PESTO

1.0

Generated by Doxygen 1.8.11

Contents

1 PESTO Documentation

1.1 Introduction

Computational models are commonly used in diverse disciplines such as computational biology, engineering, or meteorology. The parameterization of the these models is usually based on measurements or observations. The process of inferring model parameters from such data is called model calibration or parameter estimation. This parameter estimation is often not straightforward due to non-linearities in the model equations or due to the mere size and the resulting computational challenges. Therefore, efficient algorithms are required to provide robust results within acceptable time.

PESTO is a freely available Parameter EStimation TOolbox for MATLAB (MathWorks) implementing a number of state-of-the-art algorithms for parameter estimation. It provides the following features, which are explained in more detail below:

- Parameter estimation from measurement data by global optimization based on multi-start local optimization (requires `MATLAB Optimization Toolbox`)
- Parameter sampling using Markov Chain Monte Carlo (MCMC) algorithms
- Uncertainty analysis based on local approximations, parameter samples and profile-likelihood analysis (requires `MATLAB Optimization Toolbox`)
- Visualization routines for all analyses
- Parallel processing (requires `MATLAB Parallel Computing Toolbox`)
- ...

PESTO functions can be applied to any user-provided formulation of an optimization problem with an objective function that can be evaluated in MATLAB. Besides the objective function, upper and lower bounds for the function parameters need to be specified.

1.2 Availability

PESTO can be freely obtained from `https://github.com/ICB-DCM/PESTO/` by downloading the zip archive at `https://github.com/ICB-DCM/PESTO/archive/master.zip` or cloning the `git` repository via

```
git clone git@github.com:ICB-DCM/PESTO.git
```

1.3 Installation

If the zip archive was downloaded, it needs to be unzipped and the main folder has to be added to the MATLAB search path (non-recursively).

If the repository was cloned, the main folder needs to be added to the MATLAB search path (non-recursively).

Note: Detailed instructions on how to modify your MATLAB search path are provided here: `https://de.↩mathworks.com/help/matlab/matlab_env/add-remove-or-reorder-folders-on-the-search-path.↩html`

Third-party packages

PESTO provides an interfaces to several other toolboxes which are not included in the PESTO archive:

- PSwarm: `http://www.norg.uminho.pt/aivaz/pswarm/`
- MEIGO: `http://gingproc.iim.csic.es/meigo.html`
- DRAM: `http://helios.fmi.fi/~lainema/dram/`

To use their functionality, these toolboxes have to be installed separately. Please consult the respective user manuals for details.

1.4 Licensing

See `LICENSE` file in the PESTO source directory.

1.5 How to cite

This section will be updated upon publication of PESTO.

1.6 Code organization

The end-user interface is provided by the MATLAB functions and classes in the top-level directory. PESTO examples applications are provided in `/examples/`. All other folders only contain files used internally in PESTO.

1.7 Features

PESTO implements a number of state-of-the-art algorithms related to parameter estimations. The main features are described below. Various Examples demonstrate their application.

Notations and Terminology

Since most of the examples use analytical approaches for computing the gradient of the respective objective function, which quantifies the deviation of the fit for the current model parameters from the actual measurement data, the usage of the term ‚sensitivity analysis‘ may be misleading. In our context, ‚sensitivity analysis‘ is used in the context of ODE or PDE models and describes the sensitivity of the ODE/PDE state with respect to the model parameters. Those state sensitivities can be implemented in the ODE/PDE system and then used for an analytical calculation of the sensitivity of the objective function. This objective functions sensitivity will always be called the objective function gradient in our context. Finally, the behavior of the objective function by the variation of single parameters in order to find possible (non-)identifiabilities will always be referred to as ‚uncertainty analysis‘.

1.7.1 Global optimization

Non-linear optimization problems like those in parameter estimation problems tend to have multiple optima. Usually, nothing is known beforehand about their number or their location, but the user is interested in finding the global optimum. There are different techniques for this kind of problem. PESTO provides a multi-start local optimization framework and provides an interface to two global optimizers.

Multi-start local optimization

Multi-start local optimization has turned out to be a very efficient method for "global optimization": Here, random points from across the parameter space are chosen as starting points for local optimization. If an adequate number of starting points spanning the domain of interest of the parameter space is selected, the lowest/highest minimum/maximum is accepted to be the global minimum/maximum. By default, fmincon is used as a local solver.

This functionality is provided in getMultiStarts.m, getPropertyMultiStarts.m and the respective plotting routines plotMultiStarts.m and plotPropertyMultiStarts.m. See mainConversionReaction.m for an example.

Global optimizers

PESTO provides an interface to `PSwarm` and `MEIGO`. Once these toolboxes have been installed - they are not included in the PESTO archive - they can be used for parameter estimation. These optimizers are also accessed via getMultiStarts.m by setting PestoOptions::localOptimizer and PestoOptions::localOptimizerOptions accordingly. In principle, a single optimizer run (PestoOptions::n_starts = 1) should be enough for these global optimizers.

An example is included in mainConversionReaction.m.

1.7.2 Uncertainty analysis

When parameters are inferred from measurement data, the deviation of the data from the fit for the best parameter guess is usually supposed to be of stochastic nature. This means that the estimated parameters themselves underly are stochastic and underly an uncertainty. This can be quantified by performing uncertainty analysis and computing confidence intervals.

The easiest way to do this is using local approximations (based on the Hessian matrix of the objective function) at the best parameter guess. From those approximations, either threshold-based or mass-based methods can be used to compute confidence intervals for the inferred parameters. Another approach uses sampling based methods in combination with local approximations.

The most reliable way to compute confidence intervals is a third approach, based on profile likelihoods. Here, each model parameter is varied separately while the others are constantly reoptimized. In this way one finds profiles for every parameter. By fixing a confidence level using the inverse chi-squared-distribution, one gets a threshold which, together with the profile likelihood, gives reliable confidence intervals for each parameter. In this way, non-identifiable parameter can be found.

Those functionalities are provided in getParameterProfiles.m, getPropertyProfiles.m (for the profile likelihoods), getParameterConfidenceIntervals.m and getPropertyConfidenceIntervals.m (for the confidence intervals). In order to get confidence intervals based on local approximations or sampling methods, one needs to run the routines getMultiStart.m/getPropertyMultiStarts.m or getParameterSamples.m/getPropertySamples.m first. The respective visualization routines are plotParameterProfiles.m and plotPropertyProfiles.m. See mainConversionReaction.m for an example.

1.7.3 Parameter sampling

PESTO provides Markov Chain Monte Carlo (MCMC) algorithms for sampling the posterior distribution. Sampling methods such as the `Metropolis-Hastings (MH)`, adaptive Metropolis (AM) and `Metropolis-adjusted Langevin algorithm (MALA)` are currently implemented. Additionally, PE↩
STO provides an interface to the `Delayed Rejection Adaptive Metropolis (DRAM)` toolbox.

See getParameterSamples() for details and mainConversionReaction.m for examples.

1.7.4 Plotting

An integral part of PESTO are its highly customizable plotting functions for each type of analysis.

Details are provided in the documentation of the specific plotting functions:

- plotMultiStarts.m
- plotParameterProfiles.m
- plotParameterSamples.m
- plotParameterUncertainty.m
- plotPropertyMultiStarts.m
- plotPropertyProfiles.m
- plotPropertySamples.m
- plotPropertyUncertainty.m

Here some examples:

Plot of model fit using plotMultiStarts.m:

Plot of different variants of parameter confidence intervals using plotParameterUncertainty.m:

2D plot of parameter samples using plotParameterSamples.m:

Plot of parameter samples using plotParameterSamples.m:

Plot of property samples using plotPropertySamples.m:

See mainConversionReaction.m for live examples.

1.7.5 Properties

The above-mentioned methods for parameter estimation, confidence intervals, parameter profiles and parameter samples (getMultiStarts.m, getParameterConfidenceIntervals.m, getParameterProfiles.m, getParameterSamples.↩
m) all operate on the objective function parameters directly. However, sometimes not the parameters themselves, but some function thereof is of interest. To this end, PESTO provides a simple interface to achieve this without having to change the objective function. Arbitrary user-provided functions which take the objective function parameter vector as an argument are referred to as 'properties'. After having used any of the getParameter∗.m functions, the respective getProperty∗.m function can be called, to evaluate a user-provided property function with the parameters values/samples/confidences obtained from the getParameter∗.m functions.

The following functions are available to analyze and plot properties:

- getPropertyConfidenceIntervals.m
- getPropertyMultiStarts.m
- getPropertyProfiles.m
- getPropertySamples.m
- plotPropertyMultiStarts.m
- plotPropertyProfiles.m
- plotPropertySamples.m
- plotPropertyUncertainty.m

See mainConversionReaction.m for examples.

2 Examples

2.1 Examples

2.1.1 Overview

PESTO comes with a number of ready-to-run examples to demonstrate its usage and functionality. All examples come from the life science field. The are examples based on ordinary and partial differential equation (ODE / PDE), and Gaussian mixture models. More background information is provided in the respective example folder in the `main.m` script.

The following examples are included:

- Conversion reaction (`examples/conversion_reaction/runEstimation__CR.m`, ODE)
- Enzymatic catalysis (`examples/enzymatic_catalysis/`, ODE)
- Gaussian mixture (`examples/Gaussian_mixture/`)
- mRNA transfection § (`examples/mRNA_transfection/`, ODE)
- Pom1p gradient formation § (`examples/Pom1p_gradient_formation/`, PDE)
- TODO: JAK/STAT signaling (`...`, ODE)

§ These models require the freely available AMICI toolbox to run (`http://icb-dcm.github.io/AMICI/`).

The following table provides an overview of which of the PESTO functions are demonstrated in the different examples:

	conversion_reaction	enzymatic_catalysis	Gaussian_mixture	mRNA transfection	Pom1p_gradient_formation
getMultiStarts()	X	X	X	X	X
getParameterConfidenceIntervals()	X	X	X	X	X
getParameterProfiles()	X	X	X	X	X
getPropertyConfidenceIntervals()	X	X	X	X	X
getPropertyMultiStarts()	X	X	X	X	X
getPropertyProfiles()	X	X	X	X	X
getPropertySamples()	X	X	X	X	X
plotMultiStarts()	X	X	X	X	X
plotParameterProfiles()	X	X	X	X	X
plotParameterSamples()	X	X	X	X	X

	conversion_↩ reaction	enzymatic_↩ catalysis	Gaussian_↩ mixture	mRNA transfection	Pom1p_↩ gradient_↩ formation
plot↩ Parameter↩ Uncertainty()	X	X	X	X	X
plotProperty↩ MultiStarts()	X	X	X	X	X
plotProperty↩ Profiles()	X	X	X	X	X
plotProperty↩ Samples()	X	X	X	X	X
plotProperty↩ Uncertainty()	X	X	X	X	X
testGradient()	X	X	X	X	X
collectResults()	X	X	X	X	X

3 Changes in PESTO Version 1.2

Class MatlabDocMaker

(dw, 2011-11-27)

- Included documentation creation for the Windows platform and combined the old methods into one (small effective differences)
- No longer storing the doxygen binary file in the prefs as a lot of tools must be present on the path anyways. The new paradigm is to expect all required 3rd-party programmes to be available on PATH. As backup the configuration files directory is added to the Matlab PATH environment **nonpermanently** and any executables found there will thus also be usable.
- Included checks for `dot` and `latex` at the setup stage to recommend installation of those tools if not present (the default doxygen settings in Doxyfile.m4 are to use both)

(dw, 2011-11-08) Improved the createUnix method by displaying the warnings and writing the output to a log file afterwards. Not using cprintf anymore as this is 3rd party software.

(dw, 2011-11-07) Fixed a recursion bug caused by copy and paste. Now the preferences are stored on an per-application basis.

(dw, 2011-11-04) Changed the name to MatlabDocMaker in order to export it into the mtoc++ distribution later.

4 Changes in PESTO Version 1.3

Class MatlabDocMaker

(dw, 2012-02-16)

- Now also collecting error messages from mtocpp_post and adding them to the warnings.log file.
- Added the directive "LD_LIBRARY_PATH= " for unix systems, as MatLab sets it inside its executing environment. This can lead to errors if doxygen and/or mtoc++ have been built using never GLIBC (libstd) versions than the one shipped with MatLab.

(dw, 2012-01-16)

- Properly using the correct file separators everywhere now
- Hyperlinked the log file so it can be opened directly

(dw, 2012-01-14) Not displaying the "generated warnings"-text if there have not been any during documentation creation.

5 Changes in PESTO Version 1.4

Class MatlabDocMaker

(dw, 2012-10-18) Removed `m4` dependency and included constant properties for configuration file names.

(dw, 2012-09-27) Added automatic dot Graphviz tool detection on call to create.

6 Changes in PESTO Version 1.5

Class MatlabDocMaker

(dw, 2013-12-03) Fixed default value selection for properties, now not having set a description or logo does not cause an error to be thrown.

(dw, 2013-02-21) Fixed the callback for suggested direct documentation creation after MatlabDocMaker.setup (Thanks to Aurelien Queffurust)

(dw, 2013-02-12) Also added the escaping for the Logo file. Thanks to Chris Olien for the hint.

(dw, 2013-01-07) Included some backslash escaping for paths on windows platforms. Thanks to MathWorks Pilot Engineer '`Arvind Jayaraman`' for providing the feedback and code!

7 New features in PESTO Version 1.2

Class MatlabDocMaker

(dw, 2011-10-13) Added this class and moved documentation related stuff here from the KerMor class.

8 New features in PESTO Version 1.4

Class MatlabDocMaker

(dw, 2012-10-16)

- Added two more configuration variables "ProjectDescription" and "ProjectLogo" for easier configuration of the MatlabDocMaker in many cases. Thanks to Wolfgang Mennerich `http://www.mathworks.↩ com/matlabcentral/fileexchange/authors/272859` for the suggestion.
- Restructured the configuration, now only the project name function has to be implemented (the preferences tag depends on it, there might be more than one project within the same Matlab installation whos documentation is created using this tool). The rest can be provided either at setup time or later via suitable setters for the version, description and logo.
- Automatically setting HaveDot in the doxygen config whenever its found on the environment path.
- Added basic support for LaTeX documentation creation. Using the parameter `latex`=true for the create method creates the LaTeX version of the documentation in a folder "latex" in the OutputDirectory (default behaviour)

9 Hierarchical Index

9.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

SetGet

10 Class Index

10.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

11 File Index

11.1 File List

Here is a list of all documented files with brief descriptions:

12 Class Documentation

12.1 MatlabDocMaker Class Reference

MatlabDocMaker: Automated documentation creation using doxygen and mtoc++ from within MatLab.

Static Public Member Functions

- static mlhsInnerSubst< ::char, name > getProjectName ()

  Returns the project name.
- static mlhsInnerSubst< ::char, dir > getOutputDirectory ()

  Returns the directory where the applications source files reside.
- static mlhsInnerSubst< ::char, dir > getSourceDirectory ()

  Returns the directory where the applications source files reside.
- static mlhsInnerSubst< ::char, dir > getConfigDirectory ()

  Returns the directory where the applications documentation configuration files reside.
- static mlhsInnerSubst< ::char, desc > getProjectDescription ()

  Returns the short project description.
- static noret::substitute setProjectDescription (::char value)

  Sets the project description.
- static mlhsInnerSubst< ::char, version > getProjectVersion ()

  Returns the current version of the project.
- static noret::substitute setProjectVersion (::char value)

  Sets the project version.
- static mlhsInnerSubst< matlabtypesubstitute, fullPath > getProjectLogo ()

  Returns the logo image file for the project. Either an absolute path or a plain filename. For the latter case the image file is assumed to reside inside the directory returned by MatlabDocMaker.getConfigDirectory.
- static noret::substitute setProjectLogo (::char value)

  Sets the project logo. Set to ' to unset.
- static noret::substitute open ()

  Opens the generated documentation.
- static noret::substitute create (matlabtypesubstitute varargin)

  Creates the Doxygen documentation.
- static noret::substitute setup ()

  Runs the setup script for MatlabDocMaker and collects all necessary paths in order for the documentation creation to work properly.

Static Public Attributes

- static const ::char DOXYFILE_TEMPLATE = "Doxyfile.template"

  File name for the doxygen configuration file processed by the MatlabDocMaker.
- static const ::char LATEXEXTRAS_TEMPLATE = "latexextras.template"

  File name for the latex extras style file processed by the MatlabDocMaker.
- static const ::char MTOCPP_CONFIGFILE = "mtocpp.conf"

  File name the mtoc++ configuration file.

12.1.1 Detailed Description

MatlabDocMaker: Automated documentation creation using doxygen and mtoc++ from within MatLab.

Currently documentation creation for unix and windows environment is supported.

Prerequisites

The following tools must be installed and present on the PATH or reside inside the folder returned by Matlab←DocMaker.getConfigDirectory.

- `mtocpp`, `mtocpp_post` (the main tool)
- `doxygen` (mtoc++ is a filter for doxygen)

Strongly recommended

- `latex` Doxygen supports built-in latex formulas and MatlabDocMaker/mtoc++ allows for easy extra latex inclusions and notation in code
- `dot` Doxygen creates really nice inheritance graphs and collaboration diagrams with dot.

Author

Daniel Wirtz

Date

2011-10-13

Change in 1.5 (dw, 2013-12-03) Fixed default value selection for properties, now not having set a description or logo does not cause an error to be thrown.

Change in 1.5 (dw, 2013-02-21) Fixed the callback for suggested direct documentation creation after MatlabDoc←Maker.setup (Thanks to Aurelien Queffurust)

Change in 1.5 (dw, 2013-02-12) Also added the escaping for the Logo file. Thanks to Chris Olien for the hint.

Change in 1.5 (dw, 2013-01-07) Included some backslash escaping for paths on windows platforms. Thanks to MathWorks Pilot Engineer '`Arvind Jayaraman`' for providing the feedback and code!

Change in 1.4 (dw, 2012-10-18) Removed `m4` dependency and included constant properties for configuration file names.

New in 1.4 (dw, 2012-10-16)

- Added two more configuration variables "ProjectDescription" and "ProjectLogo" for easier configuration of the MatlabDocMaker in many cases. Thanks to Wolfgang Mennerich `http://www.mathworks.com/matlabcentral/fileexchange/authors/272859` for the suggestion.
- Restructured the configuration, now only the project name function has to be implemented (the preferences tag depends on it, there might be more than one project within the same Matlab installation whos documentation is created using this tool). The rest can be provided either at setup time or later via suitable setters for the version, description and logo.
- Automatically setting HaveDot in the doxygen config whenever its found on the environment path.
- Added basic support for LaTeX documentation creation. Using the parameter `latex`=true for the create method creates the LaTeX version of the documentation in a folder "latex" in the OutputDirectory (default behaviour)

Change in 1.4 (dw, 2012-09-27) Added automatic dot Graphviz tool detection on call to create.

Change in 1.3 (dw, 2012-02-16)

- Now also collecting error messages from mtocpp_post and adding them to the warnings.log file.
- Added the directive "LD_LIBRARY_PATH= " for unix systems, as MatLab sets it inside its executing environment. This can lead to errors if doxygen and/or mtoc++ have been built using never GLIBC (libstd) versions than the one shipped with MatLab.

Change in 1.3 (dw, 2012-01-16)

- Properly using the correct file separators everywhere now
- Hyperlinked the log file so it can be opened directly

Change in 1.3 (dw, 2012-01-14) Not displaying the "generated warnings"-text if there have not been any during documentation creation.

Change in 1.2 (dw, 2011-11-27)

- Included documentation creation for the Windows platform and combined the old methods into one (small effective differences)
- No longer storing the doxygen binary file in the prefs as a lot of tools must be present on the path anyways. The new paradigm is to expect all required 3rd-party programmes to be available on PATH. As backup the configuration files directory is added to the Matlab PATH environment **nonpermanently** and any executables found there will thus also be usable.
- Included checks for `dot` and `latex` at the setup stage to recommend installation of those tools if not present (the default doxygen settings in Doxyfile.m4 are to use both)

Change in 1.2 (dw, 2011-11-08) Improved the createUnix method by displaying the warnings and writing the output to a log file afterwards. Not using cprintf anymore as this is 3rd party software.

Change in 1.2 (dw, 2011-11-07) Fixed a recursion bug caused by copy and paste. Now the preferences are stored on an per-application basis.

Change in 1.2 (dw, 2011-11-04) Changed the name to MatlabDocMaker in order to export it into the mtoc++ distribution later.

New in 1.2 (dw, 2011-10-13) Added this class and moved documentation related stuff here from the KerMor class.

This class is part of the mtoc++ tool

- Homepage http://www.morepas.org/software/mtocpp/
- License http://www.morepas.org/software/mtocpp/docs/licensing.html

Definition at line 17 of file MatlabDocMaker.m.

12.1.2 Member Function Documentation

12.1.2.1 noret::substitute MatlabDocMaker::create (matlabtypesubstitute *varargin*) [static]

Creates the Doxygen documentation.

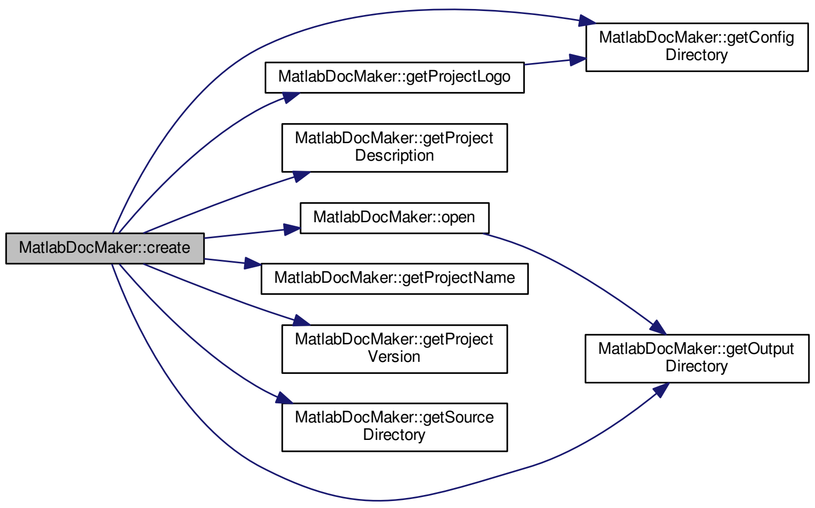
Parameters

varargin	Optional parameters for creation. `create ( [ "open", open_value ] [, "latex", latex_value ] )` *Named Parameters for varargin:* • open Set to true if the documentation should be opened after successful compilation **Default:** false • latex Set to true if LaTeX output should be generated, too. **Default:** false

Definition at line 412 of file MatlabDocMaker.m.

References DOXYFILE_TEMPLATE, getConfigDirectory(), getOutputDirectory(), getProjectDescription(), get←
ProjectLogo(), getProjectName(), getProjectVersion(), getSourceDirectory(), LATEXEXTRAS_TEMPLATE, MT←
OCPP_CONFIGFILE, and open().

Here is the call graph for this function:


MatlabDocMaker::create
MatlabDocMaker::getConfig Directory
MatlabDocMaker::getProjectLogo
MatlabDocMaker::getProject Description
MatlabDocMaker::open
MatlabDocMaker::getProjectName
MatlabDocMaker::getProject Version
MatlabDocMaker::getOutput Directory
MatlabDocMaker::getSource Directory

12.1.2.2 mlhsInnerSubst< ::char, dir > MatlabDocMaker::getConfigDirectory ( ) `[static]`

Returns the directory where the applications documentation configuration files reside.

This folder must contain at least the files "mtoc.conf" and "Doxyfile.template"

Return values

dir	The documentation configuration directory

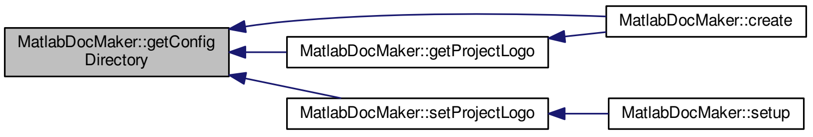
Note

This method has the MATLAB method attribute `Sealed` set to true. It cannot be overwritten. `matlab documentation of method attributes.`

Definition at line 225 of file MatlabDocMaker.m.

Referenced by create(), getProjectLogo(), and setProjectLogo().

Here is the caller graph for this function:


MatlabDocMaker::getConfig Directory
MatlabDocMaker::create
MatlabDocMaker::getProjectLogo
MatlabDocMaker::setProjectLogo
MatlabDocMaker::setup

12.1.2.3 mlhsInnerSubst< ::char, dir > MatlabDocMaker::getOutputDirectory ( ) [static]

Returns the directory where the applications source files reside.

Return values

dir	The output directory

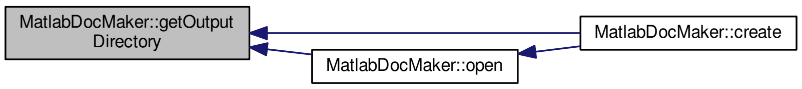
Note

This method has the MATLAB method attribute `Sealed` set to true. It cannot be overwritten. `matlab documentation of method attributes.`

Definition at line 195 of file MatlabDocMaker.m.

Referenced by create(), and open().

Here is the caller graph for this function:



12.1.2.4 mlhsInnerSubst<::char, desc > MatlabDocMaker::getProjectDescription ( ) `[static]`

Returns the short project description.

See also

setProjectDescription

Return values

desc	The short project description

Default:

[]

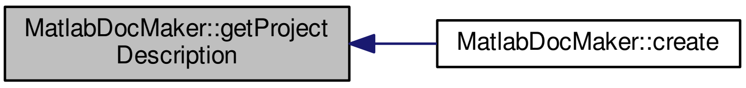
Note

This method has the MATLAB method attribute `Sealed` set to true. It cannot be overwritten. `matlab documentation of method attributes.`

Definition at line 244 of file MatlabDocMaker.m.

Referenced by create().

Here is the caller graph for this function:



12.1.2.5 mlhsInnerSubst< matlabtypesubstitute, fullPath > MatlabDocMaker::getProjectLogo ( ) [static]

Returns the logo image file for the project. Either an absolute path or a plain filename. For the latter case the image file is assumed to reside inside the directory returned by MatlabDocMaker.getConfigDirectory.

See also

setProjectLogo

Return values

logoFile	The projects logo image file.

Default:

[]

Note

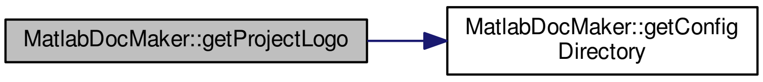
This method has the MATLAB method attribute `Sealed` set to true. It cannot be overwritten. `matlab documentation of method attributes.`

Definition at line 317 of file MatlabDocMaker.m.

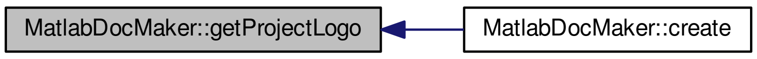
References getConfigDirectory().

Referenced by create().

Here is the call graph for this function:


MatlabDocMaker::getProjectLogo
MatlabDocMaker::getConfig Directory

Here is the caller graph for this function:



12.1.2.6 mlhsInnerSubst< ::char, name > MatlabDocMaker::getProjectName ( ) [static]

Returns the project name.

Note

Changing the return value of this method will require another execution of MatlabDocMaker.setup as the preferences storage key also depends on it.

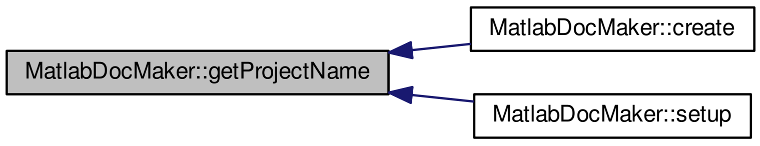
Return values

name	The project name

Definition at line 170 of file MatlabDocMaker.m.

Referenced by create(), and setup().

Here is the caller graph for this function:



12.1.2.7 mlhsInnerSubst< ::char, version > MatlabDocMaker::getProjectVersion ( ) [static]

Returns the current version of the project.

Note

The built-in @new and @change tags from the Doxyfile.template support two-level versioning a la X.X.

See also

setProjectVersion

Return values

version	The project version

Default:

[]

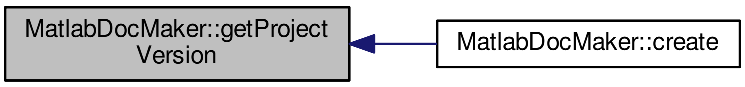
Note

This method has the MATLAB method attribute `Sealed` set to true. It cannot be overwritten. `matlab documentation of method attributes.`

Definition at line 279 of file MatlabDocMaker.m.

Referenced by create().

Here is the caller graph for this function:



12.1.2.8 mlhsInnerSubst<::char, dir > MatlabDocMaker::getSourceDirectory ( ) `[static]`

Returns the directory where the applications source files reside.

Return values

dir	The project source directory

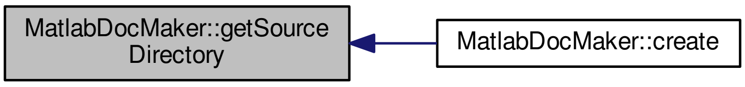
Note

This method has the MATLAB method attribute `Sealed` set to true. It cannot be overwritten. `matlab documentation of method attributes.`

Definition at line 210 of file MatlabDocMaker.m.

Referenced by create().

Here is the caller graph for this function:



12.1.2.9 noret::substitute MatlabDocMaker::open ( ) `[static]`

Opens the generated documentation.

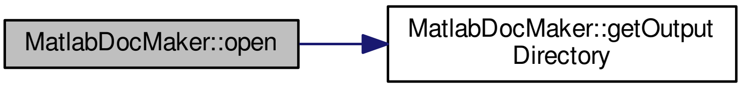
Depending on the systems `type the generated index.html is opened in the systems` default browser.

Definition at line 391 of file MatlabDocMaker.m.

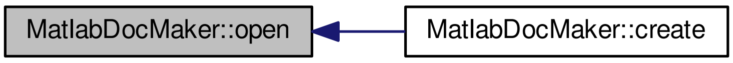
References getOutputDirectory().

Referenced by create().

Here is the call graph for this function:



Here is the caller graph for this function:



12.1.2.10 noret::substitute MatlabDocMaker::setProjectDescription (::char *value*) `[static]`

Sets the project description.

See also

getProjectDescription

Parameters

value	The description

Note

This method has the MATLAB method attribute `Sealed` set to true. It cannot be overwritten.
`matlab documentation of method attributes.`

Definition at line 260 of file MatlabDocMaker.m.

12.1.2.11 noret::substitute MatlabDocMaker::setProjectLogo (::char *value*) `[static]`

Sets the project logo. Set to ' to unset.

See the doxygen documentation for valid logo file types (wont be checked here).

See also

getProjectLogo

Parameters

value	The logo file to use. Must be either an absolute path or a plain filename, in which case the image is assumed to reside inside the MatlabDocMaker.getConfigDirectory directory.

Note

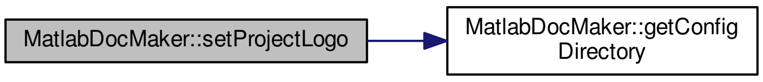
This method has the MATLAB method attribute `Sealed` set to true. It cannot be overwritten.
`matlab documentation of method attributes.`

Definition at line 347 of file MatlabDocMaker.m.

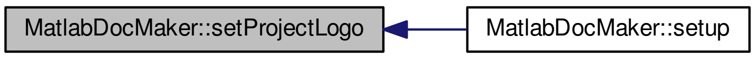
References getConfigDirectory().

Referenced by setup().

Here is the call graph for this function:


MatlabDocMaker::setProjectLogo
MatlabDocMaker::getConfig Directory

Here is the caller graph for this function:



12.1.2.12 noret::substitute MatlabDocMaker::setProjectVersion (::char *value*) `[static]`

Sets the project version.

See also

getProjectVersion

Parameters

value	The version string

Note

This method has the MATLAB method attribute `Sealed` set to true. It cannot be overwritten.
`matlab documentation of method attributes.`

Definition at line 298 of file MatlabDocMaker.m.

12.1.3 Member Data Documentation

12.1.3.1 MatlabDocMaker::DOXYFILE_TEMPLATE = "Doxyfile.template" `[static]`

File name for the doxygen configuration file processed by the MatlabDocMaker.

Assumed to reside in the MatlabDocMaker.getConfigDirectory

Default: `Doxyfile.template`

Definition at line 130 of file MatlabDocMaker.m.

Referenced by create().

12.1.3.2 MatlabDocMaker::LATEXEXTRAS_TEMPLATE = "latexextras.template" `[static]`

File name for the latex extras style file processed by the MatlabDocMaker.

Assumed to reside in the MatlabDocMaker.getConfigDirectory. If not found, no latex extras are used.

Default: `latexextras.template`

Definition at line 142 of file MatlabDocMaker.m.

Referenced by create().

12.1.3.3 MatlabDocMaker::MTOCPP_CONFIGFILE = "mtocpp.conf" `[static]`

File name the mtoc++ configuration file.

Assumed to reside in the MatlabDocMaker.getConfigDirectory. If not found, no special configuration is used.

Default: `mtocpp.conf`

Definition at line 155 of file MatlabDocMaker.m.

Referenced by create().

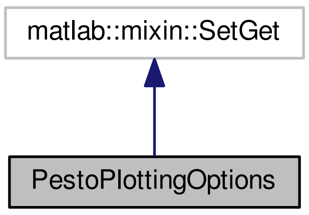
The documentation for this class was generated from the following file:

- doc/MatlabDocMaker.m

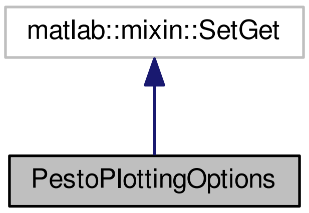
12.2 PestoPlottingOptions Class Reference

PestoPlottingOptions is class for checking and holding information on optimization parameters.

Inheritance diagram for PestoPlottingOptions:



Collaboration diagram for PestoPlottingOptions:



Public Member Functions

- PestoPlottingOptions (matlabtypesubstitute varargin)

  PestoPlottingOptions Construct a new PestoPlottingOptions object.
- mlhsInnerSubst< matlabtypesubstitute, new > **copy** ()

Public Attributes

- matlabtypesubstitute title = true

  Title of PESTO-generated plots.
- matlabtypesubstitute add_points

  Additional points to include in the plots, e.g. true parameter in the case of test examples.
- matlabtypesubstitute mark_constraint = false

  TODO: from plotmultistarts.
- matlabtypesubstitute subplot_size_1D = "[ ]"

  TODO from plotparameteruncertainty.
- matlabtypesubstitute subplot_indexing_1D = "[ ]"

  TODO.
- matlabtypesubstitute labels

  TODO.
- matlabtypesubstitute hold_on = false

  Indicates whether plots are redrawn or whether something is added to the plot.
- matlabtypesubstitute interval = "dynamic"

  Way of choosing x limits for plotting.
- matlabtypesubstitute draw_bounds = true

  Draw bounds.
- matlabtypesubstitute bounds = {""}

  Bounds used for visualization if options.interval = `static`
- matlabtypesubstitute P

  Options for profile plots.
- matlabtypesubstitute S

  Options for sample plots.
- matlabtypesubstitute MS

  Options for multi-start optimization plots.
- matlabtypesubstitute A

  Options for distribution approximation plots.
- matlabtypesubstitute MCMC = "multistart"

  Option if a user provided sampling initialization should be used for plotting an approximation of the distribution.
- matlabtypesubstitute boundary

  Options for boundary visualization.
- matlabtypesubstitute CL

  Options for confidence level plots.
- matlabtypesubstitute group_CI_by = "parprop"

  Options for the way to plot confidence intervals.
- matlabtypesubstitute op2D = struct("'b1', 0.15, 'b2', 0.02, 'r', 0.95")

  Settings for 2D plot to position subplot axes.
- matlabtypesubstitute legend

  Legend options.
- matlabtypesubstitute fontsize = struct ("'tick', 12")

  Fontsize for labels.

- matlabtypesubstitute fh_logPost_trace = "[ ]"

  figure handle for log-posterior trace plot
- matlabtypesubstitute fh_par_trace = "[ ]"

  figure handle for parameter trace plots.
- matlabtypesubstitute fh_par_dis_1D = "[ ]"

  figure handle for the parameter distribution plot. fh_par_dis = [];
- matlabtypesubstitute **fh_par_dis_2D** = "[ ]"
- matlabtypesubstitute **plot_type** = {"'parameter','posterior'"}
- matlabtypesubstitute n_max = 1e4

12.2.1 Detailed Description

PestoPlottingOptions is class for checking and holding information on optimization parameters.

This file is based on AMICI amioptions.m (`http://icb-dcm.github.io/AMICI/`)

Definition at line 17 of file PestoPlottingOptions.m.

12.2.2 Constructor & Destructor Documentation

12.2.2.1 PestoPlottingOptions::PestoPlottingOptions (matlabtypesubstitute *varargin*)

PestoPlottingOptions Construct a new PestoPlottingOptions object.

OPTS = PestoPlottingOptions() creates a set of options with each option set to its default value.

OPTS = PestoPlottingOptions(PARAM, VAL, ...) creates a set of options with the named parameters altered with the specified values.

OPTS = PestoPlottingOptions(OLDOPTS, PARAM, VAL, ...) creates a copy of OLDOPTS with the named parameters altered with the specified value

Note to see the parameters, check the documentation page for PestoPlottingOptions

Definition at line 472 of file PestoPlottingOptions.m.

References draw_bounds, group_CI_by, hold_on, interval, mark_constraint, MCMC, n_max, and title.

12.2.3 Member Data Documentation

12.2.3.1 PestoPlottingOptions::A

Initial value:

```
= struct("'plot_type', 1,     \
          'col', [0,0,1],     \
          'lw', 2,     \
          'sigma_level', 2,     \
          'name', 'P_{app}'")
```

Options for distribution approximation plots.

Struct with

- .plot_type: plot type
  - = 0 (default if no MS are provided) ... no plot
  - = 1 (default if MS are provided) ... likelihood ratio
  - = 2 ... negative log-likelihood
- .col: color of approximation lines (default: [0,0,1])
- .lw: line width of approximation lines (default: 1.5)
- .sigma_level: sigma-level which is visualized (default = 2)
- .name: name of legend entry (default = `P_{app}`)

Default: struct("'plot_type', 1, \ 'col', [0,0,1], \ 'lw', 2, \ 'sigma_level', 2, \ 'name', 'P_{app}'")

Definition at line 277 of file PestoPlottingOptions.m.

12.2.3.2 PestoPlottingOptions::add_points

Initial value:

```
= struct("'par', [],     \
          'logPost', [],     \
          'col', [0,0.8,0],     \
          'ls', '-',     \
          'lw', 1,     \
          'm', 'd',     \
          'ms', 8,     \
          'name', 'add. point'")
```

Additional points to include in the plots, e.g. true parameter in the case of test examples.

Struct with the following fields

- .par: n x m matrix of m additional points
- .col: color used for additional points (default = [0,0,0]). This can also be a m x 3 matrix of colors.
- .ls: line style (default = `-`)
- .lw: line width (default = 2)
- .m: marker style (default = `s`)
- .ms: line width (default = 8)
- .name: name of legend entry (default = `add.  point`)
- .property_MS: line width (default = 8).
- .logPost

Default: struct("'par', [], \ 'logPost', [], \ 'col', [0,0.8,0], \ 'ls', '-', \ 'lw', 1, \ 'm', 'd', \ 'ms', 8, \ 'name', 'add. point'")

Definition at line 42 of file PestoPlottingOptions.m.

12.2.3.3 PestoPlottingOptions::boundary

Initial value:

```
= struct("'mark', true,    \
           'eps', 1e-4")
```

Options for boundary visualization.

Struct with

- .mark: marking of profile points which are on the boundary
  - = 0 ... no visualization
  - = 1 (default) ... indicates points which ar close to the boundaries in one or more dimensions.
- .eps: minimal distance from boundary for which points are consider to e close do the boundary (default = 1e-4). Note that a one-norm is used.

Default: struct("'mark', true, \ 'eps', 1e-4")

Definition at line 318 of file PestoPlottingOptions.m.

12.2.3.4 PestoPlottingOptions::bounds = {""}

Bounds used for visualization if options.interval = `static`

struct with

- .min: lower bound
- .max: upper bound

Default: {""}

Definition at line 152 of file PestoPlottingOptions.m.

12.2.3.5 PestoPlottingOptions::CL

Initial value:

```
= struct("'plot_type', 0,    \
           'alpha', 0.95,    \
           'type', 'point-wise',    \
           'col', [0,0,0],    \
           'lw', 2,     \
           'name', 'cut-off'")
```

Options for confidence level plots.

Struct with

- .plot_type: plot type
  - = 0 (default) ... no plot
  - = 1 ... likelihood ratio
  - = 2 ... negative log-likelihood
- .alpha: visualized confidence level (default = 0.95)
- .type: type of confidence interval
  - = `point-wise` (default) ... point-wise confidence interval
  - = `simultanous` ... point-wise confidence interval
  - = {`point-wise`,`simultanous`} ... both
- .col: color of profile lines (default: [0,0,0])
- .lw: line width of profile lines (default: 1.5)
- .name: name of legend entry (default = `cut-off`)

Default: struct("'plot_type', 0, \ 'alpha', 0.95, \ 'type', 'point-wise', \ 'col', [0,0,0], \ 'lw', 2, \ 'name', 'cut-off'")

Definition at line 339 of file PestoPlottingOptions.m.

12.2.3.6 PestoPlottingOptions::draw_bounds = true

Draw bounds.

- true: yes
- false: no

Default: true

Note

This property has custom functionality when its value is changed.

Definition at line 141 of file PestoPlottingOptions.m.

Referenced by PestoPlottingOptions().

12.2.3.7 PestoPlottingOptions::fh_logPost_trace = "[ ]"

figure handle for log-posterior trace plot

Default: "[ ]"

Definition at line 430 of file PestoPlottingOptions.m.

12.2.3.8 PestoPlottingOptions::fh_par_dis_1D = "[ ]"

figure handle for the parameter distribution plot. fh_par_dis = [];

Default: "[ ]"

Definition at line 448 of file PestoPlottingOptions.m.

12.2.3.9 PestoPlottingOptions::fh_par_trace = "[ ]"

figure handle for parameter trace plots.

Default: "[ ]"

Definition at line 439 of file PestoPlottingOptions.m.

12.2.3.10 PestoPlottingOptions::fontsize = struct ("'tick', 12")

Fontsize for labels.

- .tick: fontsize for ticklabels (default = 12)

Default: struct ("'tick', 12")

Definition at line 420 of file PestoPlottingOptions.m.

12.2.3.11 PestoPlottingOptions::group_CI_by = "parprop"

Options for the way to plot confidence intervals.

Either all confidence intervals of one method are plotted to one window `params`, or the confidence intervals for one parameter from all methods are plotted to one window `methods`, or everthing is grouped together `all`.

Default: "parprop"

Note

This property has custom functionality when its value is changed.

Definition at line 373 of file PestoPlottingOptions.m.

Referenced by PestoPlottingOptions().

12.2.3.12 PestoPlottingOptions::hold_on = false

Indicates whether plots are redrawn or whether something is added to the plot.

- true: extension of plot
- false: new plot

Default: false

Note

This property has custom functionality when its value is changed.

Definition at line 116 of file PestoPlottingOptions.m.

Referenced by PestoPlottingOptions().

12.2.3.13 PestoPlottingOptions::interval = "dynamic"

Way of choosing x limits for plotting.

- `dynamic`: x limits depending on analysis results
- `static`: x limits depending on parameters.min and .max or on user-defined bound options.bounds.min and .max. The later are used if provided.

Default: "dynamic"

Note

This property has custom functionality when its value is changed.

Definition at line 128 of file PestoPlottingOptions.m.

Referenced by PestoPlottingOptions().

12.2.3.14 PestoPlottingOptions::labels

Initial value:

```
= struct("'y_always', true,    \
          'y_name', []")
```

TODO.

Default: struct("'y_always', true, \ 'y_name', [ ]")

Definition at line 105 of file PestoPlottingOptions.m.

12.2.3.15 PestoPlottingOptions::legend

Initial value:

```
= struct("'color', 'none',     \
           'box', 'on',     \
           'orientation', 'vertical',     \
           'position', []")
```

Legend options.

- .color: background color (default = `none`).
- .box: legend outine (default = `on`).
- .orientation: orientation of list (default = `vertical`)

Default: struct("'color', 'none', \ 'box', 'on', \ 'orientation', 'vertical', \ 'position', []")

Definition at line 402 of file PestoPlottingOptions.m.

12.2.3.16 PestoPlottingOptions::mark_constraint = false

TODO: from plotmultistarts.

Default: false

Note

This property has custom functionality when its value is changed.

Definition at line 78 of file PestoPlottingOptions.m.

Referenced by PestoPlottingOptions().

12.2.3.17 PestoPlottingOptions::MCMC = "multistart"

Option if a user provided sampling initialization should be used for plotting an approximation of the distribution.

- `user-provided`
- `multistart` (default)

Default: "multistart"

Note

This property has custom functionality when its value is changed.

Definition at line 305 of file PestoPlottingOptions.m.

Referenced by PestoPlottingOptions().

12.2.3.18 PestoPlottingOptions::MS

Initial value:

```
= struct("'plot_type', 1,    \
           'col', [1,0,0],    \
           'lw' , 2,    \
           'name_conv', 'MS - conv.',    \
           'name_nconv', 'MS - not conv.',    \
           'only_optimum', false")
```

Options for multi-start optimization plots.

Struct with

- .plot_type: plot type
  - = 0 (default if no MS are provided) ... no plot
  - = 1 (default if MS are provided) ... likelihood ratio and position of optima above threshold
  - = 2 ... negative log-likelihood and position of optima above threshold
- .col: color of local optima (default: [1,0,0])
- .lw: line width of local optima (default: 1.5)
- .name_conv: name of legend entry (default = `MS - conv.`)
- .name_nconv: name of legend entry (default = `MS - not conv.`)
- .only_optimum: only optimum is plotted

Default: struct("'plot_type', 1, \ 'col', [1,0,0], \ 'lw' , 2, \ 'name_conv', 'MS - conv.', \ 'name_nconv', 'MS - not conv.', \ 'only_optimum', false")

Definition at line 244 of file PestoPlottingOptions.m.

12.2.3.19 PestoPlottingOptions::n_max = 1e4

Note

This property has custom functionality when its value is changed.

Definition at line 462 of file PestoPlottingOptions.m.

Referenced by PestoPlottingOptions().

12.2.3.20 PestoPlottingOptions::op2D = struct("'b1', 0.15, 'b2', 0.02, 'r', 0.95")

Settings for 2D plot to position subplot axes.

Struct with

- .b1 ... offset from left and bottom border (default = 0.15)
- .b2 ... offset from left and bottom border (default = 0.02)
- .r ... relative width of subplots (default = 0.95)

Default: struct("'b1', 0.15, 'b2', 0.02, 'r', 0.95")

Definition at line 387 of file PestoPlottingOptions.m.

12.2.3.21 PestoPlottingOptions::P

Initial value:

```
= struct("'plot_type', 1,    \
           'col', [1,0,0],    \
           'lw', 2,    \
           'name', 'P'")
```

Options for profile plots.

Struct with

- .plot_type: plot type
  - = 0 (default if no profiles are provided) ... no plot
  - = 1 (default if profiles are provided) ... likelihood ratio
  - = 2 ... negative log-likelihood
- .col: color of profile lines (default: [1,0,0])
- .lw: line width of profile lines (default: 1.5)

Default: struct("'plot_type', 1, \ 'col', [1,0,0], \ 'lw', 2, \ 'name', 'P'")

Definition at line 165 of file PestoPlottingOptions.m.

12.2.3.22 PestoPlottingOptions::S

Initial value:

```
= struct("'plot_type', 0,    \
           'bins', 'optimal',    \
           'scaling', [],    \
           'hist_col',  [0.7,0.7,0.7],    \
           'sp_col', [0.7,0.7,0.7],    \
           'lin_col', [1,0,0],    \
           'lin_lw', 2,    \
           'sp_m', '.',    \
           'sp_ms', 5,    \
           'col', [1,0,0], 'lw', 2,    \
           'PT', struct('sp_m', '.',    \
               'sp_ms', 5,    \
               'lw', 1.5,    \
               'ind', [],    \
               'col', [],    \
               'plot_type', 0),    \
           'name', 'S'")
```

Options for sample plots.

- .plot_type: plot type
  - = 0 (default if no samples are provided) ... no plot
  - = 1 (default if samples are provided) ... histogram
  - = 2 ... kernel-density estimates
- .col ... color of profile lines (default: [0.7,0.7,0.7])
- .hist_col ... color of histogram (default = [0.7,0.7,0.7])

- .bins ... number of histogram bins (default: 30)
  - = `optimal` ... selection using Scott's rule
  - = `conservative` ... selection using Scott's rule / 2
  - = N (with N being an integer) ... N bins
- .sp_col: color of scatter plot (default = [0.7,0.7,0.7])
- .sp_m: marker for scatter plot (default = `.`)
- .sp_ms: marker size for scatter plot (default = 5)
- .name: name of legend entry (default = `S`)

Default: struct("'plot_type', 0, \ 'bins', 'optimal', \ 'scaling', [], \ 'hist_col', [0.7,0.7,0.7], \ 'sp_col', [0.7,0.7,0.7], \ 'lin↩ _col', [1,0,0], \ 'lin_lw', 2, \ 'sp_m', '.', \ 'sp_ms', 5, \ 'col', [1,0,0], 'lw', 2, \ 'PT', struct('sp_m', '.', \ 'sp_ms', 5, \ 'lw', 1.5, \ 'ind', [], \ 'col', [], \ 'plot_type', 0), \ 'name', 'S'")

Definition at line 188 of file PestoPlottingOptions.m.

12.2.3.23 PestoPlottingOptions::subplot_indexing_1D = "[ ]"

TODO.

Default: "[ ]"

Definition at line 96 of file PestoPlottingOptions.m.

12.2.3.24 PestoPlottingOptions::subplot_size_1D = "[ ]"

TODO from plotparameteruncertainty.

Default: "[ ]"

Definition at line 87 of file PestoPlottingOptions.m.

12.2.3.25 PestoPlottingOptions::title = true

Title of PESTO-generated plots.

- true: show
- false: don't show

Default: true

Note

This property has custom functionality when its value is changed.

Definition at line 30 of file PestoPlottingOptions.m.

Referenced by PestoPlottingOptions().

The documentation for this class was generated from the following file:

- @PestoPlottingOptions/PestoPlottingOptions.m

13 File Documentation

13.1 checkSamplingOptions.m File Reference

checkSamplingOptions.m checks the options struct opt regarding sampling inputs. If any option is missing or specified incorrectly, this module will display an error and a suggestion on how to properly specify the option. This module will not set any defaults and thus does not cause options to get silently overwritten.

Functions

- noret::substitute checkSamplingOptions (matlabtypesubstitute par, matlabtypesubstitute opt)

  checkSamplingOptions.m checks the options struct opt regarding sampling inputs. If any option is missing or specified incorrectly, this module will display an error and a suggestion on how to properly specify the option. This module will not set any defaults and thus does not cause options to get silently overwritten.
- mlhsInnerSubst< matlabtypesubstitute, flag > **mtoc_subst_checkSamplingOptions_m_tsbus_cotm_↩ Check** (matlabtypesubstitute str)

13.1.1 Detailed Description

checkSamplingOptions.m checks the options struct opt regarding sampling inputs. If any option is missing or specified incorrectly, this module will display an error and a suggestion on how to properly specify the option. This module will not set any defaults and thus does not cause options to get silently overwritten.

13.1.2 Function Documentation

13.1.2.1 noret::substitute checkSamplingOptions (matlabtypesubstitute *par,* matlabtypesubstitute *opt*)

checkSamplingOptions.m checks the options struct opt regarding sampling inputs. If any option is missing or specified incorrectly, this module will display an error and a suggestion on how to properly specify the option. This module will not set any defaults and thus does not cause options to get silently overwritten.

2017/02/14 Benjamin Ballnus

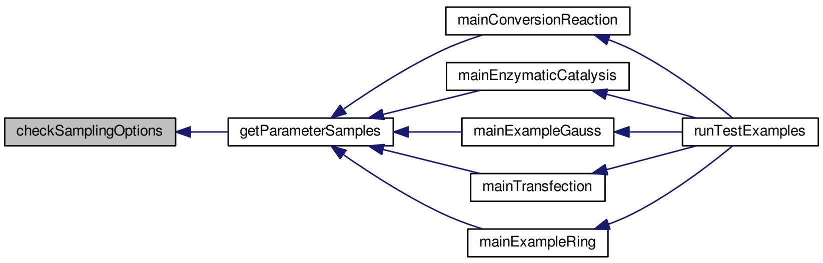
Required fields of par:

Required fields of opt:

Definition at line 17 of file checkSamplingOptions.m.

Referenced by getParameterSamples().

Here is the caller graph for this function:



13.2 collectResults.m File Reference

collectResults() collects and plots the results stored in a common folder

Functions

- mlhsInnerSubst< matlabtypesubstitute, obj > collectResults (matlabtypesubstitute foldername)

  collectResults() collects and plots the results stored in a common folder

13.2.1 Detailed Description

collectResults() collects and plots the results stored in a common folder

13.2.2 Function Documentation

13.2.2.1 mlhsInnerSubst< matlabtypesubstitute, obj > collectResults (matlabtypesubstitute *foldername*)

collectResults() collects and plots the results stored in a common folder

USAGE

[parameters] = collectResults(foldername)

History

2014/06/12 Jan Hasenauer % Initialization

Parameters

foldername	Name of folder from which results are collected.

Return values

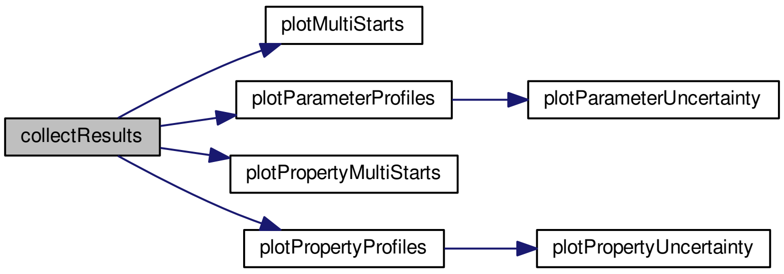
parameters	parameter struct.

Generated fields of obj:

Definition at line 17 of file collectResults.m.

References plotMultiStarts(), plotParameterProfiles(), plotPropertyMultiStarts(), and plotPropertyProfiles().

Here is the call graph for this function:



13.3 examples/conversion_reaction/logLikelihoodCR.m File Reference

Objective function for examples/conversion_reaction.

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, logL >,mlhsInnerSubst< matlabtypesubstitute, dlog←Ldtheta >,mlhsInnerSubst< matlabtypesubstitute, FIM > > logLikelihoodCR (matlabtypesubstitute theta, matlabtypesubstitute t, matlabtypesubstitute Y, matlabtypesubstitute sigma2, matlabtypesubstitute scale)

  Objective function for examples/conversion_reaction.

13.3.1 Detailed Description

Objective function for examples/conversion_reaction.

13.3.2 Function Documentation

13.3.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, logL >,mlhsInnerSubst< matlabtypesubstitute, dlogLdtheta >,mlhsInnerSubst< matlabtypesubstitute, FIM > > logLikelihoodCR (matlabtypesubstitute *theta*, matlabtypesubstitute *t*, matlabtypesubstitute *Y*, matlabtypesubstitute *sigma2*, matlabtypesubstitute *scale*)

Objective function for examples/conversion_reaction.

logLikelihood.m provides the log-likelihood, its gradient and an approximation of the Hessian matrix based on Fisher information matrix (FIM) for the conversion reaction process.

Parameters

theta	Model parameters [theta_1, theta_2]'
t	vector of time points
Y	measurement vector
sigma2	variance of the measurements (noise)
scale	`lin` or `log`

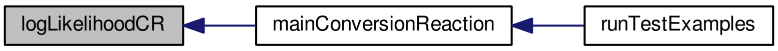
Return values

logL	double, value of log-likelihood
dlogLdtheta	double vector, gradient of log-likelihood
FIM	double array, negative Fisher information matrix, an approximation of Hessian of log-likelihood

Definition at line 17 of file logLikelihoodCR.m.

Referenced by mainConversionReaction().

Here is the caller graph for this function:



13.4 examples/conversion_reaction/mainConversionReaction.m File Reference

Main file of the conversion reaction example.

Functions

- noret::substitute mainConversionReaction ()
  Main file of the conversion reaction example.

13.4.1 Detailed Description

Main file of the conversion reaction example.

13.4.2 Function Documentation

13.4.2.1 noret::substitute mainConversionReaction ( )

Main file of the conversion reaction example.

Demonstrates the use of

- getMultiStarts()
- getParameterProfiles()
- getParameterSamples()
- getParameterConfidenceIntervals()
- getPropertyMultiStarts()
- getPropertyProfiles()

- getPropertySamples()
- getPropertyConfidenceIntervals()

This example provides a model for the interconversion of two species (X_1 and X_2) following first-order mass action kinetics with the

parameters theta_1 and theta_2 respectively

- X_1 -> X_2, rate = theta_1 ∗ [X_1]
- X_2 -> X_1, rate = theta_2 ∗ [X_2]

Measurement of [X_2] are provided as: Y = [X_2]

This file provides time-series measurement data Y and performs a multistart maximum likelihood parameter estimation based on these measurements, demonstrating the use of getMultiStarts(). The model fit is then visualized.

Profile likelihood calculation is done using getParameterProfiles().

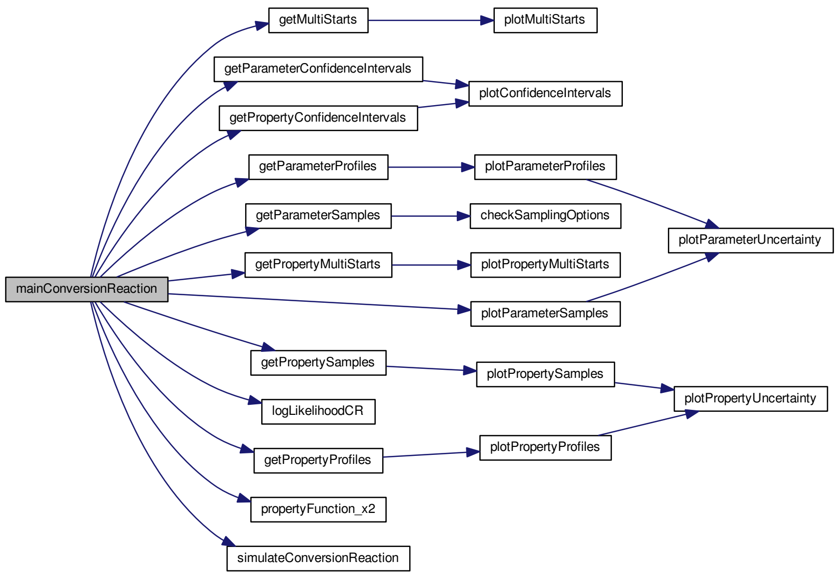
Single-chain Monte-Carlo sampling is performed by getParameterSamples() and plotted.

Definition at line 17 of file mainConversionReaction.m.

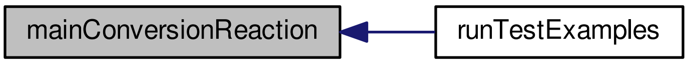
References getMultiStarts(), getParameterConfidenceIntervals(), getParameterProfiles(), getParameterSamples(), getPropertyConfidenceIntervals(), getPropertyMultiStarts(), getPropertyProfiles(), getPropertySamples(), log↩LikelihoodCR(), plotParameterSamples(), propertyFunction_x2(), and simulateConversionReaction().

Referenced by runTestExamples().

Here is the call graph for this function:



Here is the caller graph for this function:



13.5 examples/conversion_reaction/propertyFunction_theta1.m File Reference

propertyFunction_theta1 for examples/conversion_reaction

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, f >,mlhsInnerSubst< matlabtypesubstitute, grad↩_f >,mlhsInnerSubst< matlabtypesubstitute, hess_f > > propertyFunction_theta1 (matlabtypesubstitute theta)

  propertyFunction_theta1 for examples/conversion_reaction

13.5.1 Detailed Description

propertyFunction_theta1 for examples/conversion_reaction

13.5.2 Function Documentation

13.5.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, f >,mlhsInnerSubst< matlabtypesubstitute, grad_f >,mlhsInnerSubst< matlabtypesubstitute, hess_f > > propertyFunction_theta1 (matlabtypesubstitute *theta*)

propertyFunction_theta1 for examples/conversion_reaction

returns the the first reaction rate as defined in logLikelihood.m with derivatives

Parameters

theta	Model parameters [theta_1, theta_2]'

Return values

f	double, value of property function
grad↩_f	double vector, gradient of property function
hess↩_f	double array, hessian of property function

Definition at line 17 of file propertyFunction_theta1.m.

13.6 examples/conversion_reaction/propertyFunction_theta1_minus_theta2.m File Reference

propertyFunction_theta1_minus_theta2 for examples/conversion_reaction

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, f >,mlhsInnerSubst< matlabtypesubstitute, grad_↩ f >,mlhsInnerSubst< matlabtypesubstitute, hess_f > > propertyFunction_theta1_minus_theta2 (matlab-typesubstitute theta)

  propertyFunction_theta1_minus_theta2 for examples/conversion_reaction

13.6.1 Detailed Description

propertyFunction_theta1_minus_theta2 for examples/conversion_reaction

13.6.2 Function Documentation

13.6.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, f >,mlhsInnerSubst< matlabtypesubstitute, grad_f >,mlhsInnerSubst< matlabtypesubstitute, hess_f > > propertyFunction_theta1_minus_theta2 (matlabtypesubstitute *theta*)

propertyFunction_theta1_minus_theta2 for examples/conversion_reaction

returns the difference of the reaction rates as defined in logLikelihood.m with derivatives

Parameters

theta	Model parameters [theta_1, theta_2]'

Return values

f	double, value of property function
grad↩_f	double vector, gradient of property function
hess↩_f	double array, hessian of property function

Definition at line 17 of file propertyFunction_theta1_minus_theta2.m.

13.7 examples/conversion_reaction/propertyFunction_theta1_square.m File Reference

propertyFunction_theta1_square for examples/conversion_reaction

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, f >,mlhsInnerSubst< matlabtypesubstitute, grad_↩ f >,mlhsInnerSubst< matlabtypesubstitute, hess_f > > propertyFunction_theta1_square (matlabtypesubstitute theta)

  propertyFunction_theta1_square for examples/conversion_reaction

13.7.1 Detailed Description

propertyFunction_theta1_square for examples/conversion_reaction

13.7.2 Function Documentation

13.7.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, f >,mlhsInnerSubst< matlabtypesubstitute, grad_f >,mlhsInnerSubst< matlabtypesubstitute, hess_f > > propertyFunction_theta1_square (matlabtypesubstitute *theta*)

propertyFunction_theta1_square for examples/conversion_reaction

returns the square of the first reaction rate as defined in logLikelihood.m with derivatives

Parameters

theta	Model parameters [theta_1, theta_2]'

Return values

f	double, value of property function
grad↩ _f	double vector, gradient of property function
hess↩ _f	double array, hessian of property function

Definition at line 17 of file propertyFunction_theta1_square.m.

13.8 examples/conversion_reaction/propertyFunction_theta2.m File Reference

propertyFunction_theta2 for examples/conversion_reaction

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, f >,mlhsInnerSubst< matlabtypesubstitute, grad↩ _f >,mlhsInnerSubst< matlabtypesubstitute, hess_f > > propertyFunction_theta2 (matlabtypesubstitute theta)

  propertyFunction_theta2 for examples/conversion_reaction

13.8.1 Detailed Description

propertyFunction_theta2 for examples/conversion_reaction

13.8.2 Function Documentation

13.8.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, f >,mlhsInnerSubst< matlabtypesubstitute, grad_f >,mlhsInnerSubst< matlabtypesubstitute, hess_f > > propertyFunction_theta2 (matlabtypesubstitute *theta*)

propertyFunction_theta2 for examples/conversion_reaction

returns the the second reaction rate as defined in logLikelihood.m with derivatives

Parameters

theta	Model parameters [theta_1, theta_2]'

Return values

f	double, value of property function
grad←_f	double vector, gradient of property function
hess←_f	double array, hessian of property function

Definition at line 17 of file propertyFunction_theta2.m.

13.9 examples/conversion_reaction/propertyFunction_x2.m File Reference

propertyFunction_x2 for examples/conversion_reaction

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, f >,mlhsInnerSubst< matlabtypesubstitute, grad_f > > propertyFunction_x2 (matlabtypesubstitute theta, matlabtypesubstitute T, matlabtypesubstitute scale)
  propertyFunction_x2 for examples/conversion_reaction

13.9.1 Detailed Description

propertyFunction_x2 for examples/conversion_reaction

13.9.2 Function Documentation

13.9.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, f >,mlhsInnerSubst< matlabtypesubstitute, grad_f > > propertyFunction_x2 (matlabtypesubstitute *theta,* matlabtypesubstitute *T,* matlabtypesubstitute *scale*)

propertyFunction_x2 for examples/conversion_reaction

returns the value of x_2 at time T as defined in logLikelihood.m with derivatives

Parameters

theta	Model parameters [theta_1, theta_2]'
T	stopping time for simulation
scale	`lin` or `log`

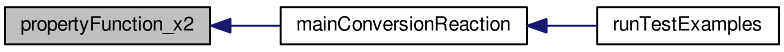
Return values

f	double, value of property function
grad↩ _f	double vector, gradient of property function

Definition at line 17 of file propertyFunction_x2.m.

Referenced by mainConversionReaction().

Here is the caller graph for this function:



13.10 examples/conversion_reaction/simulateConversionReaction.m File Reference

simulateConversionReaction for examples/conversion_reaction

Functions

- mlhsInnerSubst$<$ matlabtypesubstitute, y $>$ simulateConversionReaction (matlabtypesubstitute theta, matlabtypesubstitute t)

  simulateConversionReaction for examples/conversion_reaction

13.10.1 Detailed Description

simulateConversionReaction for examples/conversion_reaction

13.10.2 Function Documentation

13.10.2.1 mlhsInnerSubst$<$ matlabtypesubstitute, y $>$ simulateConversionReaction (matlabtypesubstitute *theta*, matlabtypesubstitute *t*)

simulateConversionReaction for examples/conversion_reaction

simulateConversionReaction performs a simulation of the conversion reaction model for the given timepoints t and parameters theta

Parameters

theta	Model parameters [theta_1, theta_2]'
t	vector of time points

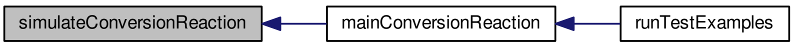
Return values

y	double vector, values of the observable Y = [x_2] at time points t

Definition at line 17 of file simulateConversionReaction.m.

Referenced by mainConversionReaction().

Here is the caller graph for this function:



13.11 examples/enzymatic_catalysis/logLikelihoodEC.m File Reference

Objective function for examples/enzymatic_catalysis.

Functions

- mlhsInnerSubst$<$ matlabtypesubstitute, varargout $>$ logLikelihoodEC (matlabtypesubstitute theta, matlabtypesubstitute yMeasured, matlabtypesubstitute sigma2, matlabtypesubstitute con0, matlabtypesubstitute nTimepoints, matlabtypesubstitute nMeasure)

  Objective function for examples/enzymatic_catalysis.

13.11.1 Detailed Description

Objective function for examples/enzymatic_catalysis.

13.11.2 Function Documentation

13.11.2.1 mlhsInnerSubst$<$ matlabtypesubstitute, varargout $>$ logLikelihoodEC (matlabtypesubstitute *theta,* matlabtypesubstitute *yMeasured,* matlabtypesubstitute *sigma2,* matlabtypesubstitute *con0,* matlabtypesubstitute *nTimepoints,* matlabtypesubstitute *nMeasure*)

Objective function for examples/enzymatic_catalysis.

logLikelihood.m provides the log-likelihood, its gradient and an approximation of the Hessian matrix based on Fisher information matrix (FIM) for the enzymatic catalysis example.

Parameters

theta	Model parameters [theta_1, theta_2, theta_3, theta_4]'
yMeasured	measurement array returned from getMeasuredData()
sigma2	variance of the measurements (noise)
con0	inititial concentrations of the experiments returnd from getInitialConcentrations()
nTimepoints	number of Time points (equidistad between 0 and 5)
nMeasure	number of experiments

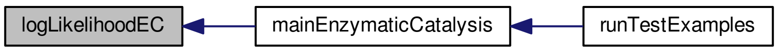
Return values

J	double, value of log-likelihood
gradJ	double vector, gradient of log-likelihood
FIM	double array, negative Fisher information matrix, an approximation of Hessian of log-likelihood

Definition at line 17 of file logLikelihoodEC.m.

Referenced by mainEnzymaticCatalysis().

Here is the caller graph for this function:



13.12 examples/enzymatic_catalysis/mainEnzymaticCatalysis.m File Reference

Main file of the enzymatic catalysis example.

Functions

- noret::substitute mainEnzymaticCatalysis ()

  Main file of the enzymatic catalysis example.

13.12.1 Detailed Description

Main file of the enzymatic catalysis example.

13.12.2 Function Documentation

13.12.2.1 noret::substitute mainEnzymaticCatalysis ( )

Main file of the enzymatic catalysis example.

Demonstrates the use of

- getParameterSamples()
- getMultiStarts()
- getParameterConfidenceIntervals()
- getParameterProfiles()

Demonstrates furthermore

- how to do sampling without multi-start local optimization beforehand
- the value of multi-start local optimization before sampling
- how to use the MEIGO toolbox for optimization
- how to compute profile likelihoods via ODE integration

This example provides a model for the reaction of a species X_1 to a species X_4, which is catalyzed by an enzyme X_2.

- X_1 + X_2 -> X_3, rate = theta_1 $*$ [X_1] $*$ [X_2]
- X_3 -> X_1 + X_2, rate = theta_2 $*$ [X_3]
- X_3 -> X_4 + X_2, rate = theta_3 $*$ [X_3]
- X_4 + X_2 -> X_3, rate = theta_4 $*$ [X_4] $*$ [X_2]

Measurements of [X_1] and [X_4] are provided as: Y = [[X_1]; [X_4]]

This file set a parameter vector, creates and saves artificial measurement data as a time series and performs a multi-start local optimization based on these measurements, demonstrating the use of getMultiStarts().

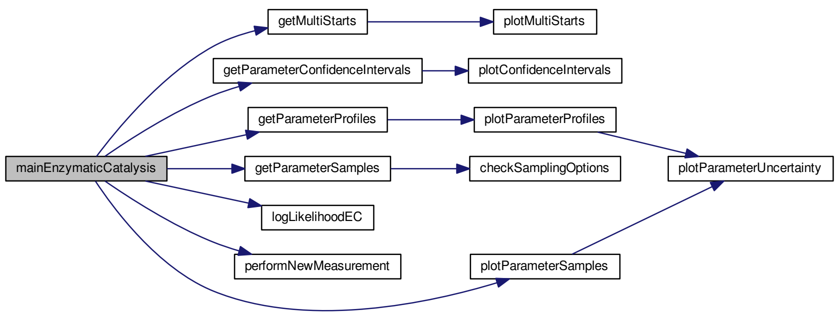
The Profile likelihoods are calculated by integrating an ODE following the profile path using getParameterProfiles with the option optionsPesto.profile_method = `integration`.

Definition at line 17 of file mainEnzymaticCatalysis.m.

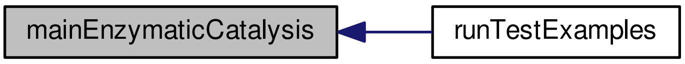
References getMultiStarts(), getParameterConfidenceIntervals(), getParameterProfiles(), getParameterSamples(), logLikelihoodEC(), performNewMeasurement(), and plotParameterSamples().

Referenced by runTestExamples().

Here is the call graph for this function:



Here is the caller graph for this function:



13.13 examples/enzymatic_catalysis/performNewMeasurement.m File Reference

performNewMeasurement.m for examples/enzymatic_catalysis

Functions

- noret::substitute performNewMeasurement (matlabtypesubstitute theta, matlabtypesubstitute nMeasure, matlabtypesubstitute nTimepoints, matlabtypesubstitute sigma2)

  performNewMeasurement.m for examples/enzymatic_catalysis

13.13.1 Detailed Description

performNewMeasurement.m for examples/enzymatic_catalysis

13.13.2 Function Documentation

13.13.2.1 noret::substitute performNewMeasurement (matlabtypesubstitute *theta*, matlabtypesubstitute *nMeasure*, matlabtypesubstitute *nTimepoints*, matlabtypesubstitute *sigma2*)

performNewMeasurement.m for examples/enzymatic_catalysis

creates artificial data for the parameter estimation of the enzymatic catalysis example for a given number of time points and a given number of measurements with different initial conditions of the chemical species.

Parameters

theta	Model parameters [theta_1, theta_2, theta_3, theta_4]'
nMeasure	number of experiments
nTimepoints	number of Time points (equidistad between 0 and 5)
sigma2	variance of the measurements (noise)

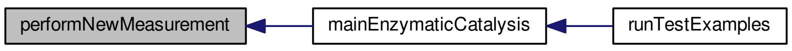
Return values

sigma2	No return values, but the files getInitialConcentrations.m and getMeasuredData.m are written to the folder of this example

Definition at line 17 of file performNewMeasurement.m.

Referenced by mainEnzymaticCatalysis().

Here is the caller graph for this function:



13.14 examples/erbb_signaling/erbb_signaling_pesto_syms.m File Reference

erbb_signaling_pesto_syms for examples/erbb_signaling

Functions

- mlhsInnerSubst< matlabtypesubstitute, model > erbb_signaling_pesto_syms ()

  erbb_signaling_pesto_syms for examples/erbb_signaling
- mlhsInnerSubst< matlabtypesubstitute, r > **mtoc_subst_erbb_signaling_pesto_syms_m_tsbus_cotm↩_pow** (matlabtypesubstitute x, matlabtypesubstitute y)
- mlhsInnerSubst< matlabtypesubstitute, r > **mtoc_subst_erbb_signaling_pesto_syms_m_tsbus_cotm↩_power** (matlabtypesubstitute x, matlabtypesubstitute y)
- mlhsInnerSubst< matlabtypesubstitute, r > **mtoc_subst_erbb_signaling_pesto_syms_m_tsbus_cotm↩_stepfunc2** (matlabtypesubstitute t, matlabtypesubstitute t_start, matlabtypesubstitute v_start, matlabtypesubstitute t_end, matlabtypesubstitute v_end)
- mlhsInnerSubst< matlabtypesubstitute, r > **mtoc_subst_erbb_signaling_pesto_syms_m_tsbus_cotm↩_stepfunc** (matlabtypesubstitute t, matlabtypesubstitute t_start, matlabtypesubstitute v_start, matlabtypesubstitute t_end, matlabtypesubstitute v_end)

13.14.1 Detailed Description

erbb_signaling_pesto_syms for examples/erbb_signaling

13.14.2 Function Documentation

13.14.2.1 mlhsInnerSubst< matlabtypesubstitute, model > erbb_signaling_pesto_syms ( )

erbb_signaling_pesto_syms for examples/erbb_signaling

creates an amimodel-object for the AMICI solver

Return values

model	amimodel object

Generated fields of model:

Definition at line 17 of file erbb_signaling_pesto_syms.m.

13.15 examples/erbb_signaling/getData_ErbB_signaling.m File Reference

getData_erbb_signaling.m for examples/Chen2009

Functions

- mlhsInnerSubst< matlabtypesubstitute, DD > getData_ErbB_signaling ()
  getData_erbb_signaling.m for examples/Chen2009

13.15.1 Detailed Description

getData_erbb_signaling.m for examples/Chen2009

13.15.2 Function Documentation

13.15.2.1 mlhsInnerSubst< matlabtypesubstitute, DD > getData_ErbB_signaling ( )

getData_erbb_signaling.m for examples/Chen2009

getData_erbb_signaling.m provides measurement data for the parameter estimation of the erbb_signaling example file.

Parameters

void	

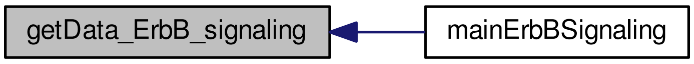
Return values

D	struct, containing the time-points of measurement, the measured Data, the variance and the initial conditions

Definition at line 17 of file getData_ErbB_signaling.m.

Referenced by mainErbBSignaling().

Here is the caller graph for this function:



13.16 examples/erbb_signaling/logLikelihoodErbBSignaling.m File Reference

Objective function for examples/erbb_signaling.

Functions

- mlhsInnerSubst< matlabtypesubstitute, varargout > logLikelihoodErbBSignaling (matlabtypesubstitute theta, matlabtypesubstitute D)

  Objective function for examples/erbb_signaling.

13.16.1 Detailed Description

Objective function for examples/erbb_signaling.

13.16.2 Function Documentation

13.16.2.1 mlhsInnerSubst< matlabtypesubstitute, varargout > logLikelihoodErbBSignaling (matlabtypesubstitute *theta*, matlabtypesubstitute *D*)

Objective function for examples/erbb_signaling.

logLikelihoodErbBSignaling.m provides the log-likelihood and its gradient for the model defined in erbb_signaling↩
_pesto_syms.m

USAGE

[llh] = logLikelihoodErbBSignaling(theta, amiData) [llh, sllh] = logLikelihoodErbBSignaling(theta, amiData)

Parameters

theta	Model parameters
amiData	an amidata object for the AMICI solver

Return values

varargout	

Return values

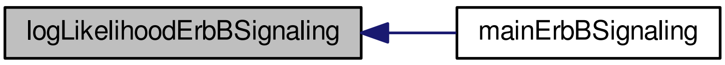
llh	Log-Likelihood, only the LogLikelihood will be returned, no sensitivity analysis is performed
sllh	Gradient of llh, The LogLikelihood and its gradient will be returned, first order adjoint sensitivity analysis is performed

Required fields of D:

Definition at line 17 of file logLikelihoodErbBSignaling.m.

Referenced by mainErbBSignaling().

Here is the caller graph for this function:



13.17 examples/erbb_signaling/mainErbBSignaling.m File Reference

Main file of the erbb_signaling example.

Functions

- noret::substitute mainErbBSignaling ()

  Main file of the erbb_signaling example.

13.17.1 Detailed Description

Main file of the erbb_signaling example.

13.17.2 Function Documentation

13.17.2.1 noret::substitute mainErbBSignaling ( )

Main file of the erbb_signaling example.

Demonstrates the use of

- getMultiStarts()

This example is a rather big biological model. It is included here to show the ability of PESTO to handle O↩ DE-based models with some hundred state variables and some hundred parameters. This model is taken from the paper "Input-output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data", by Chen et al., in 2009, PubMed, vol.5, no.239 (see `https://www.ncbi.nlm.nih.↩ gov/pubmed/19156131`).

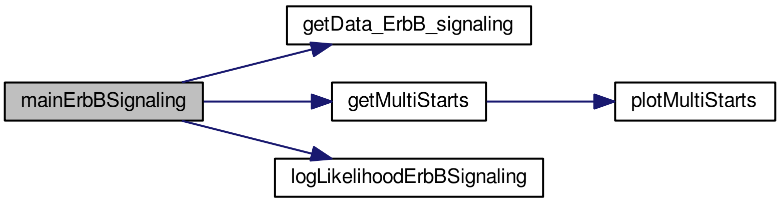
The data used is measurement data provided in the publication.

This file performs a multistart local optimization based on measured data from the referenced papers, demonstrating the use of getMultiStarts().

Definition at line 17 of file mainErbBSignaling.m.

References getData_ErbB_signaling(), getMultiStarts(), and logLikelihoodErbBSignaling().

Here is the call graph for this function:



13.18 examples/GaussExample/define_Gauss_LLH.m File Reference

Hard version rng(15); angle = pi/2+2∗pi/8; dimi = 18; scale = blkdiag(1e0∗[250,0;0,1],diag(ones(1,dimi)));.

Functions

- noret::substitute define_Gauss_LLH ()

  Hard version rng(15); angle = pi/2+2∗pi/8; dimi = 18; scale = blkdiag(1e0∗[250,0;0,1],diag(ones(1,dimi)));.

13.18.1 Detailed Description

Hard version rng(15); angle = pi/2+2∗pi/8; dimi = 18; scale = blkdiag(1e0∗[250,0;0,1],diag(ones(1,dimi)));.

13.19 examples/GaussExample/mainExampleGauss.m File Reference

mainExampleGauss.m shows how to use sampling methods in PESTO. The example problem is a mixture of Gaussian modes in the first two dimensions and a single mode in any other dimension. The dimensionality, angle, scaling and position of modes can be altered in define_Gauss_LLH.m.

Functions

- noret::substitute mainExampleGauss ()

  mainExampleGauss.m shows how to use sampling methods in PESTO. The example problem is a mixture of Gaussian modes in the first two dimensions and a single mode in any other dimension. The dimensionality, angle, scaling and position of modes can be altered in define_Gauss_LLH.m.

13.19.1 Detailed Description

mainExampleGauss.m shows how to use sampling methods in PESTO. The example problem is a mixture of Gaussian modes in the first two dimensions and a single mode in any other dimension. The dimensionality, angle, scaling and position of modes can be altered in define_Gauss_LLH.m.

13.19.2 Function Documentation

13.19.2.1 noret::substitute mainExampleGauss ( )

mainExampleGauss.m shows how to use sampling methods in PESTO. The example problem is a mixture of Gaussian modes in the first two dimensions and a single mode in any other dimension. The dimensionality, angle, scaling and position of modes can be altered in define_Gauss_LLH.m.

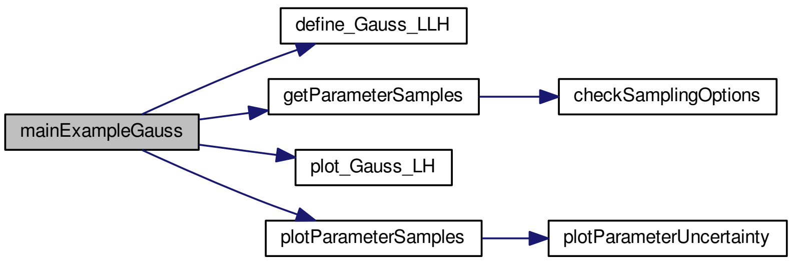
Written by Benjamin Ballnus 2/2017

Definition at line 17 of file mainExampleGauss.m.

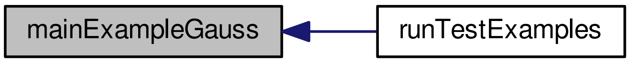
References define_Gauss_LLH(), getParameterSamples(), plot_Gauss_LH(), and plotParameterSamples().

Referenced by runTestExamples().

Here is the call graph for this function:



Here is the caller graph for this function:



13.20 examples/GaussExample/plot_Gauss_LH.m File Reference

rng(5); mu = rand(2,20)∗10; % sigma = 0.05∗ones(1,20) + (1-0.05) ∗ rand(1,20); sigma = 2.5∗ones(1,20); rng(`shuffle`);

Functions

- noret::substitute plot_Gauss_LH ()

  rng(5); mu = rand(2,20)∗10; % sigma = 0.05∗ones(1,20) + (1-0.05) ∗ rand(1,20); sigma = 2.5∗ones(1,20); rng(`shuffle`);

13.20.1 Detailed Description

rng(5); mu = rand(2,20)∗10; % sigma = 0.05∗ones(1,20) + (1-0.05) ∗ rand(1,20); sigma = 2.5∗ones(1,20); rng(`shuffle`);

13.21 examples/GaussExample/plot_gaussian_ellipsoid.m File Reference

PLOT_GAUSSIAN_ELLIPSOIDS plots 2-d and 3-d Gaussian distributions.

Functions

- mlhsInnerSubst< matlabtypesubstitute, h > plot_gaussian_ellipsoid (matlabtypesubstitute m, matlabtypesubstitute C, matlabtypesubstitute sdwidth, matlabtypesubstitute npts, matlabtypesubstitute axh)

  PLOT_GAUSSIAN_ELLIPSOIDS plots 2-d and 3-d Gaussian distributions.
- mlhsInnerSubst< matlabtypesubstitute, h > **mtoc_subst_plot_gaussian_ellipsoid_m_tsbus_cotm_↩ show2d** (matlabtypesubstitute means, matlabtypesubstitute C, matlabtypesubstitute sdwidth, matlabtypesubstitute npts, matlabtypesubstitute axh)
- mlhsInnerSubst< matlabtypesubstitute, h > **mtoc_subst_plot_gaussian_ellipsoid_m_tsbus_cotm_↩ show3d** (matlabtypesubstitute means, matlabtypesubstitute C, matlabtypesubstitute sdwidth, matlabtypesubstitute npts, matlabtypesubstitute axh)

13.21.1 Detailed Description

PLOT_GAUSSIAN_ELLIPSOIDS plots 2-d and 3-d Gaussian distributions.

13.21.2 Function Documentation

13.21.2.1 mlhsInnerSubst< matlabtypesubstitute, h > plot_gaussian_ellipsoid (matlabtypesubstitute *m,* matlabtypesubstitute *C,* matlabtypesubstitute *sdwidth,* matlabtypesubstitute *npts,* matlabtypesubstitute *axh*)

PLOT_GAUSSIAN_ELLIPSOIDS plots 2-d and 3-d Gaussian distributions.

H = PLOT_GAUSSIAN_ELLIPSOIDS(M, C) plots the distribution specified by mean M and covariance C. The distribution is plotted as an ellipse (in 2-d) or an ellipsoid (in 3-d). By default, the distributions are plotted in the current axes. H is the graphics handle to the plotted ellipse or ellipsoid.

PLOT_GAUSSIAN_ELLIPSOIDS(M, C, SD) uses SD as the standard deviation along the major and minor axes (larger SD => larger ellipse). By default, SD = 1. Note:

of the total probability mass, respectively.

of the total probability mass, respectively.

PLOT_GAUSSIAN_ELLIPSOIDS(M, C, SD, NPTS) plots the ellipse or ellipsoid with a resolution of NPTS (ellipsoids are generated on an NPTS x NPTS mesh; see SPHERE for more details). By default, NPTS = 50 for ellipses, and 20 for ellipsoids.

PLOT_GAUSSIAN_ELLIPSOIDS(M, C, SD, NPTS, AX) adds the plot to the axes specified by the axis handle AX.

Examples:

% Plot three 2-d Gaussians figure; h1 = plot_gaussian_ellipsoid([1 1], [1 0.5; 0.5 1]); h2 = plot_gaussian_ellipsoid([2 1.5], [1 -0.7; -0.7 1]); h3 = plot_gaussian_ellipsoid([0 0], [1 0; 0 1]); set(h2,`color`,`r`); set(h3,`color`,`g`);

% "Contour map" of a 2-d Gaussian figure; for sd = [0.3:0.4:4], h = plot_gaussian_ellipsoid([0 0], [1 0.8; 0.8 1], sd); end

% Plot three 3-d Gaussians figure; h1 = plot_gaussian_ellipsoid([1 1 0], [1 0.5 0.2; 0.5 1 0.4; 0.2 0.4 1]); h2 = plot_gaussian_ellipsoid([1.5 1 .5], [1 -0.7 0.6; -0.7 1 0; 0.6 0 1]); h3 = plot_gaussian_ellipsoid([1 2 2], [0.5 0 0; 0 0.5 0; 0 0 0.5]); set(h2,`facealpha`,0.6); view(129,36); set(gca,`proj`,`perspective`); grid on;

grid on; axis equal; axis tight;

Gautam Vallabha, Sep-23-2007, `Gautam.Vallabha@mathworks.com`

Definition at line 17 of file plot_gaussian_ellipsoid.m.

13.22 examples/jakstat_signaling/jakstat_pesto_syms.m File Reference

jakstat_pesto_syms for examples/jakstat_signaling

Functions

- mlhsInnerSubst< matlabtypesubstitute, model > jakstat_pesto_syms ()
  jakstat_pesto_syms for examples/jakstat_signaling

13.22.1 Detailed Description

jakstat_pesto_syms for examples/jakstat_signaling

13.22.2 Function Documentation

13.22.2.1 mlhsInnerSubst< matlabtypesubstitute, model > jakstat_pesto_syms ( )

jakstat_pesto_syms for examples/jakstat_signaling

creates an amimodel-object for the AMICI solver

Return values

model	amimodel object

Generated fields of model:

Definition at line 17 of file jakstat_pesto_syms.m.

13.23 examples/jakstat_signaling/logLikelihoodJakstat.m File Reference

Objective function for examples/jakstat_signaling.

Functions

- mlhsInnerSubst$<$ matlabtypesubstitute, varargout $>$ logLikelihoodJakstat (matlabtypesubstitute theta, matlabtypesubstitute amiData)

    Objective function for examples/jakstat_signaling.

13.23.1 Detailed Description

Objective function for examples/jakstat_signaling.

13.23.2 Function Documentation

13.23.2.1 mlhsInnerSubst$<$ matlabtypesubstitute, varargout $>$ logLikelihoodJakstat (matlabtypesubstitute *theta,* matlabtypesubstitute *amiData*)

Objective function for examples/jakstat_signaling.

logLikelihoodJakstat.m provides the log-likelihood, its gradient and an the Hessian matrix for the model for the JakStat signaling pathway as defined in jakstat_pesto_syms.m

USAGE

[llh] = getParameterProfiles(theta, amiData) [llh, sllh] = getParameterProfiles(theta, amiData) [llh, sllh, s2llh] = getParameterProfiles(theta, amiData)

Parameters

theta	Model parameters
amiData	an amidata object for the AMICI solver

Return values

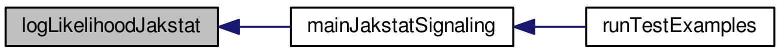
varargout	
llh	Log-Likelihood, only the LogLikelihood will be returned, no sensitivity analysis is performed
sllh	Gradient of llh, The LogLikelihood and its gradient will be returned, first order adjoint sensitivity analysis is performed
s2llh	Hessian of llh, The LogLikelihood, its gradient and the Hessian matrix will be returned, second order adjoint sensitivity analysis is performed

Required fields of amiData:

Definition at line 17 of file logLikelihoodJakstat.m.

Referenced by mainJakstatSignaling().

Here is the caller graph for this function:



13.24 examples/jakstat_signaling/mainJakstatSignaling.m File Reference

Main file of the JakStat signaling example.

Functions

- noret::substitute mainJakstatSignaling ()

    Main file of the JakStat signaling example.

13.24.1 Detailed Description

Main file of the JakStat signaling example.

13.24.2 Function Documentation

13.24.2.1 noret::substitute mainJakstatSignaling ( )

Main file of the JakStat signaling example.

Demonstrates the use of

- getMultiStarts()

Demostrates furhtermore

- how to implement a user-supplied guess for intial parameters

This example provides a model for the JakStat signaling pathway with an time resolved input of the drug E↩PO. The model has been taken from the papers "Identification of nucleocytoplasmic cycling as a remote sensor in cellular signaling by databased modeling" by Swameye et al. in 2003, PNAS, vol.100, no.3 (see `http↩://www.pnas.org/content/100/3/1028.long`) and "Comprehensive estimation of input signals and dynamics in biochemical reaction networks" by Schelker et al. in 2012, Bioinformatics, vol.28 (see `http↩://bioinformatics.oxfordjournals.org/content/28/18/i529.full`).

The data used is measurement data provided in the publications.

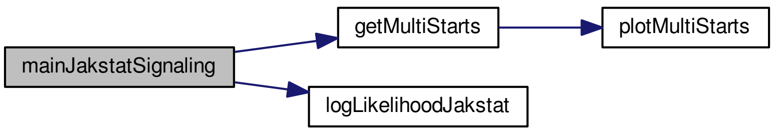
This file performs a multistart local optimization based on measured data from the referenced papers, demonstrating the use of getMultiStarts().

Definition at line 17 of file mainJakstatSignaling.m.

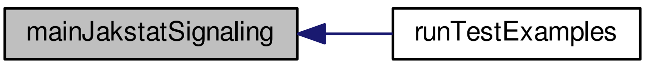
References getMultiStarts(), and logLikelihoodJakstat().

Referenced by runTestExamples().

Here is the call graph for this function:



Here is the caller graph for this function:



13.25 examples/mRNA_transfection/logLikelihoodT.m File Reference

Objective function for examples/mRNA_transfection.

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, logL >,mlhsInnerSubst< matlabtypesubstitute, dlog↩ Ldtheta >,mlhsInnerSubst< matlabtypesubstitute, FIM > > logLikelihoodT (matlabtypesubstitute varargin)

  Objective function for examples/mRNA_transfection.

13.25.1 Detailed Description

Objective function for examples/mRNA_transfection.

13.25.2 Function Documentation

13.25.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, logL >,mlhsInnerSubst< matlabtypesubstitute, dlogLdtheta >,mlhsInnerSubst< matlabtypesubstitute, FIM > > logLikelihoodT (matlabtypesubstitute *varargin*)

Objective function for examples/mRNA_transfection.

logLikelihoodT.m provides the log-likelihood, its gradient and an approximation of the Hessian matrix based on Fisher information matrix (FIM) for the conversion reaction process.

Parameters

varargin	`logLikelihoodT ( theta, t, D )` *Required Parameters for varargin:* • theta Model parameters • t vector of time points • D measurement data

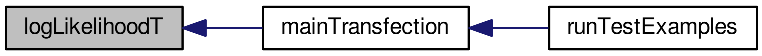
Return values

logL	double, value of log-likelihood
dlogLdtheta	double vector, gradient of the log-likelihood
FIM	double array, approximation of the Hessian matrix based on the Fisher information matrix

Definition at line 17 of file logLikelihoodT.m.

Referenced by mainTransfection().

Here is the caller graph for this function:



13.26 examples/mRNA_transfection/mainTransfection.m File Reference

Main file of the mRNA transfection example.

Functions

- noret::substitute mainTransfection ()

    Main file of the mRNA transfection example.

13.26.1 Detailed Description

Main file of the mRNA transfection example.

13.26.2 Function Documentation

13.26.2.1 noret::substitute mainTransfection ( )

Main file of the mRNA transfection example.

Demonstrates the use of

- getMultiStarts()
- getParameterProfiles()
- getParameterSamples()
- getParameterConfidenceIntervals()
- getPropertyMultiStarts()
- getPropertyProfiles()
- getPropertySamples()
- getPropertyConfidenceIntervals()

Demonstrates furthermore

- how to carry out uncertainty analysis for local (non-global) optimum
- how to use the PSwarm toolbox for optimization (commented) and what problems may occur when no gradient based approach is used

This example is a model for mRNA transfection, taken from the paper "Single-cell mRNA transfection studies↩ : Delivery, kinetics and statistics by numbers", by Leonhardt C et al., Nanomedicine: NBM, 2014, vol.10 (see http://dx.doi.org/10.1016/j.nano.2013.11.008)

The data used is measurement data provided in the publication.

This file provides time-series measurement data Y and performs a multistart maximum likelihood parameter estimation based on these measurements, demonstrating the use of getMultiStarts(). The model fit is then visualized.

Profile likelihood calculation is done by optimization and integration using getParameterProfiles() with the option optionsMultistart.profile_method = `mixed`to have comparison of both methods.

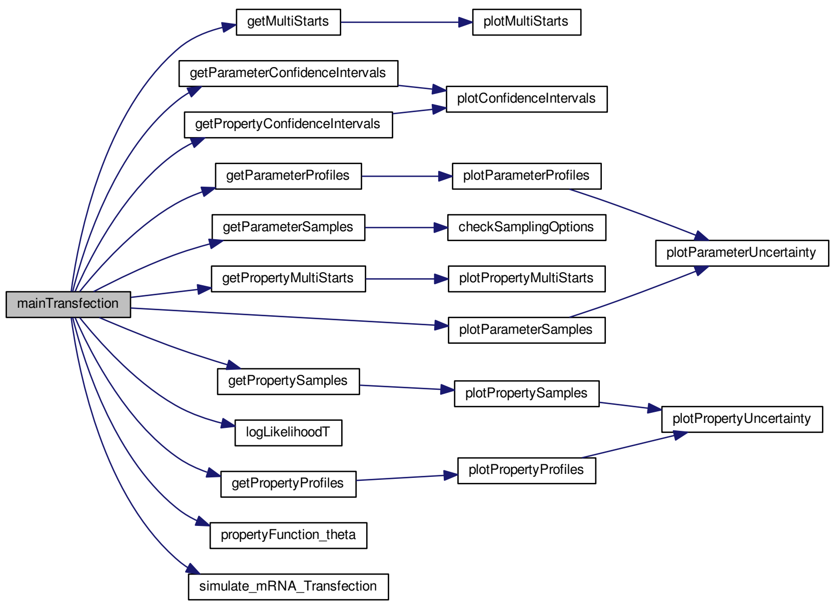
Multi-chain Monte-Carlo sampling is performed by getParameterSamples() and plotted using plotParameter↩ Uncertainty().

Definition at line 17 of file mainTransfection.m.

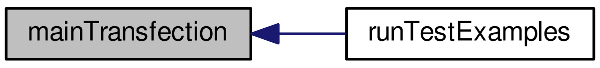
References getMultiStarts(), getParameterConfidenceIntervals(), getParameterProfiles(), getParameterSamples(), getPropertyConfidenceIntervals(), getPropertyMultiStarts(), getPropertyProfiles(), getPropertySamples(), log↩ LikelihoodT(), plotParameterSamples(), propertyFunction_theta(), and simulate_mRNA_Transfection().

Referenced by runTestExamples().

Here is the call graph for this function:



Here is the caller graph for this function:



13.27 examples/mRNA_transfection/propertyFunction_theta.m File Reference

propertyFunction_theta for examples/mRNA_transfection

Functions

- mlhsInnerSubst< matlabtypesubstitute, varargout > propertyFunction_theta (matlabtypesubstitute theta, matlabtypesubstitute i)

  propertyFunction_theta for examples/mRNA_transfection

13.27.1 Detailed Description

propertyFunction_theta for examples/mRNA_transfection

13.27.2 Function Documentation

13.27.2.1 mlhsInnerSubst< matlabtypesubstitute, varargout > propertyFunction_theta (matlabtypesubstitute *theta*, matlabtypesubstitute *i*)

propertyFunction_theta for examples/mRNA_transfection

returns the value of the i-th parameter as defined in logLikelihood.m with derivatives

Parameters

theta	parameter vector
i	index for the parameter

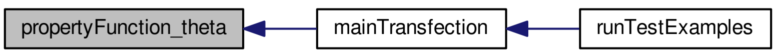
Return values

f	double, value of property function
grad_f	double vector, gradient of property function
hess_f	double array, hessian of property function

Definition at line 17 of file propertyFunction_theta.m.

Referenced by mainTransfection().

Here is the caller graph for this function:



13.28 examples/mRNA_transfection/simulate_mRNA_Transfection.m File Reference

simulate_mRNA_Transfection for examples/mRNA_transfection

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, Y >,mlhsInnerSubst< matlabtypesubstitute, X > > simulate_mRNA_Transfection (matlabtypesubstitute theta, matlabtypesubstitute t)

    simulate_mRNA_Transfection for examples/mRNA_transfection

13.28.1 Detailed Description

simulate_mRNA_Transfection for examples/mRNA_transfection

13.28.2 Function Documentation

13.28.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, Y >,mlhsInnerSubst< matlabtypesubstitute, X > > simulate_mRNA_Transfection (matlabtypesubstitute *theta*, matlabtypesubstitute *t*)

simulate_mRNA_Transfection for examples/mRNA_transfection

simulateConversionReaction performs a simulation of the transfection model for the given timepoints t and parameters theta

Parameters

theta	Model parameters [theta_1, theta_2]'
t	vector of time points

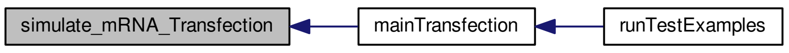
Return values

Y	Vector with values of the observables Y = [X_2] at timepoints t
X	State vector at timepoints t

Definition at line 17 of file simulate_mRNA_Transfection.m.

Referenced by mainTransfection().

Here is the caller graph for this function:



13.29 examples/Pom1p_gradient_formation/logLikelihoodPom1.m File Reference

Calculates the MLE vector and gradient for fmincon for the kinetic parameters.

Functions

- mlhsInnerSubst< matlabtypesubstitute, varargout > logLikelihoodPom1 (matlabtypesubstitute varargin)
  Calculates the MLE vector and gradient for fmincon for the kinetic parameters.

13.29.1 Detailed Description

Calculates the MLE vector and gradient for fmincon for the kinetic parameters.

13.29.2 Function Documentation

13.29.2.1 mlhsInnerSubst< matlabtypesubstitute, varargout > logLikelihoodPom1 (matlabtypesubstitute *varargin*)

Calculates the MLE vector and gradient for fmincon for the kinetic parameters.

USAGE

[...] = logLikelihoodPom1(parameter,options)

INPUTS

parameter ... log-parameter values in the order given by options.name options ... structure needed to evaluate the Likelihood .options.datafile path to load options.Data file .disc structure containing the PDE solver information .name parameter names .sign default: negative .grad_ind default: 1:length(p) .type default: `none`

Outputs

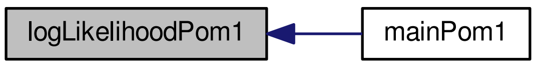
F ... negative log-Likelihood dF ... gradient of negative log-Likelihood in the order of options.name

2013/04/20 Sabrina Hock 2016/10/17 Anna Fiedler

Definition at line 17 of file logLikelihoodPom1.m.

Referenced by mainPom1().

Here is the caller graph for this function:



13.30 examples/Pom1p_gradient_formation/mainPom1.m File Reference

Main file of Pom1 example.

Functions

- noret::substitute mainPom1 ()
  Main file of Pom1 example.

13.30.1 Detailed Description

Main file of Pom1 example.

13.30.2 Function Documentation

13.30.2.1 noret::substitute mainPom1 ( )

Main file of Pom1 example.

Demonstrates the use of

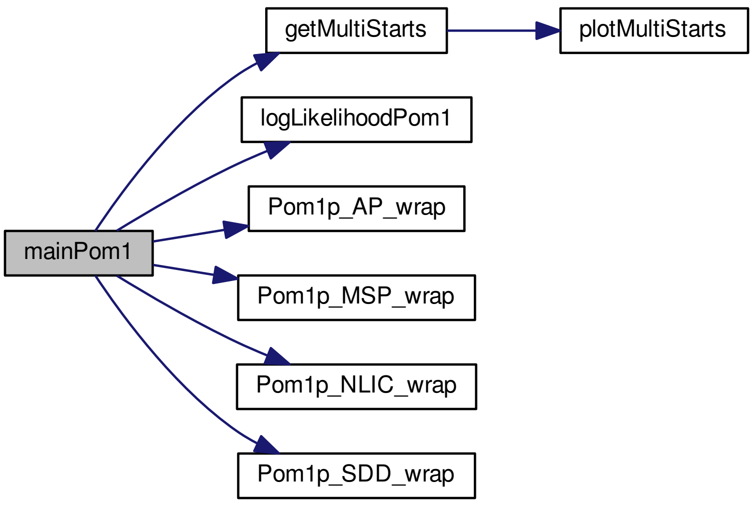
- getMultiStarts()

This example provides four models for Pom1 gradient formation. The SDD and NLIC were adapted from Saunders et al. (2012). The AP was adapted from Hachet et al. (2011) and the MSP from Hersch et al. (2015).

Definition at line 17 of file mainPom1.m.

References getMultiStarts(), logLikelihoodPom1(), Pom1p_AP_wrap(), Pom1p_MSP_wrap(), Pom1p_NLIC_wrap(), and Pom1p_SDD_wrap().

Here is the call graph for this function:



13.31 examples/Pom1p_gradient_formation/Models/Pom1p_AP_wrap.m File Reference

clear all close all clc

Functions

- noret::substitute Pom1p_AP_wrap ()
  clear all close all clc

13.31.1 Detailed Description

clear all close all clc

13.32 examples/Pom1p_gradient_formation/Models/Pom1p_MSP_wrap.m File Reference

clear all close all clc

Functions

- noret::substitute Pom1p_MSP_wrap ()
  clear all close all clc

13.32.1 Detailed Description

clear all close all clc

13.33 examples/Pom1p_gradient_formation/Models/Pom1p_NLIC_wrap.m File Reference

clear all close all clc

Functions

- noret::substitute Pom1p_NLIC_wrap ()
  clear all close all clc

13.33.1 Detailed Description

clear all close all clc

13.34 examples/Pom1p_gradient_formation/Models/Pom1p_SDD_wrap.m File Reference

clear all close all clc

Functions

- noret::substitute Pom1p_SDD_wrap ()

  clear all close all clc

13.34.1 Detailed Description

clear all close all clc

13.35 examples/RingExample/mainExampleRing.m File Reference

mainExampleGauss.m shows how to use sampling methods in PESTO. The example problem is a smudged hyper-ring. Its properties can be altered in define_Gauss_LLH.m and its dimension via ringDimension.

Functions

- noret::substitute mainExampleRing ()

  mainExampleGauss.m shows how to use sampling methods in PESTO. The example problem is a smudged hyper-ring. Its properties can be altered in define_Gauss_LLH.m and its dimension via ringDimension.

13.35.1 Detailed Description

mainExampleGauss.m shows how to use sampling methods in PESTO. The example problem is a smudged hyper-ring. Its properties can be altered in define_Gauss_LLH.m and its dimension via ringDimension.

13.35.2 Function Documentation

13.35.2.1 noret::substitute mainExampleRing ( )

mainExampleGauss.m shows how to use sampling methods in PESTO. The example problem is a smudged hyper-ring. Its properties can be altered in define_Gauss_LLH.m and its dimension via ringDimension.

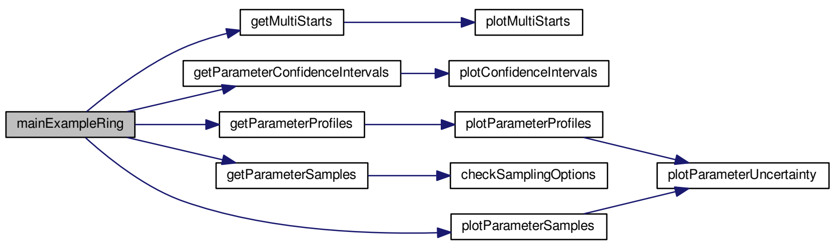
Written by Benjamin Ballnus 2/2017

Definition at line 17 of file mainExampleRing.m.

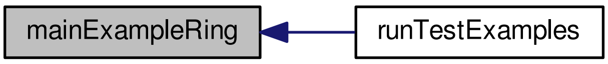
References getMultiStarts(), getParameterConfidenceIntervals(), getParameterProfiles(), getParameterSamples(), and plotParameterSamples().

Referenced by runTestExamples().

Here is the call graph for this function:



Here is the caller graph for this function:



13.36 examples/runTestExamples.m File Reference

runTestExamples runs some examples for the PESTO toolbox, to test if new implementations cause problems

Functions

- noret::substitute runTestExamples ()

  runTestExamples runs some examples for the PESTO toolbox, to test if new implementations cause problems

13.36.1 Detailed Description

runTestExamples runs some examples for the PESTO toolbox, to test if new implementations cause problems

13.37 getMultiStarts.m File Reference

getMultiStarts() computes the maximum a posterior estimate of the parameters of a user-supplied posterior function. Therefore, a multi-start local optimization is used. The parameters from the best value of the posterior function arethen used as the global optimum. To ensure that the found maximum is a global one, a sufficiently high number of multistarts must be done. Those starts can be initialized with either randomly sampled parameter values, following either a uniform distribution or a latin hypercube, or they can be sampled by a user provided initial function (provided as option.init_fun).

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, parameters >,mlhsInnerSubst< matlabtypesubstitute, fh > > getMultiStarts (matlabtypesubstitute parameters, matlabtypesubstitute objective_function, matlabtypesubstitute varargin)

  getMultiStarts() computes the maximum a posterior estimate of the parameters of a user-supplied posterior function. Therefore, a multi-start local optimization is used. The parameters from the best value of the posterior function arethen used as the global optimum. To ensure that the found maximum is a global one, a sufficiently high number of multistarts must be done. Those starts can be initialized with either randomly sampled parameter values, following either a uniform distribution or a latin hypercube, or they can be sampled by a user provided initial function (provided as option.init_fun).
- mlhsInnerSubst< matlabtypesubstitute, varargout > **mtoc_subst_getMultiStarts_m_tsbus_cotm_obj** (matlabtypesubstitute varargin)
- mlhsInnerSubst< matlabtypesubstitute, varargout > **mtoc_subst_getMultiStarts_m_tsbus_cotm_obj_↩ w_error_count** (matlabtypesubstitute varargin)
- mlhsInnerSubst< matlabtypesubstitute, stringTimePrediction > **mtoc_subst_getMultiStarts_m_tsbus_↩ cotm_updateWaitBar** (matlabtypesubstitute timePredicted)
- noret::substitute **mtoc_subst_getMultiStarts_m_tsbus_cotm_saveResults** (matlabtypesubstitute parameters, matlabtypesubstitute options, matlabtypesubstitute i)

13.37.1 Detailed Description

getMultiStarts() computes the maximum a posterior estimate of the parameters of a user-supplied posterior function. Therefore, a multi-start local optimization is used. The parameters from the best value of the posterior function arethen used as the global optimum. To ensure that the found maximum is a global one, a sufficiently high number of multistarts must be done. Those starts can be initialized with either randomly sampled parameter values, following either a uniform distribution or a latin hypercube, or they can be sampled by a user provided initial function (provided as option.init_fun).

13.37.2 Function Documentation

13.37.2.1 mlhsSubst$<$ mlhsInnerSubst$<$ matlabtypesubstitute, parameters $>$,mlhsInnerSubst$<$ matlabtypesubstitute, fh $>$ $>$ getMultiStarts (matlabtypesubstitute *parameters,* matlabtypesubstitute *objective_function,* matlabtypesubstitute *varargin*)

getMultiStarts() computes the maximum a posterior estimate of the parameters of a user-supplied posterior function. Therefore, a multi-start local optimization is used. The parameters from the best value of the posterior function arethen used as the global optimum. To ensure that the found maximum is a global one, a sufficiently high number of multistarts must be done. Those starts can be initialized with either randomly sampled parameter values, following either a uniform distribution or a latin hypercube, or they can be sampled by a user provided initial function (provided as option.init_fun).

Note: This function can exploit up to (n_start + 1) workers when running in `parallel` mode.

USAGE

- [...] = getMultiStarts(parameters,objective_function)
- [...] = getMultiStarts(parameters,objective_function,options)
- [parameters,fh] = getMultiStarts(...)

getMultiStarts() uses the following PestoOptions members

- PestoOptions::start_index
- PestoOptions::n_starts
- PestoOptions::mode
- PestoOptions::fh
- PestoOptions::fmincon
- PestoOptions::rng
- PestoOptions::proposal
- PestoOptions::save
- PestoOptions::foldername
- PestoOptions::trace
- PestoOptions::comp_type
- PestoOptions::tempsave
- PestoOptions::resetobjective
- PestoOptions::obj_type
- PestoOptions::init_threshold
- PestoOptions::plot_options

History

- 2012/05/31 Jan Hasenauer
- 2012/07/11 Jan Hasenauer
- 2014/06/11 Jan Hasenauer
- 2015/07/28 Fabian Froehlich
- 2015/11/10 Fabian Froehlich
- 2016/06/07 Paul Stapor
- 2016/10/04 Daniel Weindl
- 2016/12/04 Paul Stapor

Parameters

parameters	parameter struct
objective_function	objective function to be optimized. This function should accept one input, the parameter vector.
varargin	`1 getMultiStarts ( ..., options )` *Required Parameters for varargin:* • options A PestoOptions object holding various options for the algorithm.

Return values

parameters	updated parameter object
fh	figure handle

Required fields of parameters:

- `number --` Number of parameters
- `min --` Lower bound for each parameter
- `max --` upper bound for each parameter name = {`name1`, ...}: names of the parameters
- `guess --` initial guess for the parameters (Optional, will be initialized empty if not provided)
- `init_fun --` function to draw starting points for local optimization, must have the structure init_↩ fun(theta_0, theta_min, theta_max). (Only required if proposal == `user-supplied`)

Generated fields of parameters:

- `MS --` information about multi-start optimization
  - par0(:,i): starting point yielding ith MAP
  - par(:,i): ith MAP
  - logPost(i): log-posterior for ith MAP
  - logPost0(i): log-posterior for starting point yielding ith MAP
  - gradient(_,i): gradient of log-posterior at ith MAP
  - hessian(:,:,i): hessian of log-posterior at ith MAP
  - n_objfun(i): # objective evaluations used to calculate ith MAP
  - n_iter(i): # iterations used to calculate ith MAP
  - t_cpu(i): CPU time for calculation of ith MAP

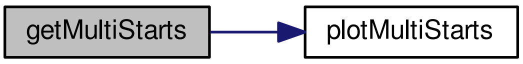
- exitflag(i): exitflag the optimizer returned for ith MAP
- par_trace(:,:,i): parameter trace for ith MAP (if options.trace == true)
- fval_trace(:,i): objective function value trace for ith MAP (if options.trace == true)
- time_trace(:,i): computation time trace for ith MAP (if options.trace == true)

Definition at line 17 of file getMultiStarts.m.

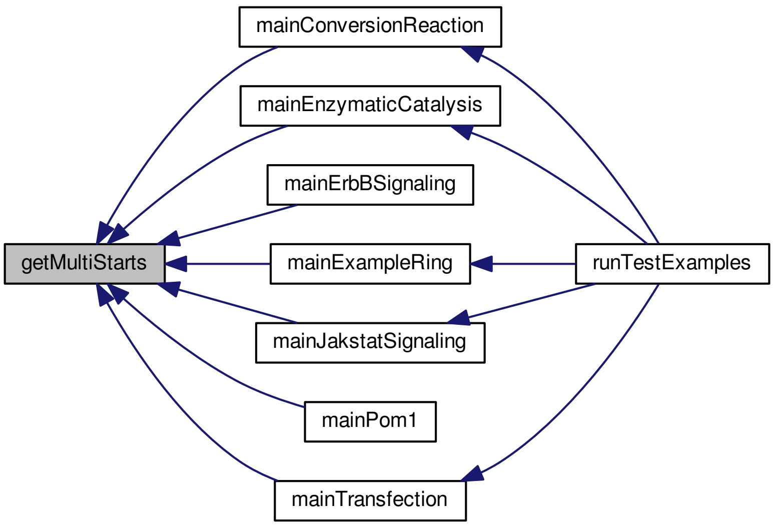
References plotMultiStarts().

Referenced by mainConversionReaction(), mainEnzymaticCatalysis(), mainErbBSignaling(), mainExampleRing(), mainJakstatSignaling(), mainPom1(), and mainTransfection().

Here is the call graph for this function:



Here is the caller graph for this function:



13.38 getParameterConfidenceIntervals.m File Reference

getParameterConfidenceIntervals() calculates the confidence intervals for the model parameters. This is done by four approaches: The values of CI.local_PL and CI.PL are determined by the point on which a threshold according to the confidence level alpha (calculated by a chi2-distribution) is reached. local_PL computes this point by a local approximation around the MAP estimate using the Hessian matrix, PL uses the profile likelihoods instead. The value of CI.local_B is computed by using the cummulative distribution function of a local approximation of the profile based on the Hessian matrix at the MAP estimate. The value of CI.S is calculated using samples for the model parameters and the according percentiles based on the confidence levels alpha.

Functions

- mlhsInnerSubst$<$ matlabtypesubstitute, parameters $>$ getParameterConfidenceIntervals (matlabtypesubstitute parameters, matlabtypesubstitute alpha, matlabtypesubstitute varargin)

  getParameterConfidenceIntervals() calculates the confidence intervals for the model parameters. This is done by four approaches: The values of CI.local_PL and CI.PL are determined by the point on which a threshold according to the confidence level alpha (calculated by a chi2-distribution) is reached. local_PL computes this point by a local approximation around the MAP estimate using the Hessian matrix, PL uses the profile likelihoods instead. The value of CI.local_B is computed by using the cummulative distribution function of a local approximation of the profile based on the Hessian matrix at the MAP estimate. The value of CI.S is calculated using samples for the model parameters and the according percentiles based on the confidence levels alpha.

13.38.1 Detailed Description

getParameterConfidenceIntervals() calculates the confidence intervals for the model parameters. This is done by four approaches: The values of CI.local_PL and CI.PL are determined by the point on which a threshold according to the confidence level alpha (calculated by a chi2-distribution) is reached. local_PL computes this point by a local approximation around the MAP estimate using the Hessian matrix, PL uses the profile likelihoods instead. The value of CI.local_B is computed by using the cummulative distribution function of a local approximation of the profile based on the Hessian matrix at the MAP estimate. The value of CI.S is calculated using samples for the model parameters and the according percentiles based on the confidence levels alpha.

13.38.2 Function Documentation

13.38.2.1 mlhsInnerSubst$<$ matlabtypesubstitute, parameters $>$ getParameterConfidenceIntervals (matlabtypesubstitute *parameters*, matlabtypesubstitute *alpha*, matlabtypesubstitute *varargin*)

getParameterConfidenceIntervals() calculates the confidence intervals for the model parameters. This is done by four approaches: The values of CI.local_PL and CI.PL are determined by the point on which a threshold according to the confidence level alpha (calculated by a chi2-distribution) is reached. local_PL computes this point by a local approximation around the MAP estimate using the Hessian matrix, PL uses the profile likelihoods instead. The value of CI.local_B is computed by using the cummulative distribution function of a local approximation of the profile based on the Hessian matrix at the MAP estimate. The value of CI.S is calculated using samples for the model parameters and the according percentiles based on the confidence levels alpha.

USAGE

- parameters = getParameterConfidenceIntervals(parameters, alpha)

History

- 2013/11/29 Jan Hasenauer
- 2016/12/01 Paul Stapor

Parameters

parameters	parameter struct
alpha	vector with desired confidence levels for the intervals

Return values

parameters	updated parameter struct

Required fields of parameters:

Generated fields of parameters:

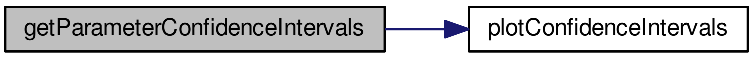
- `CI --` Information about confidence levels
  - local_PL: Threshold based approach, uses a local approximation by the Hessian matrix at the MAP estimate (requires parameters.MS, e.g. from getMultiStarts)
  - PL: Threshold based approach, uses profile likelihoods (requires parameters.P, e.g. from get↩ParameterProfiles)
  - local_B: Mass based approach, uses a local approximation by the Hessian matrix at the MAP estimate (requires parameters.MS, e.g. from getMultiStarts)
  - S: Bayesian approach, uses percentiles based on samples (requires parameters.S, e.g. from get↩ParameterSamples)

Definition at line 17 of file getParameterConfidenceIntervals.m.

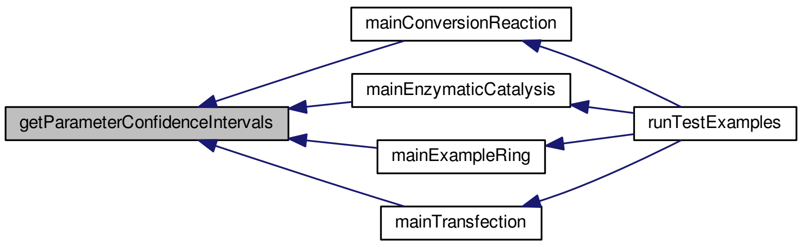
References plotConfidenceIntervals().

Referenced by mainConversionReaction(), mainEnzymaticCatalysis(), mainExampleRing(), and mainTransfection().

Here is the call graph for this function:



Here is the caller graph for this function:



13.39 getParameterProfiles.m File Reference

getParameterProfiles.m calculates the profiles likelihoods for the model parameters, starting from the maximum a posteriori estimate. This calculation is done by fixing the i-th parameter and repeatedly reoptimizing the likelihood/posterior estimate (for all i). The initial guess for the next reoptimization point is computed by extrapolation from the previous points to ensure a quick optimization.

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, parameters >,mlhsInnerSubst< matlabtypesubstitute, fh > > getParameterProfiles (matlabtypesubstitute parameters, matlabtypesubstitute objective_function, matlabtypesubstitute varargin)

  getParameterProfiles.m calculates the profiles likelihoods for the model parameters, starting from the maximum a posteriori estimate. This calculation is done by fixing the i-th parameter and repeatedly reoptimizing the likelihood/posterior estimate (for all i). The initial guess for the next reoptimization point is computed by extrapolation from the previous points to ensure a quick optimization.

13.39.1 Detailed Description

getParameterProfiles.m calculates the profiles likelihoods for the model parameters, starting from the maximum a posteriori estimate. This calculation is done by fixing the i-th parameter and repeatedly reoptimizing the likelihood/posterior estimate (for all i). The initial guess for the next reoptimization point is computed by extrapolation from the previous points to ensure a quick optimization.

13.39.2 Function Documentation

13.39.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, parameters >,mlhsInnerSubst< matlabtypesubstitute, fh > > getParameterProfiles (matlabtypesubstitute *parameters,* matlabtypesubstitute *objective_function,* matlabtypesubstitute *varargin*)

getParameterProfiles.m calculates the profiles likelihoods for the model parameters, starting from the maximum a posteriori estimate. This calculation is done by fixing the i-th parameter and repeatedly reoptimizing the likelihood/posterior estimate (for all i). The initial guess for the next reoptimization point is computed by extrapolation from the previous points to ensure a quick optimization.

Note: This function can exploit up to (n_theta + 1) workers when running in `parallel` mode.

USAGE

[...] = getParameterProfiles(parameters, objective_function) [...] = getParameterProfiles(parameters, objective_function, options) [parameters, fh] = getParameterProfiles(...)

getParameterProfiles() uses the following PestoOptions members

- PestoOptions::calc_profiles
- PestoOptions::comp_type
- PestoOptions::dJ
- PestoOptions::dR_max
- PestoOptions::fh
- PestoOptions::MAP_index
- PestoOptions::mode
- PestoOptions::obj_type
- PestoOptions::options_getNextPoint .guess .min .max .update .mode
- PestoOptions::parameter_index
- PestoOptions::parameter_method_index
- PestoOptions::profile_method
- PestoOptions::profileReoptimizationOptions

- PestoOptions::plot_options
- PestoOptions::R_min
- PestoOptions::save

History

- 2012/05/16 Jan Hasenauer
- 2014/06/12 Jan Hasenauer
- 2016/10/04 Daniel Weindl
- 2016/10/12 Paul Stapor

Parameters

parameters	parameter struct
objective_function	objective function to be optimized. This function should accept one input, the parameter vector.
varargin	`1 getParameterProfiles ( ..., options )` *Required Parameters for varargin:* • options A PestoOptions object holding various options for the algorithm.

Return values

parameters	updated parameter struct
fh	figure handle

Required fields of parameters:

- `number --` Number of parameters
- `min --` Lower bound for each parameter
- `max --` upper bound for each parameter name = {`name1`, ...}: names of the parameters
- `MS --` results of global optimization, obtained using for instance the routine `getMultiStarts.m`. MS has to contain at least
  - par: sorted list n_theta x n_starts of parameter estimates. The first entry is assumed to be the best one.
  - logPost: sorted list n_starts x 1 of of log-posterior values corresponding to the parameters listed in .par.
  - hessian: Hessian matrix (or approximation) at the optimal point

Generated fields of parameters:

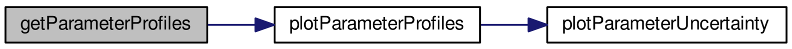
- `P(i) --` profile for i-th parameter
  - par: MAPs along profile
  - logPost: maximum log-posterior along profile
  - R: ratio

Definition at line 17 of file getParameterProfiles.m.

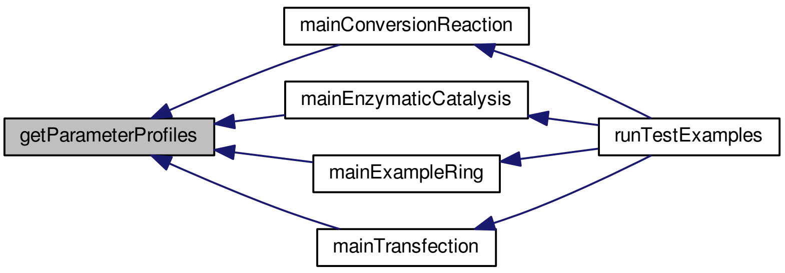
References plotParameterProfiles().

Referenced by mainConversionReaction(), mainEnzymaticCatalysis(), mainExampleRing(), and mainTransfection().

Here is the call graph for this function:



Here is the caller graph for this function:



13.40 getParameterSamples.m File Reference

getParameterSamples.m performs MCMC sampling of the posterior distribution. Note, the DRAM library routine tooparameters.minox is used internally. This function is capable of sampling with MH, AM, DRAM, MALA, PT and PHS. The sampling plotting routines should no longer be contained in here but as standalone scripts capable of using the resulting par.S.

Functions

- mlhsInnerSubst< matlabtypesubstitute, parameters > getParameterSamples (matlabtypesubstitute parameters, matlabtypesubstitute objFkt, matlabtypesubstitute opt)

    getParameterSamples.m performs MCMC sampling of the posterior distribution. Note, the DRAM library routine tooparameters.minox is used internally. This function is capable of sampling with MH, AM, DRAM, MALA, PT and PHS. The sampling plotting routines should no longer be contained in here but as standalone scripts capable of using the resulting par.S.

13.40.1 Detailed Description

getParameterSamples.m performs MCMC sampling of the posterior distribution. Note, the DRAM library routine tooparameters.minox is used internally. This function is capable of sampling with MH, AM, DRAM, MALA, PT and PHS. The sampling plotting routines should no longer be contained in here but as standalone scripts capable of using the resulting par.S.

13.40.2 Function Documentation

13.40.2.1 mlhsInnerSubst< matlabtypesubstitute, parameters > getParameterSamples (matlabtypesubstitute *parameters,* matlabtypesubstitute *objFkt,* matlabtypesubstitute *opt*)

getParameterSamples.m performs MCMC sampling of the posterior distribution. Note, the DRAM library routine tooparameters.minox is used internally. This function is capable of sampling with MH, AM, DRAM, MALA, PT and PHS. The sampling plotting routines should no longer be contained in here but as standalone scripts capable of using the resulting par.S.

parameters: parameter struct covering model options and results obtained by optimization, profiles and sampling. Optimization results can be used for initialization. The parameter struct should

at least contain

par.min: Lower parameter bounds par.max: Upper parameter bounds par.number: Number of parameters par.obj_type: Type of objective function, e.g. `log-posterior` objFkt: Objective function which measures the difference of model output and data opt : An options object holding various options for the sampling. Depending on the algorithm and particular flavor,

different options must be set

— General — opt.rndSeed: Either a number or `shuffle` opt.nIterations: Number of iterations, e.g. 1e6 opt.↩
samplingAlgorithm: Specifies the code body which will be used.

Further options (details below) depend on the choice made here

`DRAM` for Delayed Rejection Adaptive Metropolis `MALA` for Metropolis Adaptive Langevin Algorithm `PT` (default) for Metropolis-Hastings, Adaptive Metropolis, Parallel Tempering `PHS`for Parallel Hierarchical Sampling opt.theta0: Initial points for all chains. If the algorithm uses multiple chains (as `PT`), one can specify multiple theta0 as in example: opt.theta0 = repmat([0.1,1.05,-2.5,-0.5,0.4],opt.nTemps,1)'; If there is just one chain, please specify as opt.theta0 = [1;2;3;4]; It is recommendet to set theta0 by taking into account the results from a preceeding optimization. opt.sigma0: Initial covariance matrix for all chains. Example for single-chain algorithms: opt.sigma0 = 1e5∗diag(ones(1,5)); Example for multi-chain algorithms : opt.sigma0 = repmat(1e5∗diag(ones(1,5)),opt.nTemps,1); It is recommendet to set sigma0 by taking into account the results from a preceeding optimization.

— Delayed Rejection Adaptive Metropolis — opt.DRAM.regFactor : This factor is used for regularization in cases where the single-chain proposal covariance matrices are ill conditioned. Larger values equal stronger regularization. opt.DRAM.nTry : The number of tries in the delayed rejection scheme opt.DRAM.verbosityMode : Defines the level of verbosity `silent`, `visual`, `debug` or `text` opt.DRAM.adaptionInterval : Updates the proposal density only every opt.DRAM.adaptionInterval time

— Metropolis Adaptive Langevin Algorithm — Note: This algorithm uses gradients & hessian either calculated by sensitivites or finite differences. opt.MALA.regFactor : This factor is used for regularization in cases where the proposal covariance matrices are ill conditioned. Larger values equal stronger regularization.

— Parallel Tempering — opt.PT.nTemps: Initial number of temperatures (default 10) opt.PT.exponentT: The initial temperatures are set by a power law to ^opt.exponentT. (default 4) opt.PT.alpha: Parameter which controlls the adaption degeneration velocity of the single-chain proposals. Value between 0 and 1. Default 0.51. No adaption for value = 0. opt.PT.temperatureAlpha: Parameter which controlls the adaption degeneration velocity of the temperature adaption. Value between 0 and 1. Default 0.51. No effect for value = 0. opt.PT.memoryLength: The higher the value the more it lowers the impact of early adaption steps. Default 1. opt.PT.regFactor: Regularization factor for ill conditioned covariance matrices of the adapted proposal density. Regularization might happen if the eigenvalues of the covariance matrix strongly differ in order of magnitude. In this case, the algorithm adds a small diag-matrix to the covariance matrix with elements opt.regFactor. opt.PT.temperatureAdaptionScheme: Follows the temperature adaption scheme from `Vousden16` or `Lacki15`. Can be set to `none` for no temperature adaption.

— Parallel Hierarchical Sampling — opt.PHS.nChains : Number of chains (1 `mother`-chain and opt.PHS.n←
Chains-1 auxillary chains) opt.PHS.alpha : Control parameter for adaption decay. Needs values between 0 and 1. Higher values lead to faster decays, meaning that new iterations influence the single-chain proposal adaption only very weakly very quickly. opt.PHS.memoryLength : Control parameter for adaption. Higher values supress strong ealy adaption. opt.PHS.regFactor : This factor is used for regularization in cases where the single-chain proposal covariance matrices are ill conditioned. nChainsarger values equal stronger regularization. opt.PHS.trainingTime : The iterations before the first chain swap is invoked

History

- 2012/07/11 Jan Hasenauer
- 2015/04/29 Jan Hasenauer
- 2016/10/17 Benjamin Ballnus
- 2016/10/19 Daniel Weindl
- 2016/11/04 Paul Stapor
- 2017/02/01 Benjamin Ballnus

Required fields of parameters:

Required fields of opt:

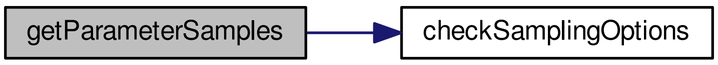
Generated fields of parameters:

Definition at line 17 of file getParameterSamples.m.

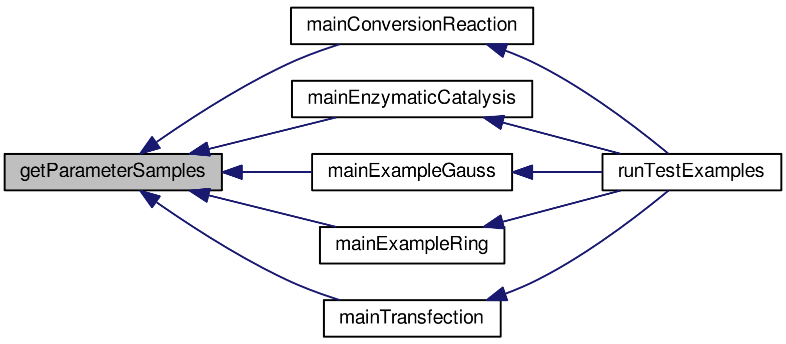
References checkSamplingOptions().

Referenced by mainConversionReaction(), mainEnzymaticCatalysis(), mainExampleGauss(), mainExampleRing(), and mainTransfection().

Here is the call graph for this function:



Here is the caller graph for this function:



13.41 getPropertyConfidenceIntervals.m File Reference

getPropertyConfidenceIntervals.m calculates the confidence intervals for the model properties. This is done by three approaches: The values of CI.local_PL and CI.PL are determined by the point on which a threshold according to the confidence level alpha (calculated by a chi2-distribution) is reached. local_PL computes this point by a local approximation around the MAP estimate using the Hessian matrix, PL uses the profile likelihoods instead. The value of CI.local_B is computed by using the cummulative distribution function of a local approximation of the profile based on the Hessian matrix at the MAP estimate.

Functions

- mlhsInnerSubst< matlabtypesubstitute, properties > getPropertyConfidenceIntervals (matlabtypesubstitute properties, matlabtypesubstitute alpha, matlabtypesubstitute varargin)

  getPropertyConfidenceIntervals.m calculates the confidence intervals for the model properties. This is done by three approaches: The values of CI.local_PL and CI.PL are determined by the point on which a threshold according to the confidence level alpha (calculated by a chi2-distribution) is reached. local_PL computes this point by a local approximation around the MAP estimate using the Hessian matrix, PL uses the profile likelihoods instead. The value of CI.local_B is computed by using the cummulative distribution function of a local approximation of the profile based on the Hessian matrix at the MAP estimate.

13.41.1 Detailed Description

getPropertyConfidenceIntervals.m calculates the confidence intervals for the model properties. This is done by three approaches: The values of CI.local_PL and CI.PL are determined by the point on which a threshold according to the confidence level alpha (calculated by a chi2-distribution) is reached. local_PL computes this point by a local approximation around the MAP estimate using the Hessian matrix, PL uses the profile likelihoods instead. The value of CI.local_B is computed by using the cummulative distribution function of a local approximation of the profile based on the Hessian matrix at the MAP estimate.

13.41.2 Function Documentation

13.41.2.1 mlhsInnerSubst< matlabtypesubstitute, properties > getPropertyConfidenceIntervals (matlabtypesubstitute *properties,* matlabtypesubstitute *alpha,* matlabtypesubstitute *varargin*)

getPropertyConfidenceIntervals.m calculates the confidence intervals for the model properties. This is done by three approaches: The values of CI.local_PL and CI.PL are determined by the point on which a threshold according to the confidence level alpha (calculated by a chi2-distribution) is reached. local_PL computes this point by a local approximation around the MAP estimate using the Hessian matrix, PL uses the profile likelihoods instead. The value of CI.local_B is computed by using the cummulative distribution function of a local approximation of the profile based on the Hessian matrix at the MAP estimate.

USAGE

- properties = getPropertyConfidenceIntervals(properties, alpha)

History

- 2013/11/29 Jan Hasenauer
- 2016/12/01 Paul Stapor

Parameters

properties	property struct
alpha	vector with desired confidence levels for the intervals

Return values

properties	updated properties struct

Required fields of properties:

Generated fields of properties:

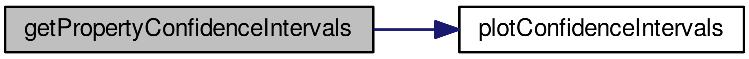
- `CI --` Information about confidence levels
  - local_PL: Threshold based approach, uses a local approximation by the Hessian matrix at the MAP estimate (requires parameters.MS, e.g. from getMultiStarts)
  - PL: Threshold based approach, uses profile likelihoods (requires parameters.P, e.g. from get↩ParameterProfiles)
  - local_B: Mass based approach, uses a local approximation by the Hessian matrix at the MAP estimate (requires parameters.MS, e.g. from getMultiStarts)

Definition at line 17 of file getPropertyConfidenceIntervals.m.

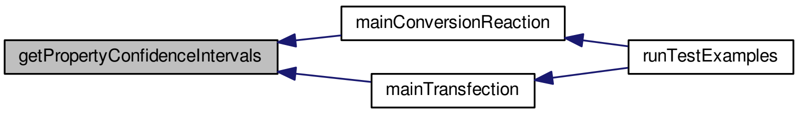
References plotConfidenceIntervals().

Referenced by mainConversionReaction(), and mainTransfection().

Here is the call graph for this function:



Here is the caller graph for this function:



13.42 getPropertyMultiStarts.m File Reference

getPropertyMultiStarts.m evaluates the properties for the different mutli-start results.

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, properties >,mlhsInnerSubst< matlabtypesubstitute, fh > > getPropertyMultiStarts (matlabtypesubstitute properties, matlabtypesubstitute parameters, matlabtypesubstitute varargin)

  getPropertyMultiStarts.m evaluates the properties for the different mutli-start results.

13.42.1 Detailed Description

getPropertyMultiStarts.m evaluates the properties for the different mutli-start results.

13.42.2 Function Documentation

13.42.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, properties >,mlhsInnerSubst< matlabtypesubstitute, fh > > getPropertyMultiStarts (matlabtypesubstitute *properties,* matlabtypesubstitute *parameters,* matlabtypesubstitute *varargin*)

getPropertyMultiStarts.m evaluates the properties for the different mutli-start results.

USAGE

[...] = getPropertyMultiStarts(properties,parameters) [...] = getPropertyMultiStarts(properties,parameters,options) [parameters,fh] = getPropertyMultiStarts(...)

getPropertyMultiStarts() uses the following PestoOptions members

- PestoOptions::mode
- PestoOptions::fh
- PestoOptions::save
- PestoOptions::foldername
- PestoOptions::comp_type

History

- 2015/03/03 Jan Hasenauer
- 2016/04/10 Daniel Weindl

Parameters

properties	property struct containing at least:
parameters	parameter struct containing at least:
varargin	`1 getPropertyMultiStarts ( ..., MS, number, min, max, options )` *Required Parameters for varargin:* • MS information about multi-start optimization • number Number of properties • min lower bound for property values • max upper bound for property values name = {`name1`,...}: names of the properties function = {`function1`,...}: functions to evaluate property values. These functions provide the values of the respective properties and the corresponding 1st and 2nd order derivatives. • options A PestoOptions object holding the options for the algorithm.

Return values

properties	updated parameter object containing:
fh	figure handle
MS	properties for multi-start optimization results • par(:,i): ith MAP • logPost(i): log-posterior for ith MAP • exitflag(i): exit flag of ith MAP • prop(j,i): values of jth property for ith MAP • prop_Sigma(:,:,i): covariance of properties for ith MAP

Required fields of parameters:

Required fields of properties:

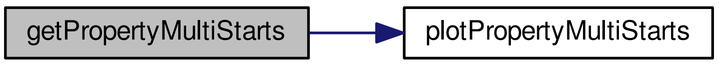
Generated fields of properties:

Definition at line 17 of file getPropertyMultiStarts.m.

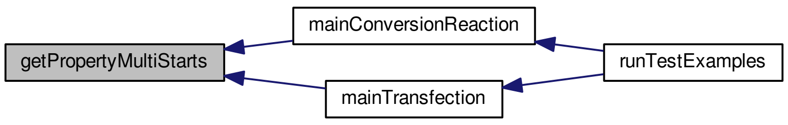
References plotPropertyMultiStarts().

Referenced by mainConversionReaction(), and mainTransfection().

Here is the call graph for this function:



Here is the caller graph for this function:



13.43 getPropertyProfiles.m File Reference

getPropertyProfiles.m calculates the profiles of user-supplied property functions, starting from the maximum a posteriori estimate. This calculation is done by varying the value of each property function respectively, starting from the value of this function at the global optimum and by reoptimizing the likelihood/posterior estimate in each variational step of the property. The initial guess for the next reoptimization point is computed by extrapolation from the previous points to ensure a quick optimization.

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, properties >,mlhsInnerSubst< matlabtypesubstitute, fh > > getPropertyProfiles (matlabtypesubstitute properties, matlabtypesubstitute parameters, matlabtypesubstitute objective_function, matlabtypesubstitute varargin)

  getPropertyProfiles.m calculates the profiles of user-supplied property functions, starting from the maximum a posteriori estimate. This calculation is done by varying the value of each property function respectively, starting from the value of this function at the global optimum and by reoptimizing the likelihood/posterior estimate in each variational step of the property. The initial guess for the next reoptimization point is computed by extrapolation from the previous points to ensure a quick optimization.
- mlhsInnerSubst< matlabtypesubstitute, varargout > **mtoc_subst_getPropertyProfiles_m_tsbus_cotm↩ _obj** (matlabtypesubstitute theta, matlabtypesubstitute fun, matlabtypesubstitute type)
- mlhsInnerSubst< matlabtypesubstitute, varargout > **mtoc_subst_getPropertyProfiles_m_tsbus_cotm↩ _obj_con** (matlabtypesubstitute theta, matlabtypesubstitute fun, matlabtypesubstitute fun_min, matlabtypesubstitute type)
- mlhsInnerSubst< matlabtypesubstitute, varargout > **mtoc_subst_getPropertyProfiles_m_tsbus_cotm_↩ prop_fun** (matlabtypesubstitute theta, matlabtypesubstitute fun, matlabtypesubstitute prop_min, matlabtypesubstitute prop_max, matlabtypesubstitute s)
- mlhsInnerSubst< matlabtypesubstitute, varargout > **mtoc_subst_getPropertyProfiles_m_tsbus_cotm↩ _prop_con_fun** (matlabtypesubstitute theta, matlabtypesubstitute fun, matlabtypesubstitute prop_min, matlabtypesubstitute prop_max, matlabtypesubstitute s)

13.43.1 Detailed Description

getPropertyProfiles.m calculates the profiles of user-supplied property functions, starting from the maximum a posteriori estimate. This calculation is done by varying the value of each property function respectively, starting from the value of this function at the global optimum and by reoptimizing the likelihood/posterior estimate in each variational step of the property. The initial guess for the next reoptimization point is computed by extrapolation from the previous points to ensure a quick optimization.

13.43.2 Function Documentation

13.43.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, properties >,mlhsInnerSubst< matlabtypesubstitute, fh > > getPropertyProfiles (matlabtypesubstitute *properties*, matlabtypesubstitute *parameters*, matlabtypesubstitute *objective_function*, matlabtypesubstitute *varargin*)

getPropertyProfiles.m calculates the profiles of user-supplied property functions, starting from the maximum a posteriori estimate. This calculation is done by varying the value of each property function respectively, starting from the value of this function at the global optimum and by reoptimizing the likelihood/posterior estimate in each variational step of the property. The initial guess for the next reoptimization point is computed by extrapolation from the previous points to ensure a quick optimization.

Note: This function can exploit up to (n_theta + 1) workers when running in `parallel` mode.

USAGE

[...] = getPropertyProfiles(properties, parameters, objective_function) [...] = getPropertyProfiles(properties, parameters, objective_function, options) [parameters, fh] = getPropertyProfiles(...)

%

getPropertyProfiles() uses the following PestoOptions members

- PestoOptions::boundary

- PestoOptions::calc_profiles
- PestoOptions::comp_type
- PestoOptions::dJ
- PestoOptions::dR_max
- PestoOptions::fh
- PestoOptions::fmincon
- PestoOptions::foldername
- PestoOptions::MAP_index
- PestoOptions::mode
- PestoOptions::obj_type
- PestoOptions::options_getNextPoint .guess .min .max .update .mode
- PestoOptions::plot_options
- PestoOptions::property_index
- PestoOptions::R_min
- PestoOptions::save

History

- 2012/03/02 Jan Hasenauer
- 2016/04/10 Daniel Weindl
- 2016/10/12 Paul Stapor

Parameters

properties	property struct
parameters	parameter struct
objective_function	objective function to be optimized. This function should accept one input, the parameter vector.
varargin	`getPropertyProfiles ( ..., options )` *Required Parameters for varargin:* • options A PestoOptions object holding various options for the algorithm.

Return values

properties	updated property struct
fh	figure handle

Required fields of properties:

- `number --` Number of properties
- `min --` Lower bound for each properties
- `max --` upper bound for each properties name = {`name1`, ...}: names of the properties function = {`function1`, ...}: functions to evaluate property values. These functions provide the values of the respective properties and the corresponding 1st and 2nd order derivatives.

Required fields of parameters:

- `number` -- Number of parameters
- `min` -- Lower bound for each parameter
- `max` -- upper bound for each parameter name = {`name1`, ...}: names of the parameters
- `MS` -- results of global optimization, obtained using for instance the routine `getMultiStarts.m`. MS has to contain at least
  - par: sorted list n_theta x n_starts of parameter estimates. The first entry is assumed to be the best one.
  - logPost: sorted list n_starts x 1 of of log-posterior values corresponding to the parameters listed in .par.
  - hessian: Hessian matrix (or approximation) at the optimal point

Generated fields of properties:

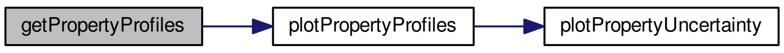
- `P(i)` -- profile for i-th parameter
  - prop: MAPs along profile
  - par: MAPs along profile
  - logPost: maximum log-posterior along profile
  - R: ratio

Definition at line 17 of file getPropertyProfiles.m.

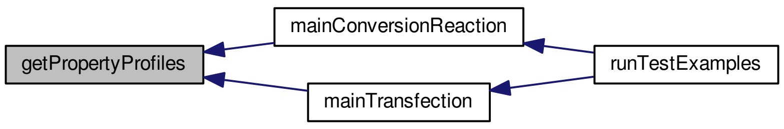
References plotPropertyProfiles().

Referenced by mainConversionReaction(), and mainTransfection().

Here is the call graph for this function:



Here is the caller graph for this function:



13.44 getPropertySamples.m File Reference

getPropertySamples.m evaluates the properties for the sampled parameters.

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, properties >,mlhsInnerSubst< matlabtypesubstitute, fh > > getPropertySamples (matlabtypesubstitute properties, matlabtypesubstitute parameters, matlabtype-substitute varargin)

  getPropertySamples.m evaluates the properties for the sampled parameters.

13.44.1 Detailed Description

getPropertySamples.m evaluates the properties for the sampled parameters.

13.44.2 Function Documentation

13.44.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, properties >,mlhsInnerSubst< matlabtypesubstitute, fh > > getPropertySamples (matlabtypesubstitute *properties*, matlabtypesubstitute *parameters*, matlabtypesubstitute *varargin*)

getPropertySamples.m evaluates the properties for the sampled parameters.

USAGE

[...] = getPropertySamples(properties,parameters) [...] = getPropertySamples(properties,parameters,options) [parameters,fh] = getPropertySamples(...)

getPropertySamples() uses the following PestoOptions members

- PestoOptions::property_index
- PestoOptions::mode
- PestoOptions::fh
- PestoOptions::save
- PestoOptions::foldername
- PestoOptions::comp_type
- PestoOptions::plot_options
- PestoOptions::MCMC.thinning

History

- 2015/04/01 Jan Hasenauer
- 2016/10/04 Daniel Weindl

Parameters

properties	property struct
parameters	parameter struct
varargin	`getPropertySamples ( ..., options )` *Required Parameters for varargin:* • options A PestoOptions object holding various options for the algorithm.

Return values

properties	updated parameter object
fh	figure handle

Required fields of properties:

- `number --` number of parameter
- `min --` lower bound for property values
- `max --` upper bound for property values
- `name --` = {`name1`,...} ... names of the parameters
- `function --` = {`function1`,...} ... functions to evaluate property values. These functions provide the values of the respective properties and the corresponding 1st and 2nd order derivatives.

Required fields of parameters:

- `S --` parameter and posterior sample. logPost ... log-posterior function along chain par ... parameters along chain *Note* This struct is obtained using getSamples.m.

Generated fields of properties:

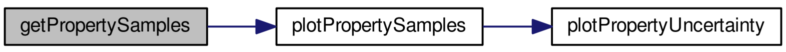
- `S --` properties for sampling results
  - par(∗,i): ith samples parameter vector
  - logPost(i): log-posterior for ith samples parameter vector
  - prop(j,i): values of jth property for ith samples parameter vector
  - prop_Sigma(∗,∗,i): covariance of properties for ith samples parameter vector

Definition at line 17 of file getPropertySamples.m.

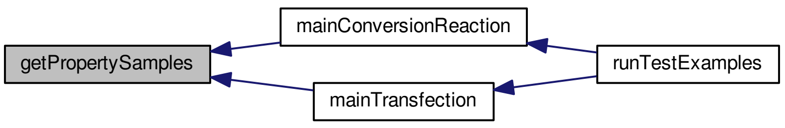
References plotPropertySamples().

Referenced by mainConversionReaction(), and mainTransfection().

Here is the call graph for this function:



Here is the caller graph for this function:



13.45 meigoDummy.m File Reference

Objective function wrapper for MEIGO / PSwarm / ... which need objective function *file$*$name* and cannot use function handles directly.

Functions

- mlhsInnerSubst< matlabtypesubstitute, f > meigoDummy (matlabtypesubstitute theta, matlabtypesubstitute fun, matlabtypesubstitute varargin)

    Objective function wrapper for MEIGO / PSwarm / ... which need objective function file$*$name and cannot use function handles directly.

13.45.1 Detailed Description

Objective function wrapper for MEIGO / PSwarm / ... which need objective function *file$*$name* and cannot use function handles directly.

13.45.2 Function Documentation

13.45.2.1 mlhsInnerSubst< matlabtypesubstitute, f > meigoDummy (matlabtypesubstitute *theta,* matlabtypesubstitute *fun,* matlabtypesubstitute *varargin*)

Objective function wrapper for MEIGO / PSwarm / ... which need objective function *file$*$name* and cannot use function handles directly.

Parameters

theta	parameter vector
fun	objective function handle
varargin	

Definition at line 17 of file meigoDummy.m.

13.46 plotConfidenceIntervals.m File Reference

plotConfidenceIntervals.m visualizes confidence itervals stored in either the parameters or properties struct .CI

Functions

- mlhsInnerSubst< matlabtypesubstitute, fh > plotConfidenceIntervals (matlabtypesubstitute pStruct, matlabtypesubstitute alpha, matlabtypesubstitute varargin)

    plotConfidenceIntervals.m visualizes confidence itervals stored in either the parameters or properties struct .CI
- mlhsInnerSubst< matlabtypesubstitute, methodsOut > **mtoc_subst_plotConfidenceIntervals_m_tsbus↩ _cotm_checkMeth** (matlabtypesubstitute methodsIn, matlabtypesubstitute pStruct, matlabtypesubstitute boolWarning)

13.46.1 Detailed Description

plotConfidenceIntervals.m visualizes confidence itervals stored in either the parameters or properties struct .CI

13.46.2 Function Documentation

13.46.2.1 mlhsInnerSubst< matlabtypesubstitute, fh > plotConfidenceIntervals (matlabtypesubstitute *pStruct*, matlabtypesubstitute *alpha*, matlabtypesubstitute *varargin*)

plotConfidenceIntervals.m visualizes confidence itervals stored in either the parameters or properties struct .CI

USAGE

fh = plotParameterUncertainty(pStruct) fh = plotParameterUncertainty(pStruct, methods) fh = plotParameter↩ Uncertainty(pStruct, methods, options)

plotMultiStarts() uses the following PestoPlottingOptions members

- PestoPlottingOptions::P
- PestoPlottingOptions::S
- PestoPlottingOptions::MS
- PestoPlottingOptions::boundary
- PestoPlottingOptions::subplot_size_1D
- PestoPlottingOptions::subplot_indexing_1D
- PestoPlottingOptions::CL
- PestoPlottingOptions::hold_on
- PestoPlottingOptions::interval
- PestoPlottingOptions::bounds
- PestoPlottingOptions::A
- PestoPlottingOptions::add_points
- PestoPlottingOptions::labels
- PestoPlottingOptions::legend
- PestoPlottingOptions::op2D
- PestoPlottingOptions::fontsize

History

- 2016/11/14 Paul Stapor

Parameters

pStruct	either the parameter or the property struct containing information about parameters and results of optimization (.MS) and uncertainty analysis (.P and .S). This structures is the output of plotMultiStarts.m, getProfiles.m or plotSamples.m.
varargin	`plotConfidenceIntervals ( ..., method, options )` *Required Parameters for varargin:* • method integer array, from which method confidence intervals should be plotted: • options options of plotting as instance of PestoPlottingOptions

Return values

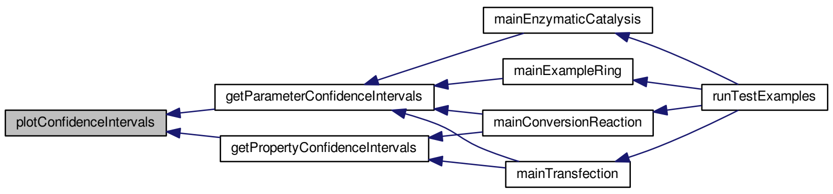
fh	figure handle

Required fields of pStruct:

Definition at line 17 of file plotConfidenceIntervals.m.

Referenced by getParameterConfidenceIntervals(), and getPropertyConfidenceIntervals().

Here is the caller graph for this function:



13.47 plotMCMCdiagnosis.m File Reference

plotMCMCdiagnosis.m visualizes the Markov chains generated by getSamples.m.

Functions

- mlhsInnerSubst< matlabtypesubstitute, fh > plotMCMCdiagnosis (matlabtypesubstitute parameters, matlabtypesubstitute varargin)

  plotMCMCdiagnosis.m visualizes the Markov chains generated by getSamples.m.

13.47.1 Detailed Description

plotMCMCdiagnosis.m visualizes the Markov chains generated by getSamples.m.

13.47.2 Function Documentation

13.47.2.1 mlhsInnerSubst< matlabtypesubstitute, fh > plotMCMCdiagnosis (matlabtypesubstitute *parameters,* matlabtypesubstitute *varargin*)

plotMCMCdiagnosis.m visualizes the Markov chains generated by getSamples.m.

USAGE

fh = plotMCMCdiagnosis(parameters) fh = plotMCMCdiagnosis(parameters,type) fh = plotMCM↩
Cdiagnosis(parameters,type,fh) fh = plotMCMCdiagnosis(parameters,type,fh,I) fh = plotMCMCdiagnosis(parameters,type,fh,↩
I,options)

History

- 2014/06/20 Jan Hasenauer
- 2016/10/10 Daniel Weindl

Parameters

parameters	parameter struct containing information about parameters and results of optimization (.MS) and uncertainty analysis (.S). This structures is the output of plotMultiStarts.m, getProfiles.m or plotSamples.m.
varargin	

Definition at line 17 of file plotMCMCdiagnosis.m.

13.48 plotMultiStarts.m File Reference

plotMultiStarts plots the result of the multi-start optimization stored in parameters.

Functions

- mlhsInnerSubst< matlabtypesubstitute, fh > plotMultiStarts (matlabtypesubstitute parameters, matlabtypesubstitute varargin)

  plotMultiStarts plots the result of the multi-start optimization stored in parameters.

13.48.1 Detailed Description

plotMultiStarts plots the result of the multi-start optimization stored in parameters.

13.48.2 Function Documentation

13.48.2.1 mlhsInnerSubst< matlabtypesubstitute, fh > plotMultiStarts (matlabtypesubstitute *parameters,* matlabtypesubstitute *varargin*)

plotMultiStarts plots the result of the multi-start optimization stored in parameters.

USAGE

fh = plotMultiStarts(parameters) fh = plotMultiStarts(parameters,fh) fh = plotMultiStarts(parameters,fh,options)

plotMultiStarts() uses the following PestoPlottingOptions members

- PestoPlottingOptions::add_points
- PestoPlottingOptions::title
- PestoPlottingOptions::draw_bounds

History:

- 2012/05/31 Jan Hasenauer
- 2016/10/07 Daniel Weindl

Parameters

parameters	parameter struct containing information about parameters and log-posterior.
varargin	`1 plotMultiStarts ( ..., fh, options )` *Required Parameters for varargin:* • fh handle of figure in which profile likelihood is plotted. If no figure handle is provided, a new figure is opened. • options options of plotting as instance of PestoPlottingOptions

Return values

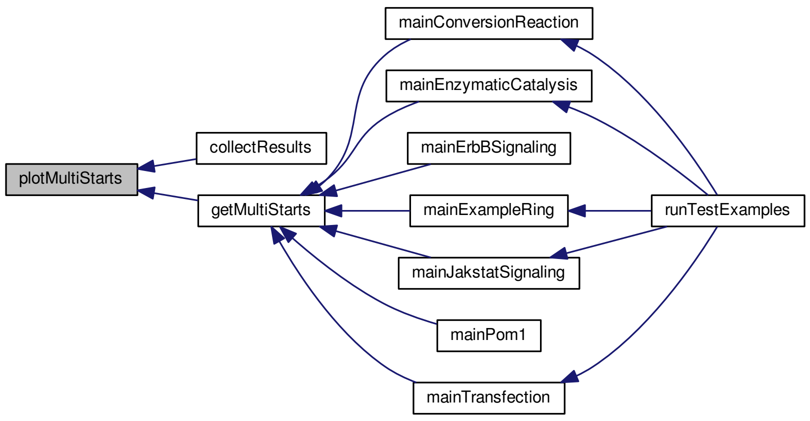
fh	figure handle

Required fields of parameters:

Definition at line 17 of file plotMultiStarts.m.

Referenced by collectResults(), and getMultiStarts().

Here is the caller graph for this function:



13.49 plotParameterProfiles.m File Reference

plotParameterProfiles.m visualizes profile likelihood. Note: This routine provides an interface for plotUncertainty.m.

Functions

- mlhsInnerSubst< matlabtypesubstitute, fh > plotParameterProfiles (matlabtypesubstitute parameters, matlabtypesubstitute varargin)

  plotParameterProfiles.m visualizes profile likelihood. Note: This routine provides an interface for plotUncertainty.m.

13.49.1 Detailed Description

plotParameterProfiles.m visualizes profile likelihood. Note: This routine provides an interface for plotUncertainty.m.

13.49.2 Function Documentation

13.49.2.1 mlhsInnerSubst< matlabtypesubstitute, fh > plotParameterProfiles (matlabtypesubstitute *parameters*, matlabtypesubstitute *varargin*)

plotParameterProfiles.m visualizes profile likelihood. Note: This routine provides an interface for plotUncertainty.m.

USAGE

fh = plotParameterProfiles(parameters) fh = plotParameterProfiles(parameters,type) fh = plotParameter↩ Profiles(parameters,type,fh) fh = plotParameterProfiles(parameters,type,fh,I) fh = plotParameterProfiles(parameters,type,fh,↩ I,options)

History

- 2012/05/31 Jan Hasenauer
- 2014/06/20 Jan Hasenauer
- 2016/10/10 Daniel Weindl

Parameters

parameters	parameter struct containing information about parameters and results of optimization (.MS) and uncertainty analysis (.P and .S). This structures is the output of plotMultiStarts.m, getProfiles.m or plotSamples.m.
varargin	`1 plotParameterProfiles ( ..., type, fh, I, options )` *Required Parameters for varargin:* • type string indicating the type of visualization: `1D` or `2D` • fh handle of figure. If no figure handle is provided, a new figure is opened. • I index of parameters which are updated. If no index is provided all parameters are updated. • options options of plotting as instance of PestoPlottingOptions

Return values

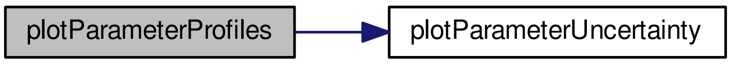
fh	figure handle

Definition at line 17 of file plotParameterProfiles.m.

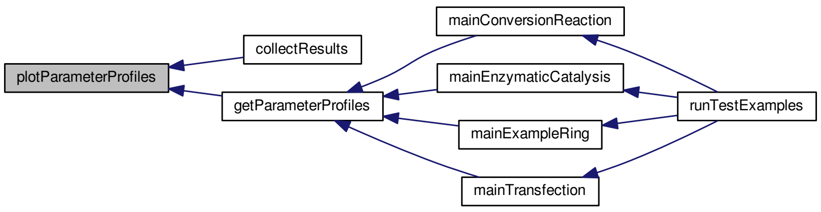
References plotParameterUncertainty().

Referenced by collectResults(), and getParameterProfiles().

Here is the call graph for this function:



Here is the caller graph for this function:



13.50 plotParameterSamples.m File Reference

plotParameterSamples.m visualizes MCMC samples. Note: This routine provides an interface for plotUncertainty.m.

Functions

- mlhsInnerSubst< matlabtypesubstitute, fh > plotParameterSamples (matlabtypesubstitute parameters, matlabtypesubstitute varargin)

  plotParameterSamples.m visualizes MCMC samples. Note: This routine provides an interface for plotUncertainty.m.

13.50.1 Detailed Description

plotParameterSamples.m visualizes MCMC samples. Note: This routine provides an interface for plotUncertainty.m.

13.50.2 Function Documentation

13.50.2.1 mlhsInnerSubst< matlabtypesubstitute, fh > plotParameterSamples (matlabtypesubstitute *parameters,* matlabtypesubstitute *varargin*)

plotParameterSamples.m visualizes MCMC samples. Note: This routine provides an interface for plotUncertainty.m.

USAGE

fh = plotParameterSamples(parameters) fh = plotParameterSamples(parameters,type) fh = plotParameter↩
Samples(parameters,type,fh) fh = plotParameterSamples(parameters,type,fh,I) fh = plotParameter↩
Samples(parameters,type,fh,I,options)

History

- 2012/05/31 Jan Hasenauer
- 2014/06/20 Jan Hasenauer
- 2016/10/10 Daniel Weindl

Parameters

parameters	parameter struct containing information about parameters and results of optimization (.MS) and uncertainty analysis (.P and .S). This structures is the output of plotMultiStarts.m, getProfiles.m or plotSamples.m.
varargin	`1 plotParameterSamples ( ..., type, fh, I, options )` *Required Parameters for varargin:* • type string indicating the type of visualization: `1D` or `2D` • fh handle of figure. If no figure handle is provided, a new figure is opened. • I index of parameters which are updated. If no index is provided all parameters are updated. • options options of plotting as instance of PestoPlottingOptions

Return values

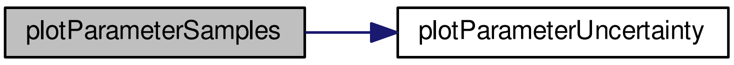
fh	figure handle

Definition at line 17 of file plotParameterSamples.m.

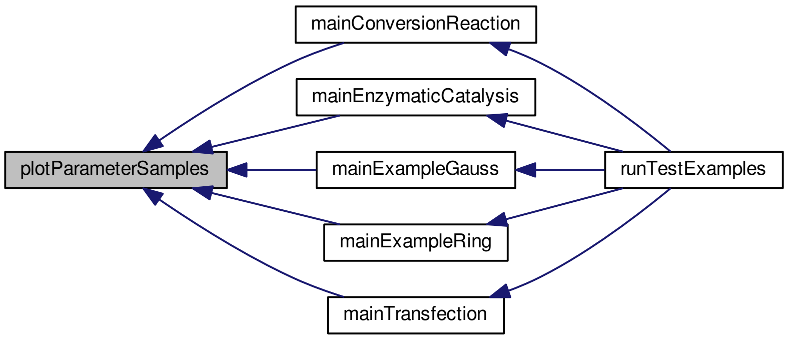
References plotParameterUncertainty().

Referenced by mainConversionReaction(), mainEnzymaticCatalysis(), mainExampleGauss(), mainExampleRing(), and mainTransfection().

Here is the call graph for this function:



Here is the caller graph for this function:



13.51 plotParameterUncertainty.m File Reference

plotParameterUncertainty.m visualizes profile likelihood and MCMC samples stored in parameters.

Functions

- mlhsInnerSubst< matlabtypesubstitute, fh > plotParameterUncertainty (matlabtypesubstitute parameters, matlabtypesubstitute varargin)

  plotParameterUncertainty.m visualizes profile likelihood and MCMC samples stored in parameters.

13.51.1 Detailed Description

plotParameterUncertainty.m visualizes profile likelihood and MCMC samples stored in parameters.

13.51.2 Function Documentation

13.51.2.1 mlhsInnerSubst< matlabtypesubstitute, fh > plotParameterUncertainty (matlabtypesubstitute *parameters,* matlabtypesubstitute *varargin*)

plotParameterUncertainty.m visualizes profile likelihood and MCMC samples stored in parameters.

USAGE

fh = plotParameterUncertainty(parameters) fh = plotParameterUncertainty(parameters,type) fh = plot↩ParameterUncertainty(parameters,type,fh) fh = plotParameterUncertainty(parameters,type,fh,I) fh = plot↩ParameterUncertainty(parameters,type,fh,I,options)

plotMultiStarts() uses the following PestoPlottingOptions members

- PestoPlottingOptions::P
- PestoPlottingOptions::S

- PestoPlottingOptions::MS
- PestoPlottingOptions::boundary
- PestoPlottingOptions::subplot_size_1D
- PestoPlottingOptions::subplot_indexing_1D
- PestoPlottingOptions::CL
- PestoPlottingOptions::hold_on
- PestoPlottingOptions::interval
- PestoPlottingOptions::bounds
- PestoPlottingOptions::A
- PestoPlottingOptions::add_points
- PestoPlottingOptions::labels
- PestoPlottingOptions::legend
- PestoPlottingOptions::op2D
- PestoPlottingOptions::fontsize

History

- 2012/05/31 Jan Hasenauer
- 2014/06/20 Jan Hasenauer
- 2016/10/10 Daniel Weindl

Parameters

parameters	parameter struct containing information about parameters and results of optimization (.MS) and uncertainty analysis (.P and .S). This structures is the output of plotMultiStarts.m, getProfiles.m or plotSamples.m.
varargin	`plotParameterUncertainty ( ..., type, fh, I, options )` *Required Parameters for varargin:* • type string indicating the type of visualization: `1D` or `2D` • fh handle of figure. If no figure handle is provided, a new figure is opened. • I index of parameters which are updated. If no index is provided all parameters are updated. • options options of plotting as instance of PestoPlottingOptions

Return values

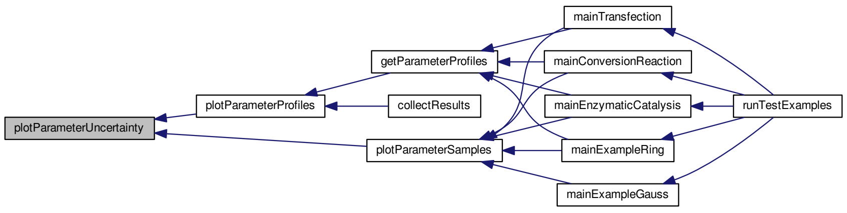
fh	figure handle

Required fields of parameters:

Definition at line 17 of file plotParameterUncertainty.m.

Referenced by plotParameterProfiles(), and plotParameterSamples().

Here is the caller graph for this function:



13.52 plotPropertyMultiStarts.m File Reference

plotPropertyMultiStarts plots the result of the multi-start optimization stored in properties.

Functions

- mlhsInnerSubst< matlabtypesubstitute, fh > plotPropertyMultiStarts (matlabtypesubstitute properties, matlabtypesubstitute varargin)

  plotPropertyMultiStarts plots the result of the multi-start optimization stored in properties.

13.52.1 Detailed Description

plotPropertyMultiStarts plots the result of the multi-start optimization stored in properties.

13.52.2 Function Documentation

13.52.2.1 mlhsInnerSubst< matlabtypesubstitute, fh > plotPropertyMultiStarts (matlabtypesubstitute *properties*, matlabtypesubstitute *varargin*)

plotPropertyMultiStarts plots the result of the multi-start optimization stored in properties.

USAGE

fh = plotPropertyMultiStarts(properties) fh = plotPropertyMultiStarts(properties,fh) fh = plotPropertyMulti↩ Starts(properties,fh,options)

History

- 2015/03/03 Jan Hasenauer

Parameters

properties	property struct containing information about properties and log-posterior.
varargin	`1 plotPropertyMultiStarts ( ..., fh, options )`
	Required Parameters for varargin: • fh handle of figure in which profile likelihood is plotted. If no figure handle is provided, a new figure is opened.

Return values

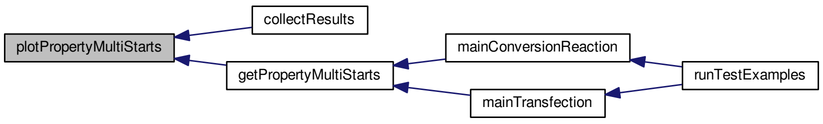
fh	figure handle

Required fields of properties:

Definition at line 17 of file plotPropertyMultiStarts.m.

Referenced by collectResults(), and getPropertyMultiStarts().

Here is the caller graph for this function:



13.53 plotPropertyProfiles.m File Reference

plotPropertyProfiles.m visualizes profile likelihood of model properties. Note: This routine provides an interface for plotPropertyUncertainty.m.

Functions

- mlhsInnerSubst< matlabtypesubstitute, fh > plotPropertyProfiles (matlabtypesubstitute properties, matlab-typesubstitute varargin)

  plotPropertyProfiles.m visualizes profile likelihood of model properties. Note: This routine provides an interface for plotPropertyUncertainty.m.

13.53.1 Detailed Description

plotPropertyProfiles.m visualizes profile likelihood of model properties. Note: This routine provides an interface for plotPropertyUncertainty.m.

13.53.2 Function Documentation

13.53.2.1 mlhsInnerSubst< matlabtypesubstitute, fh > plotPropertyProfiles (matlabtypesubstitute *properties*, matlabtypesubstitute *varargin*)

plotPropertyProfiles.m visualizes profile likelihood of model properties. Note: This routine provides an interface for plotPropertyUncertainty.m.

USAGE

fh = plotPropertyProfiles(properties) fh = plotPropertyProfiles(properties,type) fh = plotProperty↩
Profiles(properties,type,fh) fh = plotPropertyProfiles(properties,type,fh,I) fh = plotPropertyProfiles(properties,type,fh,↩
I,options)

History

- 2015/03/02 Jan Hasenauer
- 2016/10/10 Daniel Weindl

Parameters

properties	property struct containing information about properties and results of optimization (.MS) and uncertainty analysis (.P and .S).
varargin	`1 plotPropertyProfiles ( ..., type, fh, I, options )` *Required Parameters for varargin:* • type string indicating the type of visualization: `1D` or `2D` • fh handle of figure. If no figure handle is provided, a new figure is opened. • I index of properties which are updated. If no index is provided all properties are updated. • options options of plotting as instance of PestoPlottingOptions

Return values

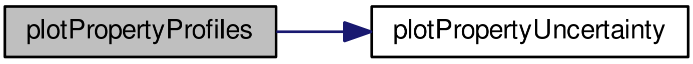
fh	figure handle

Definition at line 17 of file plotPropertyProfiles.m.

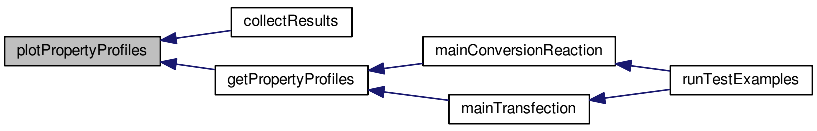
References plotPropertyUncertainty().

Referenced by collectResults(), and getPropertyProfiles().

Here is the call graph for this function:



Here is the caller graph for this function:



13.54 plotPropertySamples.m File Reference

plotPropertySamples.m visualizes samples of model properties. Note: This routine provides an interface for plot↩ PropertyUncertainty.m.

Functions

- mlhsInnerSubst< matlabtypesubstitute, fh > plotPropertySamples (matlabtypesubstitute properties, matlabtypesubstitute varargin)

  plotPropertySamples.m visualizes samples of model properties. Note: This routine provides an interface for plot↩ PropertyUncertainty.m.

13.54.1 Detailed Description

plotPropertySamples.m visualizes samples of model properties. Note: This routine provides an interface for plot↩ PropertyUncertainty.m.

13.54.2 Function Documentation

13.54.2.1 mlhsInnerSubst< matlabtypesubstitute, fh > plotPropertySamples (matlabtypesubstitute *properties,* matlabtypesubstitute *varargin*)

plotPropertySamples.m visualizes samples of model properties. Note: This routine provides an interface for plot↩ PropertyUncertainty.m.

USAGE

fh = plotPropertySamples(properties) fh = plotPropertySamples(properties,type) fh = plotProperty↩ Samples(properties,type,fh) fh = plotPropertySamples(properties,type,fh,I) fh = plotPropertySamples(properties,type,fh,↩ I,options)

History

- 2015/04/01 Jan Hasenauer
- 2016/10/10 Daniel Weindl

Parameters

properties	property struct containing information about properties and results of optimization (.MS) and uncertainty analysis (.P and .S).
varargin	`plotPropertySamples ( ..., type, fh, I, options )` *Required Parameters for varargin:* • type string indicating the type of visualization: `1D` or `2D` • fh handle of figure. If no figure handle is provided, a new figure is opened. • I index of properties which are updated. If no index is provided all properties are updated. • options options of plotting as instance of PestoPlottingOptions

Return values

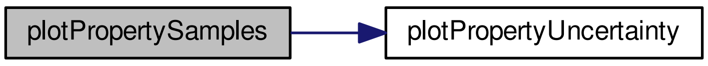
fh	figure handle

Definition at line 17 of file plotPropertySamples.m.

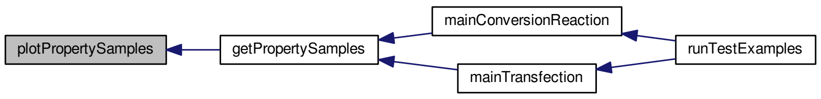
References plotPropertyUncertainty().

Referenced by getPropertySamples().

Here is the call graph for this function:



Here is the caller graph for this function:



13.55 plotPropertyUncertainty.m File Reference

plotPropertyUncertainty.m visualizes profile likelihood and MCMC samples stored in properties.

Functions

- mlhsInnerSubst< matlabtypesubstitute, fh > plotPropertyUncertainty (matlabtypesubstitute properties, matlabtypesubstitute varargin)

  plotPropertyUncertainty.m visualizes profile likelihood and MCMC samples stored in properties.

13.55.1 Detailed Description

plotPropertyUncertainty.m visualizes profile likelihood and MCMC samples stored in properties.

13.55.2 Function Documentation

13.55.2.1 mlhsInnerSubst< matlabtypesubstitute, fh > plotPropertyUncertainty (matlabtypesubstitute *properties*, matlabtypesubstitute *varargin*)

plotPropertyUncertainty.m visualizes profile likelihood and MCMC samples stored in properties.

USAGE

fh = plotPropertyUncertainty(properties,type) fh = plotPropertyUncertainty(properties,type,fh) fh = plot←PropertyUncertainty(properties,type,fh,I) fh = plotPropertyUncertainty(properties,type,fh,I,options)

History

- 2012/05/31 Jan Hasenauer
- 2014/06/20 Jan Hasenauer
- 2016/10/10 Daniel Weindl

Parameters

properties	properties struct.
varargin	`1 plotPropertyUncertainty ( ..., type, fh, I, options )` *Required Parameters for varargin:* • type string indicating the type of visualization: `1D` • fh handle of figure. If no figure handle is provided, a new figure is opened. • I index of properties which are updated. If no index is provided all parameters are updated. • options options of plotting as instance of PestoPlottingOptions

Return values

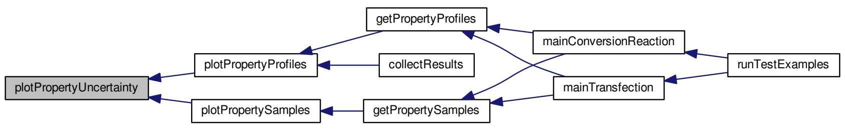
fh	figure handle

Required fields of properties:

Definition at line 17 of file plotPropertyUncertainty.m.

Referenced by plotPropertyProfiles(), and plotPropertySamples().

Here is the caller graph for this function:



13.56 runPestoTests.m File Reference

runPestoTests Run a set of PESTO unit tests

Functions

- noret::substitute runPestoTests ()

  runPestoTests Run a set of PESTO unit tests

13.56.1 Detailed Description

runPestoTests Run a set of PESTO unit tests

13.57 testGradient.m File Reference

testGradient.m calculates finite difference approximations to the gradient to check an analytical version.

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, g >,mlhsInnerSubst< matlabtypesubstitute, g_fd_↩ f >,mlhsInnerSubst< matlabtypesubstitute, g_fd_b >,mlhsInnerSubst< matlabtypesubstitute, g_fd_c > > testGradient (matlabtypesubstitute varargin)

  testGradient.m calculates finite difference approximations to the gradient to check an analytical version.
- noret::substitute **mtoc_subst_testGradient_m_tsbus_cotm_error_plot** (matlabtypesubstitute g1, matlabtypesubstitute g2, matlabtypesubstitute ee)
- noret::substitute **mtoc_subst_testGradient_m_tsbus_cotm_ratio_plot** (matlabtypesubstitute g1, matlabtypesubstitute g2, matlabtypesubstitute rr, matlabtypesubstitute ee)

13.57.1 Detailed Description

testGradient.m calculates finite difference approximations to the gradient to check an analytical version.

13.57.2 Function Documentation

13.57.2.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute, g >,mlhsInnerSubst< matlabtypesubstitute, g_fd_f >,mlhsInnerSubst< matlabtypesubstitute, g_fd_b >,mlhsInnerSubst< matlabtypesubstitute, g_fd_c > > testGradient (matlabtypesubstitute *varargin*)

testGradient.m calculates finite difference approximations to the gradient to check an analytical version.

backward differences: g_fd_f = (f(theta+eps∗e_i) - f(theta))/eps
forward differences: g_fd_b = (f(theta) - f(theta-eps∗e_i))/eps
central differences: g_fd_c = (f(theta+eps∗e_i) - f(theta-eps∗e_i))/(2∗eps)
in order to work with tensors of order n the gradient must be returned as tensor of order n+1 where the n+1th tensor dimension indexes the parameters with respect to which the differentiation was carried out

USAGE

[...] = testGradient(theta,fun,eps,il,ig) [g,g_fd_f,g_fd_b,g_fd_c] = testGradient(...)

History

- 2014/06/11 Jan Hasenauer
- 2015/01/16 Fabian Froehlich
- 2015/04/03 Jan Hasenauer
- 2015/07/28 Fabian Froehlich

Parameters

varargin	`1 testGradient ( theta, fun, eps, il, ig )` *Required Parameters for varargin:* • theta parameter vector at which gradient is evaluated. • fun function of theta for which gradients are checked. • eps epsilon used for finite difference approximation of gradient (eps = 1e-4). • il argout index/fieldname at which function values are returned (default = 1). • ig argout index/fieldname at which gradient values are returned (default = 2).

Return values

g	gradient computed by f
g_fd↩ _f	backward differences
g_fd↩ _b	forward differences
g_fd↩ _c	central differences

Definition at line 17 of file testGradient.m.

Index