GenSSI

V2-2017-09-09

Generated by Doxygen 1.8.11

Contents

1 GenSSI 2.0 General Documentation

1.1 Introduction

GenSSI is a Matlab implementation of generating series for structural identifiability as defined in

- Chiş, O.-T., Banga, J.R. and Balsa-Canto, E. (2011) Structural Identifiability of Systems Biology Models: A Critical Comparison of Methods, PLoS ONE, 6, e27755.
- Chiş, O., Banga, J.R. and Balsa-Canto, E. (2011) GenSSI: a software toolbox for structural identifiability analysis of biological models, Bioinformatics, 27, 2610-2611.

With GenSSI, the user can specify differential equation models in terms of symbolic variables in Matlab and then analyze the models to determine which parameters are globally or locally identifiable. In addition, there are some utilities for importing models from SBML, or converting models to polynomial form and for creating multi-experiment models.

1.2 Availability

The sources for GenSSI are accessible as

- Source `tarball`
- Source `zipball`
- Git repository on `github`

Once you've obtained your copy check out the Installation

1.2.1 Obtaining GenSSI via the Git versioning system

In order to always stay up to date with the latest GenSSI versions, simply pull it from our Git repository and recompile it when a new release is available. For more information about Git checkout their `website`

The Git repository can currently be found at `https://github.com/genssi-developer/GenSSI` and a direct clone is possible via

```
git clone https://github.com/genssi-developer/GenSSI.git GenSSI
```

1.2.2 License Conditions

This software is available under the `BSD license`

Copyright (c) 2016, Oana-Teodora Chiş, Julio R. Banga, Eva Balsa-Canto, Thomas S. Ligon, Fabian Fröhlich and Jan Hasenauer. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.3 Installation

If GenSSI was downloaded as a zip, it needs to be unpacked in a convenient directory. If GenSSI was obtained via cloning of the git repository, no further unpacking is necessary.

Models are generally stored in

```
GenSSI/Examples
```

but GenSSI should be able to find them in any directory that is is the Matlab path.

When a model is analyzed, GenSSI stores the results in

```
GenSSI/Results
```

To use GenSSI, start Matlab and add the GenSSI direcory to the Matlab path. To add all toolbox directories to the Matlab path, execute the Matlab script

```
genssiStartup.m
```

To use the SBML import, libSBML (`http://sbml.org/Software/libSBML/Downloading_libSBML#MATLAB`) has to be downloaded and the directory has to be included in the MATLAB path. To store the installation for further Matlab session, the path can be saved via

```
savepath
```

2 Model Definition & Simulation

In the following we will give a detailed overview how to specify models in GenSSI and how to call the code for analyzing the model. We use the Goodwin oscillator as an example.

2.1 Model Definition

This manual will guide the user to specify models in Matlab. For example implementations, see the models in the example directory.

2.1.1 Header

The model definition needs to be defined as a function which returns a struct with all symbolic definitions and options.

```
function [model] = Goodwin()
```

2.1.2 States

Create the respective symbolic variables. The name of the symbolic variable can be chosen arbitrarily.

```
syms x1 x2 x3
```

Create the state vector containing all states:

```
model.sym.x = [x1 x2 x3];
```

2.1.3 Parameters

Create the respective symbolic variables. The name of the symbolic variable can be chosen arbitrarily.

```
syms p1 p2 p3 p4 p5 p6 p7 p8
```

Create the parameter vector:

```
model.sym.p = [p1 p2 p3 p4 p5 p6 p7 p8]
```

2.1.4 Equations

Define the equations of the model.

```
A1 = -p4*x1+p1/(p2+x3^p3);
A2 = p5*x1-p6*x2;
A3 = p7*x2-p8*x3;
model.sym.xdot=[A1 A2 A3];
```

2.1.5 Controls

Define the controls.

```
g1=0;
g2=0;
g3=0;
model.sym.g=[g1 g2 Ag3];
```

Note that the matrix of controls can by empty ([]) or contain one or more controls. Each row must have the same length as the number of states (model.sym.x), and there must be one row for each control.

2.1.6 Observables

Define the observables.

```
h1 = x1;
h2 = x2;
h3 = x3;
model.sym.y = [h1 h2 h3];
```

2.1.7 Initial Conditions

Define the initial conditions.

```
model.sym.x0 = [0.3 0.9 1.3];
```

2.2 Model Analysis

The model can then be analyzed by calling genssiMain. The first parameter is the name of the model, the second parameter (Nder) is the number of derivatives to be calculated, and the third parameter (Par) is the list of parameters to be considered for identifiability. If the second parameter is absent a default number of derivatives is used. If the third parameter is absent, the full list of parameters will be used.

The fourth parameter is a struct containing options for the analysis run. If absent, defaults will be used. The options are:

```
options.verbose (default=true); maximum (verbose) information in results file
options.noRank (default=false); no rank calculation (for speed with loss)
options.closeFigure (default=true); closes figures after storing it
```

```
genssiMain('modelName',Nder,Par,options);
```

The function genssiMain will call the model function, which puts the model struct in memory. After that, it will call all other GenSSI functions required to annalyze the model.

To make this easier to find, we have also provided a simple "run" script, "runGoodwin".

```
syms p1 p2 p4 p5 p6 p7 p8;
options.closeFigure = false;
genssiMain('Goodwin',7,[p1 p2 p4 p5 p6 p7 p8],options);
```

2.3 Conversion Utilities

The GenSSI package also includes some functions for converting models from one format to another.

2.3.1 Convert from SBML Format

```
ODE = SBMLode(modelName);
ODE.writeAMICI(modelName);
```

SBMLode and writeAMICI convert an SBML model to GenSSI/AMICI format.

modelName: name of the SBML and GenSSI model (string).

Reminder: To use the SBML import, libSBML (http://sbml.org/Software/libSBML/Downloading_libSBML#MATLA
has to be downloaded and the directory has to be included in the MATLAB path.

The input SBML model should be stored in the directory

```
GenSSI/Examples
```

Note: GenSSI now uses AMICI format. This reduces duplication and gives us full functionality of the AMICI routines. Because of this, the number of derivatives and the parameters considered for identifiability have been moved from the model definition to the call to genssiMain. These parameters are more a matter of analysis than of model definition.

Note: There are limitations to this conversion. The SBML model contains a list of all parameters used by the model, but GenSSI needs a list of parameters to be considered for analysis. It may be necessary to manually edit the GenSSI model after conversion. Both GenSSI and AMICI require specification of the observables, but this is not defined in SBML. If SBMLode finds rules that are not used in the model, it will assume that they were intended as observables. In any case, it is advisable to inspect the observables and modify them as needed.

As an example for this conversion, we have chosen to use an SMBL model from the biomodels database. We begin by accessing the web site http://www.ebi.ac.uk/biomodels-main/ searching for MAPK and dowloading the file BIOMD0000000010.xml. In order to make this easier to write, we renamed the file to BIOMD10.xml, so that our SBML model is now contained in

```
BIOMD10.xml
```

To convert this SBML model to a GenSSI model with the same name, we execute the script

```
runBIOMD10;
```

The script file includes several default values. Here, the full list of parameters is contained in ODE.parameter (later model.sym.p) is copied to Par (parameter in genssiMain) and reduced in size the list of parameters Par for which identifiability analysis is performed. Using all 22 parameters is in principle possible, but does not work on a desktop computer with 16GB RAM, because the size of the Jacobian grows exponentially with number of of parameters. In addition to the parameters, we set the observables (ODE.observable and model.sym.y = [MAPK, MAPK_P, and MAPK_PP]), and the number of Lie derivatives (parameter Nder = 6 in genssiMain).

```
modelName = 'BIOMD10';
ODE = SBMLode(modelName);
% set last 3 states as observable
ODE.observable = ODE.state(6:8);
ODE.writeAMICI(modelName);
% check structural identifiability using all Vn_ parameters
Par = ODE.parameter(logical([1,0,0,0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,1,0,1,0,0]));
genssiMain('BIOMD10_syms',6,Par);
```

This model can now be analyzed using the fucntionality of GenSSI:

```
genssiMain('BIOMD10_syms',6,Par);
```

The results show that 3 of the 6 parameters are locally identifiable and the other 3 are globally identifiable.

2.3.2 Convert to Polynomial Format

```
genssiToPolynomial(modelNameIn,modelNameOut)
```

genssiToPolynomial converts a model, expressed in terms of rational expressions, to pure polynomial format. This increases the number of state variables, but can sometimes significantly reduce the computational overhead for analyzing the model.

modeNameIn: name of model to be converted (string).

modelNameOut: name of model to be created (string).

2.3.3 Create Multi-Experiment Model

```
genssiMultiExperiment(modelNameIn,mExDef,modelNameOut)
```

genssiMultiExperiment converts a GenSSI model to a new GenSSI model based on a multi-experiment definition.

modelNameIn: the name of the input model (a string).

mExDef: the name of a multi-experiment definition file (string).

modelNameOut: the name of the output model (a string).

In chemistry, it is often possible for the chemist to arbitrarily change certain parameters, such as temperature, pressure, and the concentration of specific substances. For example, in a continuous-flow stirred tank reactor (CFSTR), the feed flow provides a constant feed of substances, at a rate that can be chosen as needed. This situation has led to the concept of "controls", which are also used in identifiability analysis. From the point of theory, the controls are variables that can be changed at will, so they have a very strong positive influence on the identifiability of a model.

In contrast, in biology, certain parameters and feed rates may be varied, but most often not arbitrarily. Often, these parameters can be varied discretely by creating a new experiment with new substances or substance concentrations. Now, in identifiability, we would like to analyze multiple experiments in one model. The result is a model for which the identifiability lies somewhere between the models without controls and those with controls.

Now we will explain the parameters used by the multi-experiment model creation on the basis of a specific example.

We begin with a model for mRNA, including translation and degradation. The model definition is:

```
function model = M1_1_U2()
    syms m G d b kTL m0
    model.sym.x=[m G];
    A1=-d*m;
    A2=0;
    model.sym.xdot=[A1 A2];
    model.sym.g=[0,kTL*m;...
                 0,-b*G];
    h1=G;
    model.sym.y=[h1];
    model.sym.x0=[m0 0];
    model.sym.p=[d b kTL m0];
end
```

The states (model.sym.x) are m (mRNA) and G (GFP, green fluorescent protein), of which only GFP is observed (model.sym.y). The differential equations (model.sym.xdot) define degradation of mRNA (-d∗m), degradation of GFP (-b∗G) and translation of mRNA to GFP (kTL∗m). Note that translation and GFP degradation have been defined as controls (model.sym.g).

Based on this model, we define a total of 4 experiments, in the experiment definition function. The expriments are defined by modifying the controls and the initial conditions. The expriments are:

1) original configuration

2) change in transfection (m0) via x0: The initial concentration of mRNA is changed.

3) change in translation via control u1 = uInh: The rate of translation is changed, for example by treating the cell with an antibiotic.

4) change in GFP degradation via control u2 = uDeg: The rate of GFP degradation i changed, for example by using destabilized GFP (d2eGFP) instead of normal GFP (eGFP).

The experiment definition is:

```
function multiExp = M1_1_eDef4()
    syms d b kTL m0Exp1
    multiExp.sym.Nexp=4;
    multiExp.sym.g = [1,1;...
                      1,1;...
                      .5,1;...
                      1,.75];
    multiExp.sym.x0 = [m0Exp1,0,...
                       0.5*m0Exp1,0,...
                       m0Exp1,0,...
                       m0Exp1,0];
    multiExp.sym.p=[d,b,kTL,m0Exp1];
end
```

The number of expriments is coded in multiExp.Nexp=4.

The variable multiExp.sym.g defines the changes in the controls and the variable multiExp.sym.x0 defines the changes in the initial conditions. Both of these variables contain one row for each experiment and one column for each state variable. In the first experiment (original configuration), the controls are 1 and the initial mRNA concentration is m0Exp1. In the second expriment (change in transfection), the initial concentration of mRNA is changed. In the third experiment (change in translation), the rate of translation is changed by changing the value of the first control. In the fourth experiment (change in GFP degradation), the rate of GFP degradation is changed by changing the value of the second control.

The original model is converted to the multi-experiment model by means of this line of code:

```
genssiMultiExperiment('M1_1_U2','M1_1_eDef4','M1_1_ME4');
```

The result of the conversion is the following (multi-experiment) model:

```
function model = M1_1_ME4()
    syms mExp1 GExp1 mExp2 GExp2 mExp3 GExp3 mExp4 GExp4
    syms d b kTL m0Exp1
    model.sym.x = [mExp1,GExp1,mExp2,GExp2,mExp3,GExp3,mExp4,GExp4];
    model.sym.g = [];
    model.sym.p = [d,b,kTL,m0Exp1];
    model.sym.x0 = [m0Exp1,0,m0Exp1/2,0,m0Exp1,0,m0Exp1,0];
    model.sym.y = [GExp1,GExp2,GExp3,GExp4];
    model.sym.xdot = [-d*mExp1,...
                      kTL*mExp1 - GExp1*b,...
                      -d*mExp2,...
                      kTL*mExp2 - GExp2*b,...
                      -d*mExp3,...
                      (kTL*mExp3)/2 - GExp3*b,...
                      -d*mExp4,...
                      kTL*mExp4 - (3*GExp4*b)/4];
end
```

Based on the original 2 state variables and 4 experiments, we now have 2∗4=8 state variables (model.sym.x). In addition, all parameters now appear directly in the differential equations (model.sym.xdot), and there are no controls.

2.3.4 Transform Model

```
genssiTransformation(modelNameIn,transDef,modelNameOut)
```

genssiTransformation converts a GenSSI model to a new GenSSI model based on a transformation definition.

modelNameIn: the name of the input model (a string).

transDef: the name of a transformation definition file (string).

modelNameOut: the name of the output model (a string).

When we analyze equations for the purpose of determining identifiability, it is sometimes useful to make two changes. The first change is removing redundant equations, which can reduce the number of state variables and the number of parameters. The second change is rescaling the variables, which can reduce the number of parameters. Both of these changes are supported by genssiTransformation.

We begin with a model for mRNA, including translation and degradation. In contrast with the simpler model used for the multi-experiment conversion (above), this model involves mRNA degradation via the action of an enzyme. The model definition is:

```
function model = M2_1_Y1()
    syms m G E1 mE d1 d2 d3 b kTL m0 E0
    model.sym.x=[m G E1 mE];
    A1=-d1*m-d2*m*E1;
    A2=+kTL*m-b*G;
    A3=+d3*mE-d2*m*E1;
    A4=-d3*mE+d2*m*E1;
    model.sym.xdot=[A1 A2 A3 A4];
    model.sym.g=[];
    h1=G;
    model.sym.y=[h1];
    model.sym.x0=[m0 0 E0 0];
    model.sym.p=[d1 d2 d3 b kTL m0 E0];
end
```

The state variables in this model (model.sym.x) are mRNA (m), GFP (G), enzyme (E1), and the mRNA-enzyme complex (mE). We have used E1 as the name of the enzyme instead of E since there are cases where Matlab will treat the symbolic variable E as e or exp(1). The differential equations show mRNA (m) decreasing due to degradation (-d1∗m) and decreasing due to complexation (-d2∗m∗E1). GFP (G) increases due to translation (kTL∗m) and decreases due to complexation (-d2∗m∗E1). The enzyme (E1) decreases due to complexation (-d2∗m∗E1) and increases due to decomplexation (d3∗mE). The change in the complex (mE) is exactly the opposite of the change in the enzyme. As a result of this, we know that E1-E1(0)=-(mE-mE(0)) or, since mE(0)=0, E1-E0+mE=0. In addition, we know that the concentration of GFP always depends on the product of mRNA(0)∗kTL, so we will be able to reduce the number of parameters by rescaling (dividing by m0).

With these observations, we can create our transformation definition:

```
function transDef = M2_1_tDef()
    syms m G E1 mE m0 E0
    syms mdm0 E1dm0
    syms d1 d2 d3 b kTL
    syms d2tm0 kTLtm0 E0dm0
    transDef.sym.Transformation = [m/m0;G;E1/m0];
    transDef.sym.Constraint = [E1-E0+mE];
    transDef.sym.p = [d1,d2tm0,d3,b,kTLtm0,E0dm0];
    syms mnew Gnew E1new
    transDef.sym.stateSubs = [mdm0,mnew;...
                              G,Gnew;...
                              E1dm0,E1new];
    transDef.sym.parSubs = [d2*m0,d2tm0;...
                            kTL*m0,kTLtm0;...
                            E0/m0,E0dm0];
end
```

The transformation is defined by two variables. The first, transDef.sym.Transformation, defines the rescaling. It contains one entry for each element of the final state vector. Since we are converting a model with 4 state variables to 3, this contains 3 elements. The second part of the definition is the constraint, transDef.sym.Constraint. This will be equated to zero during the transformation, so it is E1-E0+mE=0, or, equivalently, E1-E1(0)=-(mE-mE(0)). The variable transDef.sym.Constraint can contain multiple constraints, separated by a semicolon. Finally, there are two optional definitions of substitutions. They contain a variable number of rows, each of which is a substitution, or change of names. During the transformation process, new variables are created for the state vector and the parameters, and the names are changed via the rules "∗"->"t" (for "times") and "/"->"d" (for "divided by"). These names can be considered as suggestions for the new names, and the user can override them with the stateSubs and parSubs definitions. For example, mdm0 is replaced by mnew in the state vector, and d2∗m0 is replaced by d2tm0 in the parameters. It is considered good practice to leave these 2 definitions out the first time the transformation is run, and then add them in later in order to gain more control over the naming.

This transformation is started by running the following line of code:

```
genssiTransformation('M2_1_Y1','M2_1_tDef','M2_2');
```

The result of the transformation is the following new model:

```
function model = M2_2()
    syms mnew Gnew E1new
    syms d1 d2tm0 d3 b kTLtm0 E0dm0
    model.sym.x = [mnew,Gnew,E1new];
    model.sym.g = [];
    model.sym.p = [d1,d2tm0,d3,b,kTLtm0,E0dm0];
    model.sym.x0 = [1,0,E0dm0];
    model.sym.y = [Gnew];
    model.sym.xdot = [- d1*mnew - E1new*d2tm0*mnew,...
                      kTLtm0*mnew - Gnew*b,...
                      (E0*d3)/m0 - E1new*d2tm0*mnew - E1new*d3];
end
```

In this transformed model, the new state variables are mnew, Gnew, and E1new, based on the definition transDef.stateSubs. We could just as well have left the names of mdm0, G, and E1dm0, or used the simpler names of m, G, and E1 for the new state vector. A similar remark is valid for the parameter names. The resulting differential equations (model.sym.xdot) are less readable than in the original model, but we have eliminated one state variable and one parameter.

3 Code Organization

In the following we will briefly outline how the GenSSI code is organized. For a more detailed description we refer the reader to the documentation of the individual functions.

3.1 Directory Structure

The main, or root, directory, which we refer to as GenSSI, contains most of the GenSSI functions. In addition, the following subdirectories are used:

- GenSSI/Auxiliary contains some auxiliary functions, such as genssiRemoveZeroRows.
- GenSSI/Examples contains GenSSI model definitions.
- GenSSI/Results contains the results of analysis.
- GenSSI/Docu contains tools for creating the GenSSI documentation, as well as input and output of that process.
- GenSSI/Docu/config contains configuration files for the documentation tools.
- GenSSI/Docu/input contains input for document creation, including .dox files.
- GenSSI/Docu/output contains the output of document creation.

3.2 Document Creation

New versions of the documentation are created with the help of:

- MatlabDocMaker.m (in GenSSI/Docu)
- mtoc++ (needs to be installed and available via the path variable)
- Doxygen (needs to be installed and available via the path variable)
- LaTex (needs to be installed and available via the path variable)
- Graphviz (needs to be installed and available via the path variable)
- Ghostscript (needs to be installed and available via the path variable)

The documentation configuration is changed by editing the files in the GenSSI/Docu/config directory and by running

```
MatlabDocMaker.setup
```

A new version of the documentation is created by calling

```
MatlabDocMaker.create('latex',true)
```

This results in an html version of the guide (index.html and many other files in GenSSI/Docu/output), and a pdf version (refman.pdf in GenSSI/Docu/output/latex).

4 Hierarchical Index

4.1 Class Hierarchy

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

5 Class Index

5.1 Class List

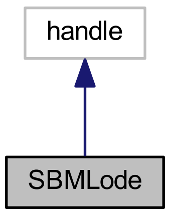
Here are the classes, structs, unions and interfaces with brief descriptions:

6 Class Documentation

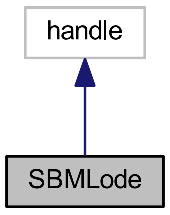
6.1 SBMLode Class Reference

SBMLMODEL provides an intermediate container between the SBML definition and an amimodel object.

Inheritance diagram for SBMLode:



Collaboration diagram for SBMLode:



Public Member Functions

- SBMLode (matlabtypesubstitute filename)

  SBMLode extracts information from an SBML definition and stores it in a symbolic format.
- noret::substitute importSBML (matlabtypesubstitute filename)

  importSBML parses information from the SBML definition and populates the SBMLode object from this information.
- noret::substitute checkODE ()

  checkODE checks whether the length of various variable names exceeds namelengthmax (would cause troube with symbolic processing later on).
- noret::substitute writeAMICI (matlabtypesubstitute modelname)

  writeAMICI writes the symbolic information from an SBMLode object into an AMICI model definition file

Public Attributes

- matlabtypesubstitute state = sym.empty("")
  states
- matlabtypesubstitute observable = sym.empty("")
  observables
- matlabtypesubstitute observable_name = sym.empty("")
  names of observables
- matlabtypesubstitute param = sym.empty("")
  parameter names
- matlabtypesubstitute parameter = sym.empty("")
  parameter expressions
- matlabtypesubstitute constant = sym.empty("")
  constants
- matlabtypesubstitute reaction = sym.empty("")
  reactions
- matlabtypesubstitute compartment = sym.empty("")
  compartments
- matlabtypesubstitute volume = sym.empty("")
  compartment volumes
- matlabtypesubstitute kvolume = sym.empty("")
  condition volumes
- matlabtypesubstitute initState = sym.empty("")
  initial condition of states
- matlabtypesubstitute condition = sym.empty("")
  condition
- matlabtypesubstitute flux = sym.empty("")
  reaction fluxes
- matlabtypesubstitute stochiometry = sym.empty("")
  reaction stochiometry
- matlabtypesubstitute xdot = sym.empty("")
  right hand side of reconstructed differential equation
- matlabtypesubstitute trigger = sym.empty("")
  event triggers
- matlabtypesubstitute bolus = sym.empty("")
  event boli
- matlabtypesubstitute funmath = cell.empty("")
  mathematical experessions for function
- matlabtypesubstitute funarg = cell.empty("")
  string function signature
- matlabtypesubstitute time_symbol = char.empty("")
  symbol of time
- matlabtypesubstitute pnom = double.empty("")
  nominal parameters
- matlabtypesubstitute knom = double.empty("")
  nominal conditions

6.1.1 Detailed Description

Definition at line 17 of file SBMLode.m.

6.1.2 Constructor & Destructor Documentation

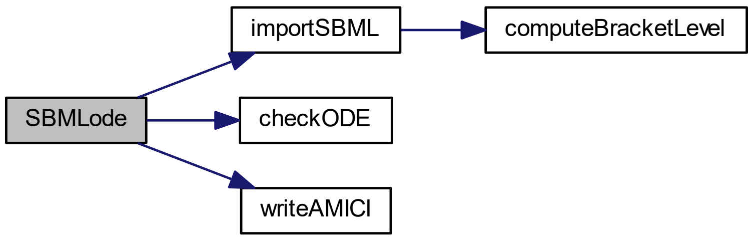
6.1.2.1 SBMLode (matlabtypesubstitute *filename*)

Parameters

filename	target name of the model (excluding the suffix .xml/.sbml)

Definition at line 207 of file SBMLode.m.

Here is the call graph for this function:



6.1.3 Member Function Documentation

6.1.3.1 noret::substitute importSBML (matlabtypesubstitute *filename*)

Parameters

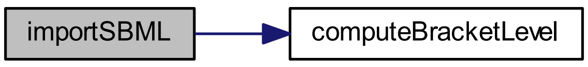
filename	target name of the model

Return values

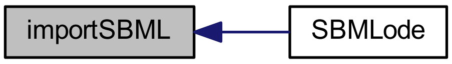
filename	void

Definition at line 18 of file importSBML.m.

Here is the call graph for this function:



Here is the caller graph for this function:



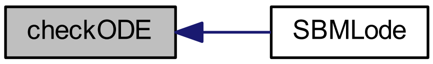
6.1.3.2 noret::substitute checkODE ( )

Return values

void	

Definition at line 18 of file checkODE.m.

Here is the caller graph for this function:



6.1.3.3 noret::substitute writeAMICI (matlabtypesubstitute *modelname*)

Parameters

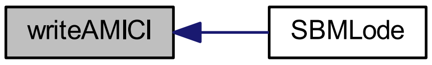
modelname	target name of the model (_syms.m will be appended to the name)

Return values

modelname	void

Definition at line 18 of file writeAMICI.m.

Here is the caller graph for this function:



6.1.4 Member Data Documentation

6.1.4.1 state = sym.empty("")

Default: sym.empty("")

Definition at line 28 of file SBMLode.m.

6.1.4.2 observable = sym.empty("")

Default: sym.empty("")

Definition at line 36 of file SBMLode.m.

6.1.4.3 observable_name = sym.empty("")

Default: sym.empty("")

Definition at line 44 of file SBMLode.m.

6.1.4.4 param = sym.empty("")

Default: sym.empty("")

Definition at line 52 of file SBMLode.m.

6.1.4.5 parameter = sym.empty("")

Default: sym.empty("")

Definition at line 60 of file SBMLode.m.

6.1.4.6 constant = sym.empty("")

Default: sym.empty("")

Definition at line 68 of file SBMLode.m.

6.1.4.7 reaction = sym.empty("")

Default: sym.empty("")

Definition at line 76 of file SBMLode.m.

6.1.4.8 compartment = sym.empty("")

Default: sym.empty("")

Definition at line 84 of file SBMLode.m.

6.1.4.9 volume = sym.empty("")

Default: sym.empty("")

Definition at line 92 of file SBMLode.m.

6.1.4.10 kvolume = sym.empty("")

Default: sym.empty("")

Definition at line 100 of file SBMLode.m.

6.1.4.11 initState = sym.empty("")

Default: sym.empty("")

Definition at line 108 of file SBMLode.m.

6.1.4.12 condition = sym.empty("")

Default: sym.empty("")

Definition at line 116 of file SBMLode.m.

6.1.4.13 flux = sym.empty("")

Default: sym.empty("")

Definition at line 124 of file SBMLode.m.

6.1.4.14 stochiometry = sym.empty("")

Default: sym.empty("")

Definition at line 132 of file SBMLode.m.

6.1.4.15 xdot = sym.empty("")

Default: sym.empty("")

Definition at line 140 of file SBMLode.m.

6.1.4.16 trigger = sym.empty("")

Default: sym.empty("")

Definition at line 148 of file SBMLode.m.

6.1.4.17 bolus = sym.empty("")

Default: sym.empty("")

Definition at line 156 of file SBMLode.m.

6.1.4.18 funmath = cell.empty("")

Default: cell.empty("")

Definition at line 164 of file SBMLode.m.

6.1.4.19 funarg = cell.empty("")

Default: cell.empty("")

Definition at line 172 of file SBMLode.m.

6.1.4.20 time_symbol = char.empty("")

Default: char.empty("")

Definition at line 180 of file SBMLode.m.

6.1.4.21 pnom = double.empty("")

Default: double.empty("")

Definition at line 188 of file SBMLode.m.

6.1.4.22 knom = double.empty("")

Default: double.empty("")

Definition at line 196 of file SBMLode.m.

7 File Documentation

7.1 Auxiliary/amiciStructToSource.m File Reference

amiciStructToSource converts a model definition (struct) to a source format (MATLAB function file) and saves the results in the examples directory.

Functions

- noret::substitute amiciStructToSource (matlabtypesubstitute model)

  amiciStructToSource converts a model definition (struct) to a source format (MATLAB function file) and saves the results in the examples directory.

7.1.1 Function Documentation

7.1.1.1 noret::substitute amiciStructToSource (matlabtypesubstitute *model*)

Parameters

model	model definition (struct)

Return values

model	void

Definition at line 17 of file amiciStructToSource.m.

7.2 Auxiliary/genssiComputeLieDerivatives.m File Reference

genssiComputeLieDerivatives computes Lie derivatives of the output functions (model.sym.y), the state vectors (model.sym.x), and the initial conditions (model.sym.x0) with respect to the equations (model.sym.xdot) and controls (model.sym.g)

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiComputeLieDerivatives (matlabtypesubstitute model, matlabtypesubstitute options)

  genssiComputeLieDerivatives computes Lie derivatives of the output functions (model.sym.y), the state vectors (model.sym.x), and the initial conditions (model.sym.x0) with respect to the equations (model.sym.xdot) and controls (model.sym.g)
- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > mtoc_subst_genssiComputeLieDerivatives_m_tsbus_cotm_jacRank (matlabtypesubstitute LDer, matlabtypesubstitute rankVector, matlabtypesubstitute order, matlabtypesubstitute model, matlabtypesubstitute options)

  jacRank computes jacobian and rank, producing output text

7.2.1 Function Documentation

7.2.1.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiComputeLieDerivatives (matlabtypesubstitute *model*, matlabtypesubstitute *options*)

Parameters

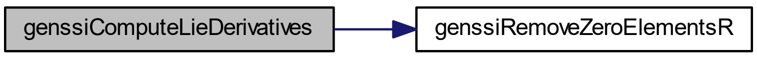
model	model definition (struct)
options	processing options (struct)

Return values

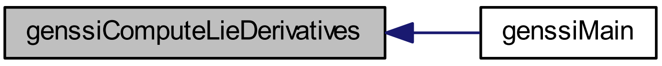
options	processing options (struct)
VectorLieDerivatives	a vector of all Lie derivatives

Definition at line 17 of file genssiComputeLieDerivatives.m.

Here is the call graph for this function:



Here is the caller graph for this function:



7.2.1.2 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > mtoc_subst_genssiComputeLieDerivatives_m_tsbus_cotm_jacRank (matlabtypesubstitute *LDer*, matlabtypesubstitute *rankVector*, matlabtypesubstitute *order*, matlabtypesubstitute *model*, matlabtypesubstitute *options*)

Parameters

LDer	model definition (struct)
rankVector	vector of ranks (for results)
order	for output text, computing derivatives of order
model	model definition
options	processing options (struct)

Return values

rankFull	boolean, true if rank is full
rankVector	vector of ranks (for results)

Definition at line 114 of file genssiComputeLieDerivatives.m.

7.3 Auxiliary/genssiComputeReducedTableau.m File Reference

genssiComputeReducedTableau computes reduced tableaus of the jacobian by eliminating rows and columns where solutions to relationships can found or excluded.

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiComputeReducedTableau (matlabtypesubstitute model, matlabtypesubstitute results, matlabtypesubstitute VectorLieDerivatives, matlabtypesubstitute JacParam, matlabtypesubstitute options)

  genssiComputeReducedTableau computes reduced tableaus of the jacobian by eliminating rows and columns where solutions to relationships can found or excluded.

7.3.1 Function Documentation

7.3.1.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiComputeReducedTableau (matlabtypesubstitute *model,* matlabtypesubstitute *results,* matlabtypesubstitute *VectorLieDerivatives,* matlabtypesubstitute *JacParam,* matlabtypesubstitute *options*)

Parameters

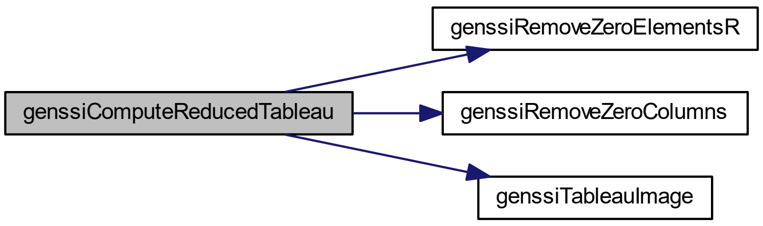
model	model definition (struct)
results	results of compute tableau (symbolic matrix)
VectorLieDerivatives	vector of Lie derivatives (symbolic array)
JacParam	jacobian with respect to parameters (symbolic matrix)
options	options (struct)

Return values

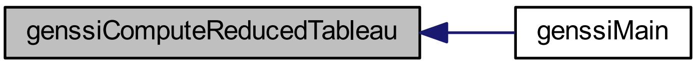
options	options (struct)
results	results of compute tableau (symbolic matrix)
RJacParam01	reduced tableau (binary matrix)
ECC	equations (symbolic matrix)
rParam	reduced list of parameters (symbolic array)

Definition at line 17 of file genssiComputeReducedTableau.m.

Here is the call graph for this function:



Here is the caller graph for this function:



7.4 Auxiliary/genssiComputeTableau.m File Reference

genssiComputeTableau computes the tableau based on the jacobian of the Lie derivatives.

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiComputeTableau (matlabtypesubstitute model, matlabtypesubstitute VectorLieDerivatives, matlabtypesubstitute options)

  genssiComputeTableau computes the tableau based on the jacobian of the Lie derivatives.

7.4.1 Function Documentation

7.4.1.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiComputeTableau (matlabtypesubstitute *model,* matlabtypesubstitute *VectorLieDerivatives,* matlabtypesubstitute *options*)

Parameters

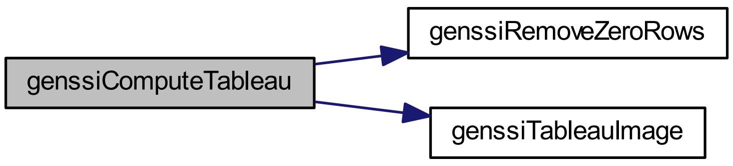
model	model definition (struct)
VectorLieDerivatives	vector of Lie derivatives (symbolic array)
options	options (struct)

Return values

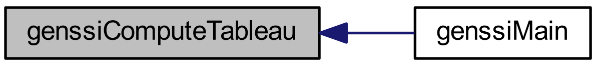
options	options (struct)
results	results of calculations (struct)
JacParam	jacobian of the Lie derivatives with respect to the parameters (symbolic matrix)

Definition at line 17 of file genssiComputeTableau.m.

Here is the call graph for this function:



Here is the caller graph for this function:



7.5 Auxiliary/genssiOrderTableau.m File Reference

genssiOrderTableau orders tableaus, searches for new opportunities to eliminate rows or columns be solving equations, and creates new (reduced) tableaus.

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiOrderTableau (matlabtypesubstitute model, matlabtypesubstitute results, matlabtypesubstitute RJacParam01, matlabtypesubstitute ECC, matlabtypesubstitute rParam, matlabtypesubstitute options)

  genssiOrderTableau orders tableaus, searches for new opportunities to eliminate rows or columns be solving equations, and creates new (reduced) tableaus.
- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > mtoc_subst_genssiOrderTableau_m_ (matlabtypesubstitute Param, matlabtypesubstitute Param_local, matlabtypesubstitute global_ident_par, matlabtypesubstitute Mat_index, matlabtypesubstitute RJacparam_new, matlabtypesubstitute RJacParam01_nonzero_rows, matlabtypesubstitute sum_RJacParam01_nonzero_rows_t, matlabtypesubstitute ECC, matlabtypesubstitute ECC_new, matlabtypesubstitute options)

displayRelevantParameters displays relevant parameters

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > mtoc_subst_genssiOrderTableau_m_tsbus_cotm_displayReducedTableau (matlabtypesubstitute ECC_remaining, matlabtypesubstitute Param_local, matlabtypesubstitute Param_display, matlabtypesubstitute global_ident_par, matlabtypesubstitute display_tableau_RJacparam_new, matlabtypesubstitute number_fig, matlabtypesubstitute options)

  displayReducedTableau displays reduced tableaus
- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > mtoc_subst_genssiOrderTableau_m_tsbus_cotm_displayRemainingParameters (matlabtypesubstitute ECC_remaining, matlabtypesubstitute Param_local, matlabtypesubstitute Param_remaining, matlabtypesubstitute global_ident_par, matlabtypesubstitute display_tableau_RJacparam_new, matlabtypesubstitute row_index_1, matlabtypesubstitute tableau_for_second_reduced_tableau, matlabtypesubstitute parameters_for_second_reduced_tableau, matlabtypesubstitute number_fig, matlabtypesubstitute options)

  displayReducedTableau displays the remaining parameters
- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > mtoc_subst_genssiOrderTableau_m_tsbus_cotm_solveRemPar (matlabtypesubstitute ECC, matlabtypesubstitute Param, matlabtypesubstitute Param_local, matlabtypesubstitute global_ident_par)

  solveRemPar solves the remaining parameters
- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > mtoc_subst_genssiOrderTableau_m_tsbus_cotm_getIndexOfDuplicateParams (matlabtypesubstitute ECC, matlabtypesubstitute RJacParam01_nonzero_rows)

  getIndexOfDuplicateParams gets index of duplicate parameters

7.5.1 Function Documentation

7.5.1.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiOrderTableau (matlabtypesubstitute *model*, matlabtypesubstitute *results*, matlabtypesubstitute *RJacParam01*, matlabtypesubstitute *ECC*, matlabtypesubstitute *rParam*, matlabtypesubstitute *options*)

Parameters

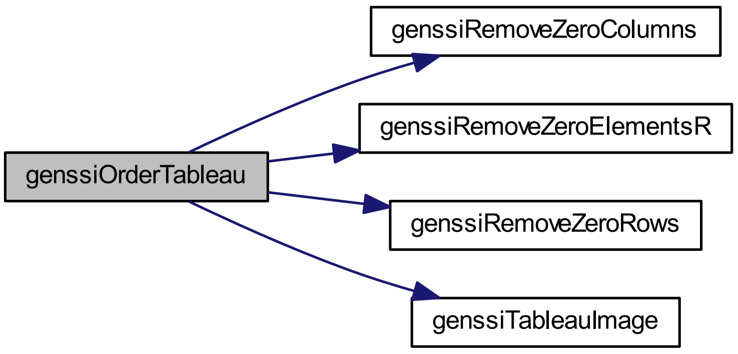
model	model definition (struct)
results	results of previous steps (struct)
RJacParam01	reduced tableau, i.e. binary form of jacobian of the Lie derivatives with respect to the parameters (binary matrix)
ECC	equations (symbolic array)
rParam	reduced list of parameters (symbolic array)
options	options (struct)

Return values

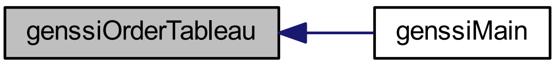
options	options (struct)
results	results of previous steps (struct)

Definition at line 17 of file genssiOrderTableau.m.

Here is the call graph for this function:



Here is the caller graph for this function:



7.6 Auxiliary/genssiPolySys.m File Reference

genssiPolySys converts a model to polynomial form. [X,DX,X0} = genssiPolySys(X,DX,X0)

Functions

- mlhsInnerSubst< matlabtypesubstitute > genssiPolySys (matlabtypesubstitute varargin)
  genssiPolySys converts a model to polynomial form. [X,DX,X0} = genssiPolySys(X,DX,X0)

7.6.1 Function Documentation

7.6.1.1 mlhsInnerSubst< matlabtypesubstitute > genssiPolySys (matlabtypesubstitute *varargin*)

Parameters

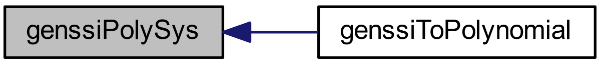
varargin	generic input arguments `1 genssiPolySys ( X, DX, X0 )` *Required Parameters for varargin:* • X the state variables of the input model (a vector) • DX the ODEs of the input model (a vector) • X0 the initial conditions of the input model (a vector)

Return values

varargout	generic output arguments
X	the state variables of the output model (a vector)
DX	the ODEs of the output model (a vector)
X0	the initial conditions of the output model (a vector)

Definition at line 17 of file genssiPolySys.m.

Here is the caller graph for this function:



7.7 Auxiliary/genssiRemoveZeroColumns.m File Reference

genssiRemoveZeroColumns removes zero columns from a matrix

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiRemoveZeroColumns (matlabtypesubstitute matrixIn)

    genssiRemoveZeroColumns removes zero columns from a matrix

7.7.1 Function Documentation

7.7.1.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiRemoveZeroColumns (matlabtypesubstitute *matrixIn*)

Parameters

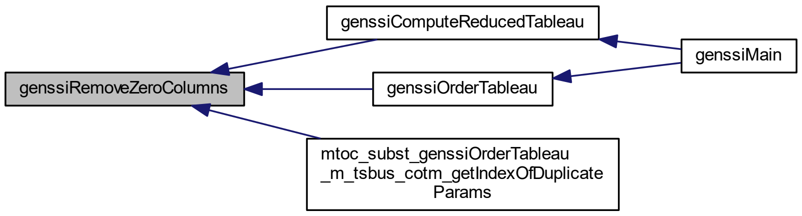
matrixIn	input (matrix)

Return values

matrixOut	output (matrix)
keepBoolean	boolean vector of indices kept (array)
keepIndex	vector of indices keept (array)

Definition at line 17 of file genssiRemoveZeroColumns.m.

Here is the caller graph for this function:


genssiComputeReducedTableau
genssiMain
genssiRemoveZeroColumns
genssiOrderTableau
mtoc_subst_genssiOrderTableau _m_tsbus_cotm_getIndexOfDuplicate Params

7.8 Auxiliary/genssiRemoveZeroElementsC.m File Reference

genssiRemoveZeroElements removes zero columns from a row vecor

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiRemoveZeroElementsC (matlabtypesubstitute vectorIn)

  genssiRemoveZeroElements removes zero columns from a row vecor

7.8.1 Function Documentation

7.8.1.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiRemoveZeroElementsC (matlabtypesubstitute *vectorIn*)

Parameters

vectorIn	input (array)

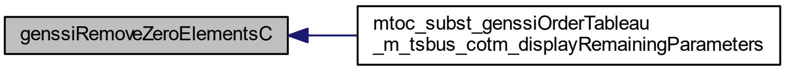
Return values

Return values

vectorOut	output (array)
keepBoolean	boolean vector of indices kept (array)
keepIndex	vector of indices kept (array)

Definition at line 17 of file genssiRemoveZeroElementsC.m.

Here is the caller graph for this function:


genssiRemoveZeroElementsC
mtoc_subst_genssiOrderTableau
_m_tsbus_cotm_displayRemainingParameters

7.9 Auxiliary/genssiRemoveZeroElementsR.m File Reference

genssiRemoveZeroElements removes zero columns from a row vecor

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiRemoveZeroElementsR (matlabtypesubstitute vectorIn)

    genssiRemoveZeroElements removes zero columns from a row vecor

7.9.1 Function Documentation

7.9.1.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiRemoveZeroElementsR (matlabtypesubstitute *vectorIn*)

Parameters

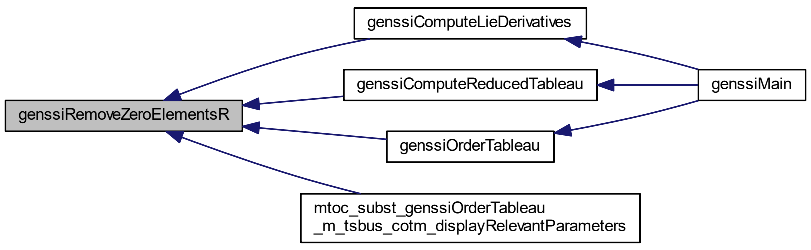
vectorIn	input (array)

Return values

vectorOut	output (array)
keepBoolean	boolean vector of indices kept (array)
keepIndex	vector of indices kept (array)

Definition at line 17 of file genssiRemoveZeroElementsR.m.

Here is the caller graph for this function:


genssiComputeLieDerivatives
genssiComputeReducedTableau
genssiMain
genssiRemoveZeroElementsR
genssiOrderTableau
mtoc_subst_genssiOrderTableau
_m_tsbus_cotm_displayRelevantParameters

7.10 Auxiliary/genssiRemoveZeroRows.m File Reference

genssiRemoveZeroRows removes zero rows from a matrix

Functions

- mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiRemoveZeroRows (matlabtypesubstitute matrixIn)

  genssiRemoveZeroRows removes zero rows from a matrix

7.10.1 Function Documentation

7.10.1.1 mlhsSubst< mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute >,mlhsInnerSubst< matlabtypesubstitute > > genssiRemoveZeroRows (matlabtypesubstitute *matrixIn*)

Parameters

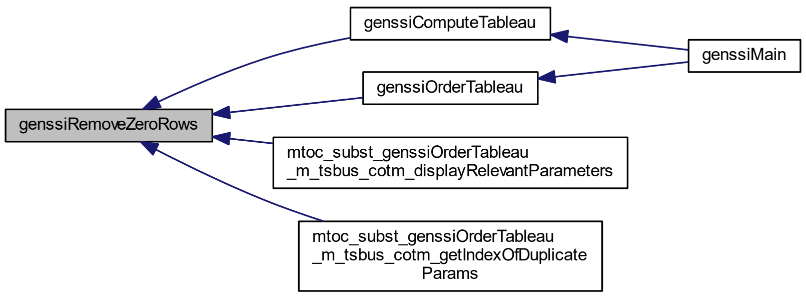
matrixIn	input (matrix)

Return values

matrixOut	output (matrix)
keepBoolean	boolean vector of indices kept (array)
keepIndex	vector of indices keept (array)

Definition at line 17 of file genssiRemoveZeroRows.m.

Here is the caller graph for this function:


genssiComputeTableau
genssiMain
genssiOrderTableau
genssiRemoveZeroRows
mtoc_subst_genssiOrderTableau_m_tsbus_cotm_displayRelevantParameters
mtoc_subst_genssiOrderTableau_m_tsbus_cotm_getIndexOfDuplicateParams

7.11 Auxiliary/genssiReportInputs.m File Reference

genssiReportInputs reports inputs, i.e. model definition.

Functions

- mlhsInnerSubst< matlabtypesubstitute > genssiReportInputs (matlabtypesubstitute model, matlabtypesubstitute options)

  genssiReportInputs reports inputs, i.e. model definition.

7.11.1 Function Documentation

7.11.1.1 mlhsInnerSubst< matlabtypesubstitute > genssiReportInputs (matlabtypesubstitute *model,* matlabtypesubstitute *options*)

Parameters

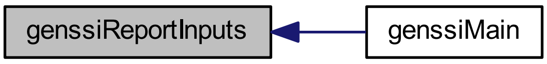
model	model definition (struct)
options	options (struct)

Return values

options	options (struct)

Definition at line 17 of file genssiReportInputs.m.

Here is the caller graph for this function:



7.12 Auxiliary/genssiReportResults.m File Reference

genssiReportResults reports the results of the analysis.

Functions

- mlhsInnerSubst$<$ matlabtypesubstitute $>$ genssiReportResults (matlabtypesubstitute model, matlabtypesubstitute results, matlabtypesubstitute options)

  genssiReportResults reports the results of the analysis.

7.12.1 Function Documentation

7.12.1.1 mlhsInnerSubst$<$ matlabtypesubstitute $>$ genssiReportResults (matlabtypesubstitute *model,* matlabtypesubstitute *results,* matlabtypesubstitute *options*)

Parameters

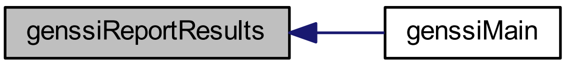
model	model definition (struct)
results	results of previous steps (struct)
options	options (struct)

Return values

options	options (struct)

Definition at line 17 of file genssiReportResults.m.

Here is the caller graph for this function:



7.13 Auxiliary/genssiStructToSource.m File Reference

genSsiStructToSource converts a model definition (struct) to a source format (Matlab function file) and saves the results in the examples directory.

Functions

- noret::substitute genssiStructToSource (matlabtypesubstitute model, matlabtypesubstitute modelName)

  genSsiStructToSource converts a model definition (struct) to a source format (Matlab function file) and saves the results in the examples directory.

7.13.1 Function Documentation

7.13.1.1 noret::substitute genssiStructToSource (matlabtypesubstitute *model,* matlabtypesubstitute *modelName*)

Parameters

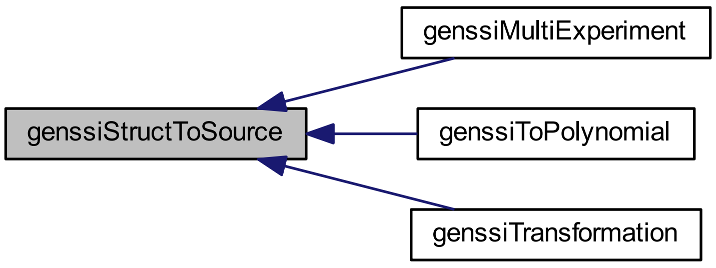
model	model definition (struct)
modelName	model name

Return values

modelName	void

Definition at line 17 of file genssiStructToSource.m.

Here is the caller graph for this function:



7.14 Auxiliary/genssiTableauImage.m File Reference

genssiTableauImage displays an identifiability tableau

Functions

- noret::substitute genssiTableauImage (matlabtypesubstitute figNum, matlabtypesubstitute tabMat, matlabtypesubstitute paramDisplay, matlabtypesubstitute options)

  genssiTableauImage displays an identifiability tableau

7.14.1 Function Documentation

7.14.1.1 noret::substitute genssiTableauImage (matlabtypesubstitute *figNum,* matlabtypesubstitute *tabMat,* matlabtypesubstitute *paramDisplay,* matlabtypesubstitute *options*)

Parameters

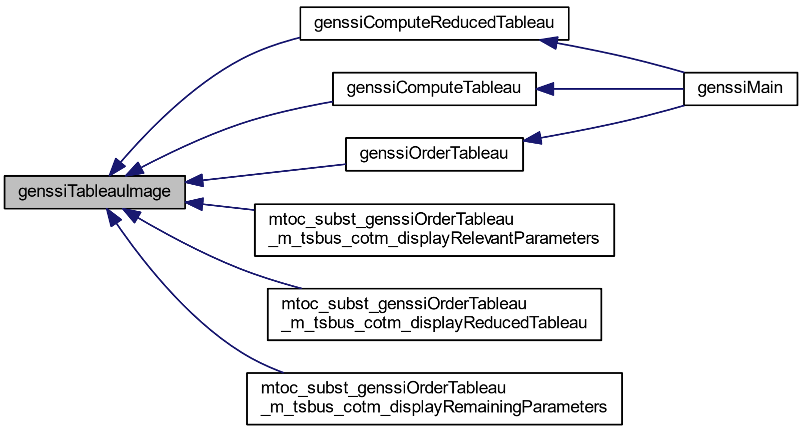
figNum	figure number
tabMat	matrix containing tableau
paramDisplay	parameter vector
options	options

Return values

options	void

Definition at line 17 of file genssiTableauImage.m.

Here is the caller graph for this function:


genssiComputeReducedTableau
genssiComputeTableau
genssiOrderTableau
genssiMain
genssiTableauImage
mtoc_subst_genssiOrderTableau _m_tsbus_cotm_displayRelevantParameters
mtoc_subst_genssiOrderTableau _m_tsbus_cotm_displayReducedTableau
mtoc_subst_genssiOrderTableau _m_tsbus_cotm_displayRemainingParameters

7.15 genssiMain.m File Reference

genssiMain is the main function of GenSSI. It reads a model and calls all other functions necessary for analyzing the model.

Functions

- mlhsInnerSubst< matlabtypesubstitute > genssiMain (matlabtypesubstitute modelName, matlabtypesubstitute Nder, matlabtypesubstitute Par, matlabtypesubstitute optionsIn)

  genssiMain is the main function of GenSSI. It reads a model and calls all other functions necessary for analyzing the model.

7.15.1 Function Documentation

7.15.1.1 mlhsInnerSubst< matlabtypesubstitute > genssiMain (matlabtypesubstitute *modelName,* matlabtypesubstitute *Nder,* matlabtypesubstitute *Par,* matlabtypesubstitute *optionsIn*)

Parameters

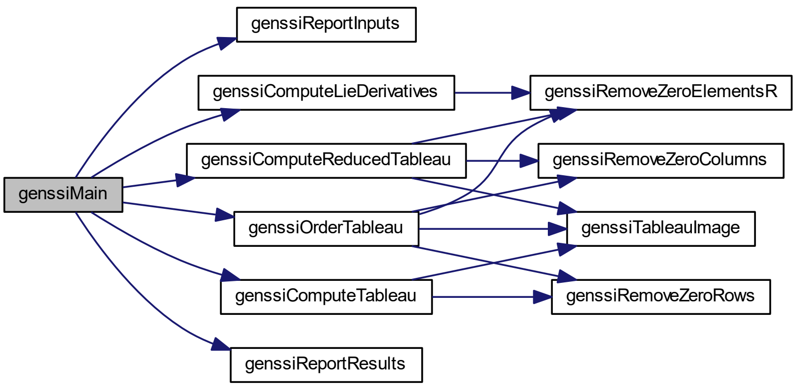
modelName	the name of the model to be analyzed (a string)
Nder	number of Lie derivatives
Par	vector of parameters to be considered
optionsIn	struct of options for analysis

Return values

options	struct containing options

Definition at line 17 of file genssiMain.m.

Here is the call graph for this function:



7.16 genssiMultiExperiment.m File Reference

genssiMultiExperiment converts a GenSSI model to a new GenSSI model based on a multi-experiment definition.

Functions

- mlhsInnerSubst< matlabtypesubstitute > genssiMultiExperiment (matlabtypesubstitute varargin)

  genssiMultiExperiment converts a GenSSI model to a new GenSSI model based on a multi-experiment definition.

7.16.1 Function Documentation

7.16.1.1 mlhsInnerSubst< matlabtypesubstitute > genssiMultiExperiment (matlabtypesubstitute *varargin*)

Parameters

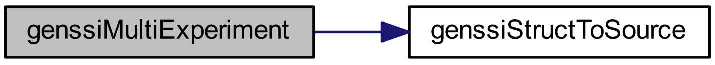
varargin	generic input arguments `genssiMultiExperiment ( modelNameIn, mExDef, modelNameOut )` *Required Parameters for varargin:* • modelNameIn the name of the input model (a string) • mExDef the name of a multi-experiment definition file (string) • modelNameOut the name of the output model (a string)

Return values

varargout	generic output arguments

Definition at line 17 of file genssiMultiExperiment.m.

Here is the call graph for this function:



7.17 genssiStartup.m File Reference

genssiStartup adds all paths required for GenSSI. It should be called at the beginning of a session.

Functions

- noret::substitute genssiStartup ()

  genssiStartup adds all paths required for GenSSI. It should be called at the beginning of a session.

7.18 genssiToPolynomial.m File Reference

genssiToPolynomial converts a GenSSI model to polynomial form. It reads the input model, converts to polynomial form, and creates an output model as a Matlab function modelNameOut.m and as a Matlab file modelnameOut.mat, both in the Examples folder.

Functions

- mlhsInnerSubst< matlabtypesubstitute > genssiToPolynomial (matlabtypesubstitute modelNameIn, matlabtypesubstitute modelNameOut)

    genssiToPolynomial converts a GenSSI model to polynomial form. It reads the input model, converts to polynomial form, and creates an output model as a Matlab function modelNameOut.m and as a Matlab file modelnameOut.mat, both in the Examples folder.

7.18.1 Function Documentation

7.18.1.1 mlhsInnerSubst< matlabtypesubstitute > genssiToPolynomial (matlabtypesubstitute *modelNameIn,* matlabtypesubstitute *modelNameOut*)

Parameters

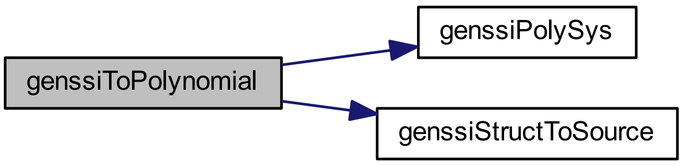
modelNameIn	the name of the input model (a string)
modelNameOut	the name of the output model (a string)

Return values

modelNameOut	void

Definition at line 17 of file genssiToPolynomial.m.

Here is the call graph for this function:



7.19 genssiTransformation.m File Reference

genssiTransformation converts a GenSSI model to a new GenSSI model based on a transformation definition.

Functions

- mlhsInnerSubst< matlabtypesubstitute > genssiTransformation (matlabtypesubstitute varargin)

  genssiTransformation converts a GenSSI model to a new GenSSI model based on a transformation definition.

7.19.1 Function Documentation

7.19.1.1 mlhsInnerSubst< matlabtypesubstitute > genssiTransformation (matlabtypesubstitute *varargin*)

Parameters

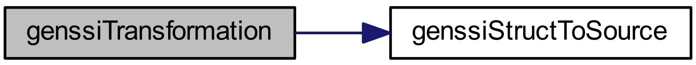
varargin	generic input arguments `1 genssiTransformation ( modelNameIn, transDef, modelNameOut )` *Required Parameters for varargin:* • modelNameIn the name of the input model (a string) • transDef the name of a transformation definition file (string) • modelNameOut the name of the output model (a string)

Return values

varargout	generic output arguments

Definition at line 17 of file genssiTransformation.m.

Here is the call graph for this function:



7.20 SBMLimporter/computeBracketLevel.m File Reference

Compute the bracket level for the input string cstr. The bracket level is computed for every char in cstr and indicates how many brackets have been opened up to this point. The bracket level is useful to parse the arguments of functions in cstr. For this purpose functions will have the same bracket level as the opening bracket of the corresponding function call.

Functions

- mlhsInnerSubst<::∗int > computeBracketLevel (::∗char cstr)

  Compute the bracket level for the input string cstr. The bracket level is computed for every char in cstr and indicates how many brackets have been opened up to this point. The bracket level is useful to parse the arguments of functions in cstr. For this purpose functions will have the same bracket level as the opening bracket of the corresponding function call.

7.20.1 Function Documentation

7.20.1.1 mlhsInnerSubst< ::*int > computeBracketLevel (::*char *cstr*)

Parameters

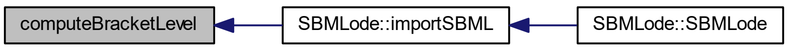
cstr	input string

Return values

brl	bracket levels

Definition at line 17 of file computeBracketLevel.m.

Here is the caller graph for this function:



Index