FM\_BSA Instructions and documentation

# Introduction

FM\_BSA is a class-based MATLAB system for estimating the parameters of wide-band FM signals from prior knowledge of the modulated signal. The class is divided into multiple methods so that many different types of experiments can be managed using different sets of methods. Also methods can be replaced of overridden for future refinements.

# Citations:

Kyriazis G. A., “Estimating parameters of complex modulated signals from prior information about their arbitrary waveform components,” IEEE Trans. Instrum. Meas., v. 62, no. 6, pp. 1681-1686, June 2013. %

Citation:

Kyriazis G. A., “A Cartesian method to improve the results and save computation time in Bayesian signal analysis,” in Advanced Mathematical and Computational Tools in Metrology and Testing X (AMCTM X), Series on Advances in Mathematics for Applied Sciences, vol. 86, F. Pavese; W. Bremser; A.G. Chunovkina; N. Fischer; A.B. Forbes (eds.), World Scientific, 2015, pp. 229-240.

# LICENSE:

NIST-developed software is provided by NIST as a public service. You may use, copy, and distribute copies of the software in any medium, provided that you keep intact this entire notice. You may improve, modify, and create derivative works of the software or any portion of the software, and you may copy and distribute such modifications or works. Modified works should carry a notice stating that you changed the software and should note the date and nature of any such change. Please explicitly acknowledge the National Institute of Standards and Technology as the source of the software.

NIST-developed software is expressly provided "AS IS." NIST MAKES NO WARRANTY OF ANY KIND, EXPRESS, IMPLIED, IN FACT, OR ARISING BY OPERATION OF LAW, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, AND DATA ACCURACY. NIST NEITHER REPRESENTS NOR WARRANTS THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT ANY DEFECTS WILL BE CORRECTED. NIST DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OF THE SOFTWARE OR THE RESULTS THEREOF, INCLUDING BUT NOT LIMITED TO THE CORRECTNESS, ACCURACY, RELIABILITY, OR USEFULNESS OF THE SOFTWARE.

You are solely responsible for determining the appropriateness of using and distributing the software and you assume all risks associated with its use, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and the unavailability or interruption of operation. This software is not intended to be used in any situation where a failure could cause risk of injury or damage to property. The software developed by NIST employees is not subject to copyright protection within the United States.

# Quick Start Instructions

A quick demonstration of the system is available using the MATLAB unit test class-based class: **test\_FM\_BSA**

This class can be used to run regression tests on the FM\_BSA class which demonstrate its capabilities. It can also be used to run some experiments which were used to create the plots used in the draft conference paper on the grid search technique.

To run the class-based testing, first be sure that the folder and all subfolders containing the FM\_BSA class as well as the AnalyticTimeseries have been added to the MATLAB path.

To instantiate the unit test class, enter at the MATLAB command line:

**testCase = test\_FM\_BSA();**

This only needs to be done once even if the class is later modified (for example if tests or experiments are later added or commented/uncommented, then you do not need to re-instantiate the class.

If you want to see intermediate graphics of the objective function contour plots during grid search and optimization, you can turn on debug mode with the command line command:

**testcase.debug = true;**

You can turn off the plots by setting the testcase.debug property to “false”

To execute the regression tests and/or he experiments, enter at the MATLAB command line:

**res = testCase.run;**

When the above command is executed, all functions inside the section **methods (Test)** will be executed. As of now, this includes only two regression tests: **test50f0\_2m0\_2a5 and test50f0\_5m0\_5a0 are run.**

**%% Test Methods**

**% These functions will be called on >> "res = run(testCase);"**

**methods (Test)**

**function regressionTests(self)**

**self.fig = 1;**

**test50f0\_2m0\_2a5(self); self.fig=self.fig+1; % Phase Mod, fm = 2, k = 2.5**

**test50f0\_5m0\_5a0(self); self.fig=self.fig+1; % Phase Modu, fm = 2, k = 2.5**

**end**

**function experiments(self)**

**%FcarrDfContour(self)**

**%GridSearchThreshold(self)**

**end**

**end**

These two tests are both 3 phase FM modulation with Fm = 2.0 and 5.0 Hz respectively and Km = 2.5 and 5.0. Therefore, the first test has a peak frequency deviation of 5 Hz and the second has 25 Hz.

When these tests run the first function called is **setTsDefaults();** which sets up some default SignalParameters parameters for the simulated time series that will be created. the test functions then add some of their own parameters then calls **getTimeSeries()**which instantiates an **AnalyticTS\_class()** object **self.TS**. The constructor for this class automatically creates an analytic time series (real and imaginary values). The SignalParmeters 2D array describes most sinusoidal signal types used in power system (particularly PMU) testing.

The timeseries is given a name, then the function **self.runMod1Second(true);** is called. This function runs one parameter estimate for each nominal cycle period over the duration of one second. In other words, at 50 Hz nominal frequency, it will estimate 50 sets of parameters, advancing by 1/50 second each estimate.

**self.runMod1Second()** Instantiates an **FM\_BSA\_Class()**instance, passing the prior information to its constructor as well as the signal data. A loop is run for each of the 6-phase input signal to be analyzed by the class, Then the following class methods are called:

**[startPt] = FM.GridSearch();** Performs a grid search for good starting point estimates.

**[endpt\_BSA] = FM.BSA\_Est(startPt);** Performs the BSA to estimate the endpoint parameters.

**[estParams] = FM.Param\_Est(endpt\_BSA);** From the endpoint parameters, calculates Modulo, Carrier phase angle (Phi\_BSA), Carrier Frequency (Fcarr\_BSA), Modulation phase angle (Phim\_BSA) and peak frequency deviation (dF\_BSA).

**[Synx(phase),Freq(phase),ROCOF(phase)] = FM.Synx\_Calc(estParams);** Calculates the Complex Phasor, the instantaneous Frequency and the Instantaneous ROCOF at the center of the analysis window.

For each parameter estimation, the phasor, frequency and ROCOF at the center of the analysis window is calculated as the actual value of the estimation. In the AnalyticTimeSeries class, the expected values at the center of the window is also calculated. After one second of data has been calculated, plots of the Phasor Total Vector Error (TVE), Magnitude Error (ME) and Phase Error (PE) are shown as well as Frequency Error (FE) and ROCOF Error (RFE) are shown.

|  |  |
| --- | --- |
|  |  |

# Experiments

You can comment out the regression Tests and uncomment the experiments:

**%% Test Methods**

**% These functions will be called on >> "res = run(testCase);"**

**methods (Test)**

**function regressionTests(self)**

**self.fig = 1;**

**%test50f0\_2m0\_2a5(self); self.fig=self.fig+1; % Phase Modulation, fm = 2, k = 2.5**

**%test50f0\_5m0\_5a0(self); self.fig=self.fig+1; % Phase Modulation, fm = 2, k = 2.5**

**end**

**function experiments(self)**

**self.debug=true;**

**FcarrDfContour(self)**

**GridSearchThreshold(self)**

**end**

**end**

then re-run the testcase using **res = testCase.run;**

The experiment, **FcarrDfContour(self),** generates the objective function contour plots used in the paper and shown here:

|  |  |  |
| --- | --- | --- |
|  |  |  |

The commented experiment, **GridSearchThreshold(self)** generates theplots shown here:

|  |  |
| --- | --- |
|  |  |

before running **GridSearchThreshold(self),** setting testcase.debug = true will display contour plots for each value of Fm and Km which objective function minima is used in the above figures. An .mp4 animation will be created in the MATLAB working folder if testcase.makeAnimation = true;

As of this version (2002 Jan 31), the surface fit equation and coefficients are:

val(x,y) = p00 + p10\*x + p01\*y + p20\*x^2 + p11\*x\*y + p02\*y^2

Coefficients (with 95% confidence bounds):

p00 = -10.88 (-11, -10.75)

p10 = 0.3855 (0.3203, 0.4507)

p01 = 1.038 (0.9724, 1.103)

p20 = -0.02687 (-0.03749, -0.01625)

p11 = -0.0004427 (-0.009796, 0.008911)

p02 = -0.1067 (-0.1173, -0.09605)