of standard normal samples with the desired correlation between each vector. The next step is to transform these to the desired target distribution. This is realized using the so called NORTA (NORmal To Anything) algorithm [8] based on an inverse transform sampling method.

In an intermediate step, the CDF of the normal distribution (hard-coded in the tool) is applied to each column in the matrix  $\mathbf{N}$ .

$$\mathbf{U} = [\mathbf{u}_1 \quad \dots \quad \mathbf{u}_m], \quad \mathbf{u}_i = \Phi(\mathbf{n}_i) \quad (6)$$

In this specific case, this results in a set of vectors  $\mathbf{u}_i$  whose samples represent a uniform distribution between 0 and 1. Then, similarly the inverse CDF (ICDF) of the target distribution is applied to each value in the vectors  $\mathbf{u}_i$ .

$$\mathbf{X} = [\mathbf{x}_1 \quad \dots \quad \mathbf{x}_m], \quad \mathbf{x}_i = ICDF_{\text{target}}(\mathbf{u}_i) \quad (7)$$

The ICDFs of the target distributions are also represented numerically in Eq. (7) and each  $x_i$  is interpolated for the current  $u_i$  value.

This finally gives vectors of random samples  $\mathbf{x}_i$  for each error signal which describe random variables with the desired PDF. And, more important, the transformation method preserves the correlation.

Preserving the correlation under any monotonic transformation is only valid assuming rank correlation (i.e. Spearman coefficients), but not for linear correlation as e.g. represented by Pearson product-moment coefficients [8], [9]. As the matrix  $\Sigma$  itself requires Pearson product-moment coefficients  $\rho_p$  for the setup, first a conversion from Spearman ( $\rho_s$ ) to Pearson ( $\rho_p$ ) coefficients is internally realized [9]:

$$\rho_p = 2 \sin\left(\rho_s \frac{\pi}{6}\right) \quad (8)$$

This also implies that correlation coefficients specified in PEET represent Spearman rank correlation coefficients, not Pearson product-moment coefficients.

#### 4.2 Concept of Statistical Domains

The concept of statistical domains is a compliant generalisation of the statistical interpretation concept in [1] and [2] which allows a more flexible definition of requirements.

Basically, the statistical interpretation is a rule for how the “pair” of temporal and ensemble properties of each error source shall be evaluated, i.e. either taking into account the worst-case or the statistical distribution of the domains “time” and “ensemble”.

While the domain “time” is unique for all error sources, different “ensemble” domains can exist in a pointing scenario (as indicated in Figure 9) dependent on the “background” of the randomness, e.g. random misalignments due to the manufacturing, different sensor noise levels due to different orientation in space or temperature conditions.

The statistical interpretation inseparably treats contributions from these domains and there is no possibility to adapt their statistical evaluation individually apart from manually “remodelling” error source descriptions. With the domain concept, the statistical treatment (worst-case or statistical) of the domain “time” and each “ensemble” domain can be configured individually according to the evaluation needs of a requirement.

This allows for instance, the evaluation of a budget for the worst case satellite (“temporal interpretation” for manufacturing errors), but statistical evaluation of errors varying over an

ensemble of observations (“mixed interpretation”).

Additionally, also different levels of confidence for contributions from different domains can be defined.

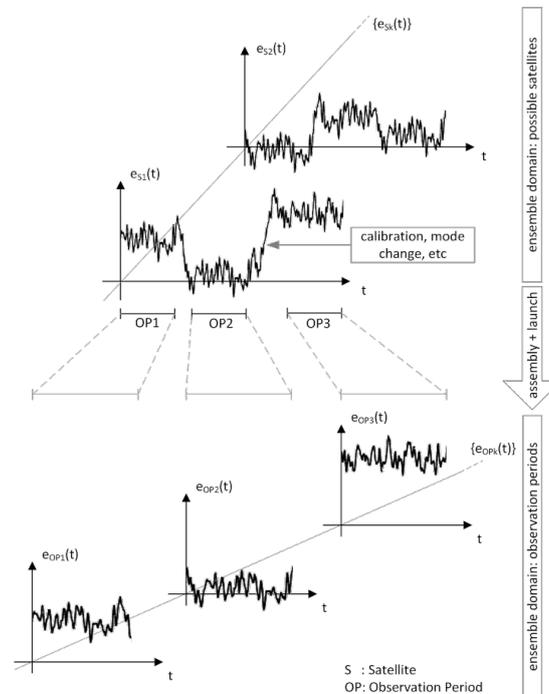


Figure 9: Different ensemble domains “satellite” (e.g. misalignments) and “observation period”

### 4.3 Error Source Dependencies

In the PEET prototype, the dependencies between different error sources could be defined by specifying either full correlation or no correlation between all axes of a single error source or entirely between different error sources.

In PEET 1.0, the user is able to specify correlation individually (between axes of a single error source or axes of different sources) by specifying correlation coefficients. The specified correlation matrix is automatically checked for validity by the tool and a valid “close” realization is provided in case of an infeasible setup [8].

Furthermore, the tool distinguishes between temporal and ensemble correlation, i.e. time-random sources can be correlated in time while their specification parameters can be correlated over an ensemble in the given domain and valid setup combinations are provided automatically.

The dependencies for random process error sources described as power spectral densities, can either be defined by coherence factors or an explicit definition of cross-spectra.

As - different to the simplified statistical method - also the phase relation between periodic signals affect the error PDF, phase properties can also explicitly be defined.

## 5 TOOL VERIFICATION AND VALIDATION

To verify the runtime routines of the tool, a comprehensive test campaign was carried out especially for all lower level algorithms. This covers static and dynamic transfer of all error signal classes,

application of statistical interpretation and time-windowed error metrics to the raw data as well as implementation tests with all specific source and system models.

A special focus was put on the verification of the numerical approach for the "advanced statistical method". The sample-based evaluation for all fundamental distributions (also in combinations of temporal and parameter distributions in a single error source) was compared to analytical results wherever closed-form solution for the convolution integral exist. The envisaged goal of less than 1% error could be met with a selected number of around one million samples per error source and axis and with around 10000 bins for the generation and integration of the PDFs. This relative error goal was safely reached also for the critical case of unbounded or heavily tailed distributions (where sufficient samples need to "fall" in the outer bins) and performance requirements with a level of confidence of up to "3-Sigma" (usually even up to "4-Sigma" with less than 5% error).

Verification and validation on system level was carried out by Airbus DS. For that purpose, two case studies (EDRS and the fictive PointingSat mission) were set up and evaluated using PEET. The results were compared to an in-house tool both with respect to benchmarking and accuracy of the budgets. This assessment confirmed a comparable accuracy but with a significant reduction of computation time and an easier setup and adaptability of the pointing scenarios. Further details on the application of the tool to a case study is presented in a specific paper [10].

## 6 CURRENT APPLICATIONS

The prototype version of PEET has already proven to be the tool of choice for pointing error engineering activities in a number of ESA projects (MetOp-SG, Proba-3, Juice, EDRS, Euclid to name a few) either as main tool or for cross-checking the results obtained using other tools.

With the availability of PEET 1.0, a verified and more user-friendly version of the tool, enhanced with advanced statistical methods, is available for the European space community and, as a consequence, it is expected to be widely used in ESA projects. Early confirmation of this is the endorsement of PEET 1.0 by the MetOp-SG project as the sole platform for exchange of information between the various entities involved (ESA, Eumetsat, industrial prime contractors, and industrial sub-contractors, including payload suppliers). Other ESA projects are expected to follow a similar approach.

## 7 SUMMARY AND OUTLOOK

The developed and elaborated concepts during the PEET project phase partially go beyond the level of detail described in the ECSS standard [1] and the ESA Pointing Error Engineering Handbook [2]. It is expected, that this knowledge will flow back into an update of the handbook in 2017.

The tool itself provides traceability and a common platform for exchange of multi-level budgets within a project. It allows an efficient handling of budgets especially when a variety of requirements on different instruments exists. Providing accurate budgets (applicable to general performance problems, not only pointing) in line with the ECSS standard and the ESA Pointing Error Engineering Handbook - but without the restriction on the applicability of the central limit theorem - PEET has the potential to become a standard tool that can be used for any future ESA mission.

## 8 DISCLAIMER

The view expressed in this paper can in no way be taken to reflect the official opinion of the European Space Agency.

## 9 REFERENCES

- [1] European Cooperation on Space Standardization, *Control Performance Standard ECSS-E-ST-60-10C*, ESA-ESTEC Requirements & Standards Division, 2008.
- [2] ESA Engineering Standardisation Board, *Pointing Error Engineering Handbook ESSB-HB-E-003*, ESA-ESTEC Requirements & Standards Division, 2011.
- [3] Hirth M. et al., *The Pointing Error Engineering Tool (PEET): From Prototype to Release Version*, 6th International Conference on Astrodynamics Tools and Techniques, Darmstadt, Germany, 2016
- [4] Pittelkau, M.E., “Pointing Error Definitions, Metrics, and Algorithms”, *American Astronautical Society*, AAS 03-559, p. 901, 2003.
- [5] Bendat J.S., Piersol A.G., *Random Data – Analysis and Measurement Procedures*, Wiley, 3rd edition, 2000
- [6] IEEE Aerospace and Electronic Systems Society, *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
- [7] Gentle, J.E, *Computational Statistics*, Springer, 2009
- [8] Gosh S., *Dependence in Stochastic Simulation Models*, PhD thesis, Cornell University 2004
- [9] *Simulating Dependent Random Variables Using Copulas*, Mathworks Online Documentation, via <http://de.mathworks.com/help/>, last accessed April 4, 2017
- [10] Ott T. et al., *PointingSat – High Precision Pointing Error Analysis with ESA PEET v1.0*, 10th International ESA Conference on Guidance, Navigation & Control Systems, Salzburg, Austria, 2017