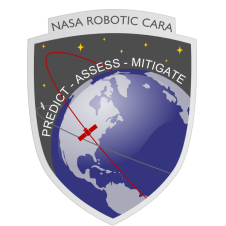


Software Development Kit: Covariance Realism

CONJUNCTION ASSESSMENT AND RISK ANALYSIS (CARA) PROGRAM



*Matthew Hejduk*

*Astrorum, Inc*

*Waco, Tx*

*Travis Lechtenberg*

*Doyle Hall*

*Luis Baars*

*Omitron, Inc*

*Colorado Springs, CO*

|  |
| --- |
|  |

National Aeronautics and

Space Administration

Goddard Space Flight Center,

Greenbelt, MD 20771

**April 2021**

**Preface**

This document outlines the Covariance Realism Assessment submitted as part of the Software Development Kit (SDK). The SDK is intended to provide both industry and government customers with a code base with which to perform standard calculations inherent to the Collision Avoidance (CA) problem and as outlined in the CA Standard.

# Table of Contents

[1.0 Introduction 4](#_Toc71002916)

[1.1 Required Software 4](#_Toc71002917)

[2.0 Covariance Realism Software 5](#_Toc71002918)

[2.1 Overview 5](#_Toc71002919)

[2.2 Software Modules 5](#_Toc71002920)

[3.0 Acronyms 13](#_Toc71002921)

[4.0 References 14](#_Toc71002922)

**List of Tables**

[Table 1: Residual Evaluation Routine Input Parameters 7](#_Toc71002923)

[Table 2: Residual Evaluation Routine Output Parameters 8](#_Toc71002924)

[Table 3: Residual Evaluation Unit Test Cases 9](#_Toc71002925)

[Table 4: Scale Factor Optimization Routine Input Parameters 10](#_Toc71002926)

[Table 5: Scale Factor Optimization Routine Output Parameters 11](#_Toc71002927)

[Table 6: Scale Factor Optimization Unit Test Cases 11](#_Toc71002928)

# Introduction

The CARA Software Development Kit (SDK) contains entries and artifacts for each major algorithm needed to perform the required Collision Avoidance (CA) calculations outlined in the CA Standard. For these algorithms, the SDK will include a version of the algorithm, a driver program to process input residual data, producing the needed calculation or output, and a series of test cases that exercise the algorithm and produce validated results.

This document describes a specific algorithm, its associated inputs and outputs, the methodology used within the algorithm and examples of usage.

## Required Software

The following list is of software and hardware requirements for use of this SDK:

* Matlab 2019b
* Statistics and Machine Learning Toolbox
* Parallel Computing Toolbox (Recommended but optional)

# Covariance Realism Software

## Overview

The Covariance Realism Assessment software package examines residual sets to determine whether their associated covariances reasonably represent the residuals’ expected distribution. The typical application is to examine, for a particular spacecraft, sets of predicted ephemerides that provide both predicted states and predicted position covariances, in the presence of a definitive (as-flown) ephemeris that will allow residuals between the predicted and definitive ephemerides to be calculated. The accompanying technical memo *Covariance Realism Evaluation Approaches.pdf* gives both abbreviated theoretical treatments and practical information regarding how best to proceed with position covariance realism assessments. The present document repeats some of this information but is more focused on practical issues regarding the use of the software package.

It is an overworn expression to state that a particular endeavor is more an art than a science. At the same time, it is certainly true that there are no hard-and-fast rules or test result sets that indicate whether the covariances contained in a particular predicted ephemeris set should be considered realistic. In the accompanying *Covariance Realism Test Results.pptx* file, each (synthetic) test case represents a clear situation; but to evaluate even these requires some subtlety of examination of results; and data for real ephemerides are often even more ambiguous. The accompanying conference paper *Zaidi and Hejduk 2016.pdf* outlines how a covariance realism evaluation and compensation investigation was performed for a particular satellite constellation set; perhaps this can give some implementation ideas. NASA missions who wish assistance in the interpretation of position covariance realism results should contact NASA CARA.

## Software Modules

This SDK consists of several software modules intended to encompass the needed analyses for assessing covariance realism. The included modules are the following:

|  |  |
| --- | --- |
| Module Name | Description |
| AnalysisDriver.m | A basic driver that loads a test input set of data contained within a Matlab data file and runs the entire covariance realism software string. A good module to examine to understand how the entire software flows. |
| EvaluateResiduals.m | Checks normalized residuals for conformity to a Gaussian distribution (by component) and Mahalanobis distances (technically the square of Mahalanobis distances) for conformity to a 3-DoF chi-square distribution. |
| FullSpecDistVec.m | Subfunction called by other routines to evaluate a fully-specified empirical distribution’s conformity to a hypothesized parent distribution. |
| DetermineOptimalScaleFactor.m | Performs distribution testing iteratively to determine the optimal covariance scale factor to achieve desired distribution matching. This module uses parallel processing to assess this, but may be adapted to function without it to the detriment of processing speed. |
| CreateTestInput0X.m | Creates one of the test input files. These files are also included, but providing this code allows the user to understand exactly how the test datasets are produced. |

The two main routines to be called and used in the context of covariance realism are the *“EvaluateResiduals.m”* and *“DetermineOptimalScaleFactor.m”* routines.

### Residual Evaluation - Description

The first pass of the residual evaluation process examines the normalized position residuals without any scaling factor applied; this gives a sense of the realism of the covariances as produced, without any artificial alteration.

First, the residuals are normalized by dividing each individual residual by the standard deviation of the appropriate component from the accompanying covariance. This creates a set of “z-variables” that should have a mean of 0 and a standard deviation of 1. It is presumed that the residuals are unbiased, meaning that the mean of each component’s residuals is 0; otherwise, a covariance matrix cannot properly represent the errors. The second step is to produce a series of random samples, without replacement, from the set of residuals (separated by component) in order to produce *i* sets of *j* samples, one *i* x *j* set for each component. Each of *i* sets is then tested for normality, using both the more permissive Cramér – von Mises test and the more demanding Anderson-Darling test. Then the Mahalanobis Distance is computed for each residual group (that is, for each ordered triple of RIC residuals and its associated covariance) is computed. Following the same resampling procedure as described above, one i x j set of Mahalanobis Distance samples is tested for conformity to a 3-DoF chi-squared distribution.

Following this, the code will generate a series of output graphs based on the input data, these graphs are composed of the following assuming a figure number offset number of 10:

* **Figure 11: Estimated PDFs of Resampled Normalized Residuals**. This chart gives estimated PDFs (from kernel density estimation) of the means of all of the resampled sets, by component. This allows a quick visual confirmation whether the residuals appear to be unbiased. One should not expect the peak of this distribution to be exactly at zero, especially because the PDF estimation is not fully precise; but it does allow one to tell whether there is an appreciable bias.
* **Figure 12: Portion of Samples Passing Normality Test**. This chart gives a CDF by p-value of the results of both the Cramér – von Mises and Anderson-Darling tests. To read these results, determine the p-value with which one is comfortable, and the cumulative percentage value that the graph associates with that p-value is the percentage of cases that achieved that value or a higher value. One should not expect compliance at the 100% level; but below 90% one starts to get increasingly uncomfortable.
* **Figure 13: Portion of Samples Passing 3-DoF Chi-Squared Test**. As in Figure 12 above, a CDF of cumulative percentage vs p-value is given; and the same interpretation guidance applies.

For the figures above, an empty graph indicates that none of the samples passed the assigned tests. In some of the illustrative test cases, this does happen; in actual practice it is rare to perform truly that badly.

### Residual Evaluation – Source Code Description

The function contained within the SDK primarily concerned with the evaluation of residuals is the:

EvaluateResiduals.m

routine, which performs the analysis described above.

As inputs, the routine accepts the following:

Table 1: Residual Evaluation Routine Input Parameters

|  |  |
| --- | --- |
| **Input Variable** | **Definition** |
| Residuals | [NX3] Set of position residuals (units irrelevant as long as coincident with Covariance units) |
| Covariances | [NX3X3] Set of position covariances matching the residuals |
| Scale Factor | [Double] (optional) scale factor by which to multiply the covariance. Because a pre- and post-multiplication is presumed, covariance is actually multiplied by the square of the scale factor (default = 1) |
| Options | [Structure] (optional) controls a variety of options for the covariance realism analysis, available fields:  NumberOfTrials: number of sample trials to take from dataset for GOF testing (10000 recommended/default)  TrialSampleSize: number of samples from dataset to take for each trial (50 recommended/default)  SignificanceLevel: GOF test p-value (0.02 recommended/default) |
| FigureOffset | [Integer] (optional) Offsets to use in applying Figure numbers; useful if analyzing several different. Recommended value = 10; set to 0 to suppress generation graphs (Default = 0) |
| PropagationStateText | [String] (optional) allows the propagation state to be displayed in graph titles (default = '') |
| MeansGraph | [Boolean] (optional, Default = false) flag to determine whether to produce graph of means |

The Residual Evaluation routine outputs the following:

Table 2: Residual Evaluation Routine Output Parameters

|  |  |
| --- | --- |
| Output Variable | Definition |
| NormalityTestFullResiduals | [3X4] Results from testing for Gaussian behavior the entire residual set, by component. Each row is the results from a particular test (Cramer - von Mises, Watson [not used], and Anderson-Darling), and each column is a component (radial, in-track, cross-track, and a blank column to keep the array from being 3 x 3 and thus confusing). Array gives p-value: 0.009 means < 0.01; 0.26 means > 0.25 |
| NormalityTestResampled ResidualsInterpolatable | [3X4] Same idea as the previous array, but here resampled results are given (percentage of trials that passed, according to specified p-value). Same definition of array contents |
| Chi2TestFullM2 | [3X4] Gives results of the test of full normalized position error vectors (full dataset) for conformity to a 3-DoF chi-squared distribution. Only column 4 is populated. Array contains p-value information |
| Chi2TestResampled M2Interpolatable | [3X4] Same as previous array but with results from all of the trials from the resampling approach. Same array definitions, but percent of cases passing (at the specified p-value) is what is reported |

Validation cases for this algorithm are contained within the unit test suite for the SDK at:

..\SDK\UnitTest\CovarianceRealism\EvaluateResiduals\_UnitTest.m

These test cases were developed using defined distributions of residuals and covariance matrices which were intended to give specified results.

Table 3: Residual Evaluation Unit Test Cases

|  |  |
| --- | --- |
| Test ID | Description |
| test01 | Normal distribution, μ=0, σ=1; covariances reflect this. |
| test02 | Normal distribution, μ=0, σ=1; covariances based on normal distribution with μ=0, σ=4 (should generate scale factor of ~0.25 |
| test03 | Normal distribution, μ=0, σ=1; covariances based on normal distribution with μ=0, σ=0.333 (should generate scale factor of ~3 |
| test04 | Student’s t-distribution, ν=3; covariances based on sample covariances of this same exponential distribution (should produce failure on the first pass and marginal results when rescaled) |
| test05 | Exponential distribution, μ=1; covariances based on sample covariances of this same exponential distribution (should produce failure on the first pass and marginal results when rescaled) |

### Scale Factor Optimization - Description

In this module, an analysis is performed to explore whether scaling the covariance by a single multiplicative factor would improve the situation appreciably. Since the covariance production process can undersize or oversize covariances (with the former being far more common), it is often helpful to know which of these is taking place and the degree to which it is observed. It is not recommended that a scale factor simply be applied operationally as a solution to any apparent covariance mis-sizing; instead, the root cause of the problem should be identified and repaired accordingly. However, the size of the scale factor, and whether it is greater or less than unity, are both helpful data in assessing the type and degree of any mis-sizing problem; and re-running analyses after attempted repairs and examining the scale factor’s size in the post-repair state can help to assess whether the repairs have been successful.

This algorithm operates by determines a single scale factor for the covariance that minimizes one of selected goodness-of-fit test statistics for testing for a normalized residual set's conformity to a 3-DoF chi-squared distribution.

After the code identifies the correct scaling of the input covariance, the code will generate an output graph based on the input data, this graph number is defined assuming a figure number offset number of 20:

* **Figure 21: Estimated PDF of Scale Factors**. A KDE of the set of scale factors produced by analyzing each sample is given, along with a point that represents the scale factor obtained by analyzing the entire dataset, without resampling. One can eyeball the peak of the distribution to see what the most common scale factor is and how well that aligns with the full-sample calculation.

For the figures above, an empty graph indicates that none of the samples passed the assigned tests. In some of the illustrative test cases, this does happen; in actual practice it is rare to perform truly that badly.

### Scale Factor Optimization – Source Code Description

The function contained within the SDK primarily concerned with the evaluation of residuals is the:

DetermineOptimalScaleFactor.m

routine, which performs the analysis described above.

As inputs, the routine accepts the following:

Table 4: Scale Factor Optimization Routine Input Parameters

|  |  |
| --- | --- |
| **Input Variable** | **Definition** |
| Residuals | [NX3] Set of position residuals (units irrelevant as long as coincident with Covariance units) |
| Covariances | [NX3X3] Set of position covariances matching the residuals |
| Options | [Structure] (optional) controls a variety of options for the covariance realism analysis, available fields:  NumberOfTrials: number of sample trials to take from dataset for GOF testing (10000 recommended/default)  TrialSampleSize: number of samples from dataset to take for each trial (50 recommended/default)  SignificanceLevel: GOF test p-value (0.02 recommended/default) |
| Test Stat | [Integer] (optional) the statistic to use when calculating optimal scale factors (1 - Cramer - von Mises; 3 = Anderson-Darling) (Default = 1) |
| FigureOffset | [Integer] (optional) Offsets to use in applying Figure numbers; useful if analyzing several different. Recommended value = 10; set to 0 to suppress generation graphs (Default = 0) |

The Scale Factor Optimization routine outputs the following:

Table 5: Scale Factor Optimization Routine Output Parameters

|  |  |
| --- | --- |
| Output Variable | Definition |
| FullM2ScaleFactor | [Double] Scale factor, based on the full set of residuals, that minimizes the selected test statistic. |

Validation cases for this algorithm are contained within the unit test suite for the SDK at:

..\SDK\UnitTest\CovarianceRealism\DetermineOptimalScaleFactor\_UnitTest.m

These test cases were developed using defined distributions of residuals and covariance matrices which were intended to give specified results.

Table 6: Scale Factor Optimization Unit Test Cases

|  |  |
| --- | --- |
| Test ID | Description |
| test01 | Normal distribution, μ=0, σ=1; covariances reflect this. |
| test02 | Normal distribution, μ=0, σ=1; covariances based on normal distribution with μ=0, σ=4 (should generate scale factor of ~0.25 |
| test03 | Normal distribution, μ=0, σ=1; covariances based on normal distribution with μ=0, σ=0.333 (should generate scale factor of ~3 |
| test04 | Student’s t-distribution, ν=3; covariances based on sample covariances of this same exponential distribution (should produce failure on the first pass and marginal results when rescaled) |
| test05 | Exponential distribution, μ=1; covariances based on sample covariances of this same exponential distribution (should produce failure on the first pass and marginal results when rescaled) |

# Acronyms

|  |  |
| --- | --- |
| CARA | Conjunction Assessment Risk Analysis |
| CDF | Cumulative Distribution Function |
| CDM | Conjunction Data Message |
| ECI | Earth Centered Inertial |
| HBR | Hard Body Radius |
| Pc | Probability of Collision |
| SDK | Software Development Kit |

# References