GenSSI

'2017-08-02'

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 converting models to polynomial form, or to or from AMICI format.

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/thomassligon/GenSSI and a direct clone is possible via

```
git clone https://github.com/thomassligon/GenSSI.git GenSSI
```

1.2.2 License Conditions

This software is available under the BSD license

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.

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 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 Name

Give the model a name.

```
model.Name = 'Goodwin';
```

2.1.3 Derivatives

Set the number of derivatives to be calculated.

```
model.Nder = 8;
```

2.1.4 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.X = [x1 x2 x3];
```

Define the number of states.

```
model.Neq = 3;
```

2.1.5 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 parameters vector of parameters to be considered for identifiability.

```
model.Par = [p1 p2 p3 p4 p5 p6 p7 p8];
```

Specify the number of parameters to be considered for identifiability.

```
model.Npar = 8;
```

2.1.6 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.F=[A1 A2 A3];
```

2.1.7 Controls

Define the controls.

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

Define the number of controls.

```
model.Noc = 0;
```

Note that the length of the control vector should match the number of states, even if there are fewer controls.

2.1.8 Observables

Define the observables.

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

Define the number of obserables.

```
model.Nobs = 1;
```

2.1.9 Initial Conditions

Define the initial conditions.

```
model.IC = [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, and the second parameter is the format. If the format is absent, the model is assumed to be a function, as described above. If it is equal to 'mat', the model is assumed to be a Matlab file with name Modelname.mat (e.g. Goodwin.mat) and containing the model struct.

```
genssiMain('Goodwin')
```

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

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

```
genssiFromSBML(modelNameIn,modelNameOut)
```

genssiFromSBML converts an SBML model to GenSSI format.

modelNameIn: name of the SBML model (string).

modelNameOut: name of the GenSSI model (string).

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. In addition, the GenSSI model created by the conversion contains default values for parameters such as the number of derivatives. It may be necessary to manually edit the GenSSI model after conversion.

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, so that our SBML model is now contained in

```
Kholodenko.xml
```

Now we are ready to call the conversion routine, where we have chosen to use the same name for the GenSSI model.

```
genssiFromSBML('Kholodenko','Kholodenko');
```

As mentioned above, we need to make some manual modifications to the GenSSI model. First, we will show the final version of this model, and then explain the changes that were made.

```
function model = Kholodenko()
    syms MKKK MKKK_P MKK MKK_P MKK_PP MAPK MAPK_P MAPK_PP
    syms V1_J0 Ki_J0 n_J0 K1_J0 V2_J1 KK2_J1 k3_J2 KK3_J2 k4_J3 KK4_J3 V5_J4
    syms KK5_J4 V6_J5 KK6_J5 k7_J6 KK7_J6 k8_J7 KK8_J7 V9_J8 KK9_J8 V10_J9 KK10_J9
    model.Name = 'Kholodenko';
    model.Nder = 6;
    model.X = [MKKK,MKKK_P,MKK,MKK_P,MKK_PP,MAPK,MAPK_P,MAPK_PP];
    model.Neq = 8;
    model.G = [0,0,0,0,0,0,0,0];
    model.Noc = 0;
    model.P = [V1_J0,Ki_J0,n_J0,K1_J0,V2_J1,KK2_J1,k3_J2,KK3_J2,k4_J3,...
               KK4_J3,V5_J4,KK5_J4,V6_J5,KK6_J5,k7_J6,KK7_J6,k8_J7,...
               KK8_J7,V9_J8,KK9_J8,V10_J9,KK10_J9];
    model.Par = [V1_J0,V2_J1,V5_J4,V6_J5,V9_J8,V10_J9];
    model.Npar = 6;
    model.IC = [90,10,280,10,10,280,10,10];
    model.H = [MAPK,MAPK_P,MAPK_PP];
    model.Nobs = 3;
    model.F = [(MKKK_P*V2_J1)/(KK2_J1 + MKKK_P) - (MKKK*V1_J0)/(((MAPK_PP/Ki_J0)^n_J0 + 1)*(K1_J0 + MKKK)),
      ...
       (MKKK*V1_J0)/(((MAPK_PP/Ki_J0)^n_J0 + 1)*(K1_J0 + MKKK)) - (MKKK_P*V2_J1)/(KK2_J1 + MKKK_P),...
       (MKK_P*V6_J5)/(KK6_J5 + MKK_P) - (MKK*MKKK_P*k3_J2)/(KK3_J2 + MKK),...
       (MKK_PP*V5_J4)/(KK5_J4 + MKK_PP) - (MKK_P*V6_J5)/(KK6_J5 + MKK_P)...
           + (MKK*MKKK_P*k3_J2)/(KK3_J2 + MKK) - (MKK_P*MKKK_P*k4_J3)/(KK4_J3 + MKK_P),...
       (MKK_P*MKKK_P*k4_J3)/(KK4_J3 + MKK_P) - (MKK_PP*V5_J4)/(KK5_J4 + MKK_PP),...
       (MAPK_P*V10_J9)/(KK10_J9 + MAPK_P) - (MAPK*MKK_PP*k7_J6)/(KK7_J6 + MAPK),...
       (MAPK_PP*V9_J8)/(KK9_J8 + MAPK_PP) - (MAPK_P*V10_J9)/(KK10_J9 + MAPK_P)...
           + (MAPK*MKK_PP*k7_J6)/(KK7_J6 + MAPK) - (MAPK_P*MKK_PP*k8_J7)/(KK8_J7 + MAPK_P),...
       (MAPK_P*MKK_PP*k8_J7)/(KK8_J7 + MAPK_P) - (MAPK_PP*V9_J8)/(KK9_J8 + MAPK_PP)];
end
```

Here are the changes we made:

First, we split the long lines to make the file more readable (syms, model.P, and model.F). For very large models, some users might prefer to use the .mat version of the model instead of the .m version.

Next we defined the observables (model.H and model.Nobs). Here, we chose the three downstream species MAPK, MAPK_P, and MAPK_PP.

Then we moved the full list of parameters from model.Par to model.P. This is not necessary, since model.P is only used in some of the conversion routings, such as conversion to AMICI.

At this point, we need to make an important change. The full list of 22 parameters is too long for analysis on our 8GB PC, because each Lie derivative increases the size of the Jacobian matrix by a facter of Npar, so 22 means that the memory usage grows very fast and leads to a problem. Here we need to choose a smaller number of parameters. We have chosen the 6 'V' parameters (model.Par and model.Npar). In this context, we also changed the number of derivatives (model.Nder) to 6. These changes were made on the basis of experimentation, by running the analysis multiple times, and searching for a combination where the number of parameters and derivatives were small enough to enable the analysis to run to completion, but with enough parameters to give a meaningful result and enough derivatives to achieve full rank of the Jacobian (equal to the number of parameters).

Finally, we are ready to analyze the model in GenSSI.

```
genssiMain('Kholodenko');
```

The results show that 3 of the parameters are locally identifiable and the other 3 are globally identifiable (assuming the other 16 to be fixed).

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 Convert to AMICI Format

```
genssiToAMICI(modelNameIn,modelNameOut)
```

genssiToAMICI converts a GenSSI model to AMICI format. The AMICI package uses Sundials Cvodes to efficiently solve ODEs from within Matlab. It is available at https://github.com/AMICI-developer/AMICI.

modelNameIn: name of the GenSSI model (string).

modelNameOut: name of the AMICI model (string).

Note: There are limitations to this conversion. The GenSSI model contains a list of parameters to be considered for analysis, but AMICI needs a "sym" statement containing a list of all parameters used by the model. It may be necessary to manually edit the AMICI model after conversion. In order to avoid this, the GenSSI model can contain a variable model.P, containing all parameters (model.Par only contains the parameters to be considered for identifiability).

2.3.4 Convert from AMICI Format

```
genssiFromAMICI(modelNameIn,modelNameOut)
```

genssiFromAMICI converts an AMICI model to GenSSI format.

modelNameIn: name of the AMICI model (string).

modelNameOut: name of the GenSSI model (string).

Note: There are limitations to this conversion. The AMICI model contains a list of all parameters used by the model, but GenSSI needs a list of parameters to be considered for analysis. In addition, the GenSSI model created by the conversion contains default values for parameters such as the number of derivatives. It may be necessary to manually edit the GenSSI model after conversion.

2.3.5 Convert from Model Structure to Model Source Format

```
genssiStructToSource(model)
```

```
amiciStructToSource(model)
```

genssiStructToSource reads the GenSSI model struct and converts it to source format (Matlab function definition), and amiciStructToSource does the same for AMICI models. In general, the source format is more convenient for smaller models, since it is easier to modify, but the struct format, typically saved in a Matlab file (e.g. Goodwin.mat) is more convenient for large models, since it does not require editing of long lines of code.

model: model definition (struct).

2.3.6 Create Multi-Experiment Model

```
genssiMultiExperiment(modelNameIn,fileFormat,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).

fileFormat: either 'function' (default), if the model is a MatLab function, or 'mat', if the model is a .mat file.

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 tempurature, 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()
    model.Name='M1_1_U1';
    syms  m G d b kTL m0
    model.Nder=4;
    model.X=[m G];
    model.Neq=2;
    A1=-d*m;
    A2=0;
    model.F=[A1 A2];
    model.G=[0,kTL*m;...
             0,-b*G];
    model.Noc=2;
    h1=G;
    model.H=[h1];
    model.Nobs=1;
    model.IC=[m0 0];
    model.P=[d b kTL m0];
    model.Par=[d b kTL m0];
    model.Npar=4;
end
```

The states (model.X) are m (mRNA) and G (GFP, green fluorescent protein), of which only GFP is observed (model.H). The differential equations (model.F) 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.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 IC: 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()
    multiExp.Name='M1_1_eDef4';
    syms d b kTL m0Exp1
    multiExp.Nexp=4;
    multiExp.U = [1,1;...
                  1,1;...
                  .5,1;...
                  1,.75];
    multiExp.IC = [m0Exp1,0,...
                   0.5*m0Exp1,0,...
                   m0Exp1,0,...
                   m0Exp1,0];
    multiExp.P=[d,b,kTL,m0Exp1];
    multiExp.Par=[d,b,kTL,m0Exp1];
end
```

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

The variable multiExp.U defines the changes in the controls and the variable multiExp.IC 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','function','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.Name = 'M1_1_ME4';
    model.Nder = 4;
    model.X = [mExp1,GExp1,mExp2,GExp2,mExp3,GExp3,mExp4,GExp4];
    model.Neq = 8;
    model.G = [0,0,0,0,0,0,0,0];
    model.Noc = 0;
    model.P = [d,b,kTL,m0Exp1];
    model.Par = [d,b,kTL,m0Exp1];
    model.Npar = 4;
    model.IC = [m0Exp1,0,m0Exp1/2,0,m0Exp1,0,m0Exp1,0];
    model.H = [GExp1,GExp2,GExp3,GExp4];
    model.Nobs = 4;
    model.F = [-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.X). In addition, all parameters now appear directly in the differential equations (model.F), and there are no controls.

2.3.7 Transform Model

```
genssiTransformation(modelNameIn,fileFormat,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).

fileFormat: either 'function' (default), if the model is a MatLab function, or 'mat', if the model is a .mat file.

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()
    model.Name='M2_1_Y1';
    syms m G E1 mE d1 d2 d3 b kTL m0 E0
    model.Nder=8;
    model.Neq=4;
    model.Npar=7;
    model.Noc=0;
    model.Nobs=1;
    model.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.F=[A1 A2 A3 A4];
    g1=0;g2=0;g3=0;g4=0;
    model.G=[g1 g2 g3 g4];
    h1=G;
    model.H=[h1];
    model.IC=[m0 0 E0 0];
    model.P=[d1 d2 d3 b kTL m0 E0];
    model.Par=[d1 d2 d3 b kTL m0 E0];
end
```

The state variables in this model (model.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()
    transDef.Name='M2_1_tDef';
    syms m G E1 mE m0 E0
    syms mdm0 E1dm0
    syms d1 d2 d3 b kTL
    syms d2tm0 kTLtm0 E0dm0
    transDef.Transformation = [m/m0;G;E1/m0];
    transDef.Constraint = [E1-E0+mE];
    transDef.P = [d1,d2tm0,d3,b,kTLtm0,E0dm0];
    transDef.Par = [d1,d2tm0,d3,b,kTLtm0,E0dm0];
    syms mnew Gnew E1new
    transDef.stateSubs = [mdm0,mnew;...
                          G,Gnew;...
                          E1dm0,E1new];
    transDef.parSubs = [d2*m0,d2tm0;...
                        kTL*m0,kTLtm0;...
                        E0/m0,E0dm0];
end
```

The transformation is defined by two variables. The first, transDef.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.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.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','function','M2_1_tDef','M2_2_test');
```

The result of the transformation is the following new model:

```
function model = M2_2_test()
    syms mnew Gnew E1new
    syms d1 d2tm0 d3 b kTLtm0 E0dm0
    model.Name = 'M2_2_test';
    model.Nder = 8;
    model.X = [mnew,Gnew,E1new];
    model.Neq = 3;
    model.G = [0,0,0,0];
    model.Noc = 0;
    model.P = [d1,d2tm0,d3,b,kTLtm0,E0dm0];
    model.Par = [d1,d2tm0,d3,b,kTLtm0,E0dm0];
    model.Npar = 6;
    model.IC = [1,0,E0dm0];
    model.H = [Gnew];
    model.Nobs = 1;
    model.F = [- 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.F) 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 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

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 Class Index

4.1 Class List

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

5 Class Documentation

5.1 cell Class Reference

A MatLab cell array or matrix.

5.1.1 Detailed Description

This class is an artificially created class in doxygen to allow more precise type declarations

5.2 char Class Reference

A MatLab character array.

5.2.1 Detailed Description

This class is an artificially created class in doxygen to allow more precise type declarations and represents string-like types.

5.3 colvec Class Reference

A matlab column vector.

5.3.1 Detailed Description

This class is an artificially created class in doxygen to allow more precise type declarations

5.4 double Class Reference

A double value.

5.4.1 Detailed Description

This class is an artificially created class in doxygen to allow more precise type declarations. The MatLab type associated with this class is double.

5.5 function_handle Class Reference

A MatLab function handle.

5.5.1 Detailed Description

This class is an artificially created class in doxygen to allow more precise type declarations

5.6 integer Class Reference

An integer value.

5.6.1 Detailed Description

This class is an artificially created class in doxygen to allow more precise type declarations. Matlab types associated with this class are all int-types (int8, uint8 etc).

5.7 logical Class Reference

A boolean value.

5.7.1 Detailed Description

This class can be seen as synonym for boolean values/flags used inside classes. In order to stick with matlab conventions/datatypes, this class was named logical instead of bool or boolean.

This class is an artificially created class in doxygen to allow more precise type declarations

5.8 rowvec Class Reference

A matlab row vector.

5.8.1 Detailed Description

This class is an artificially created class in doxygen to allow more precise type declarations

5.9 struct Class Reference

A MatLab struct.

5.9.1 Detailed Description

This class is an artificially created class in doxygen to allow more precise type declarations

5.10 varargin Class Reference

A variable number of input arguments.

5.10.1 Detailed Description

This class is an artificially created class in doxygen to allow more precise type declarations.

For more information about the varargin argument see the MatLab documentation on varargin.

5.11 varargout Class Reference

A variable number of output arguments.

5.11.1 Detailed Description

This class is an artificially created class in doxygen to allow more precise type declarations.

For more information about the varargout argument see the MatLab documentation on varargout.

6 File Documentation

6.1 genssiComputeLieDerivatives.m File Reference

genssiComputeLieDerivatives computes Lie derivatives of the output functions (model.H), the state vectors (model.X), and the initial conditions (model.IC) with respect to the equations (model.F) and controls (model.G)

Functions

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

  genssiComputeLieDerivatives computes Lie derivatives of the output functions (model.H), the state vectors (model.X), and the initial conditions (model.IC) with respect to the equations (model.F) and controls (model.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

6.1.1 Function Documentation

6.1.1.1 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:

6.1.1.2 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

Return values

rankVector	vector of ranks (for results)

Definition at line 106 of file genssiComputeLieDerivatives.m.

6.2 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.

6.2.1 Function Documentation

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

Parameters

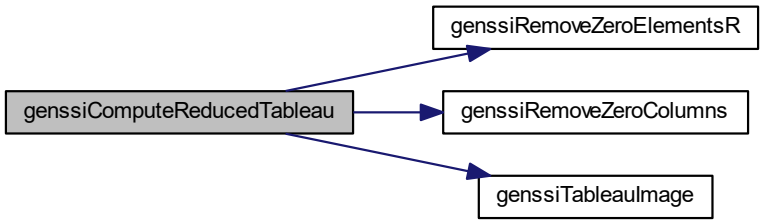
model	model definition (struct)
results	results of compute tableau (symbolic matrix)
VectorLieDerivatives	vector of Lie derivatives (symbolic array)
JacParam	jacobian with respect to parameters (symbolic matrix)
options	options (struct)

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:

6.3 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.

6.3.1 Function Documentation

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

Parameters

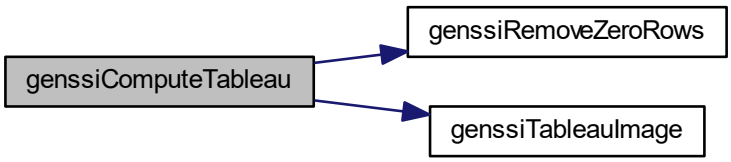
model	model definition (struct)
VectorLieDerivatives	vector of Lie derivatives (symbolic array)
options	options (struct)

Return values

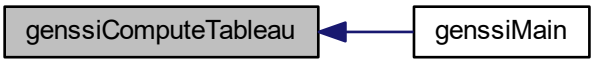
options	options (struct)
results	results of calculations (struct)
JacParam	jacobian of the Lie derivatives with respect to the parameters (symbolic matrix)

Definition at line 17 of file genssiComputeTableau.m.

Here is the call graph for this function:



Here is the caller graph for this function:



6.4 genssiFromAmici.m File Reference

GenSsiFromAmici converts an AMICI model to a GenSSI model and puts the results into the examples directory.

Functions

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

  GenSsiFromAmici converts an AMICI model to a GenSSI model and puts the results into the examples directory.

6.4.1 Function Documentation

6.4.1.1 mlhsInnerSubst< matlabtypesubstitute > genssiFromAmici (matlabtypesubstitute *modelNameIn*, matlabtypesubstitute *modelNameOut*)

Parameters

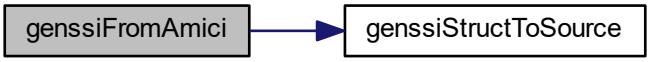
modelNameIn	name of the AMICI model (string)
modelNameOut	name of the GenSSI model (string)

Return values

modelNameOut	void

Definition at line 17 of file genssiFromAmici.m.

Here is the call graph for this function:



Here is the caller graph for this function:

6.5 genssiFromSBML.m File Reference

GenSsiFromSBML converts an SBML model to a GenSSI model and puts the results into the examples directory.

Functions

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

  GenSsiFromSBML converts an SBML model to a GenSSI model and puts the results into the examples directory.

6.5.1 Function Documentation

6.5.1.1 mlhsInnerSubst< matlabtypesubstitute > genssiFromSBML (matlabtypesubstitute *modelNameIn*, matlabtypesubstitute *modelNameOut*)

Parameters

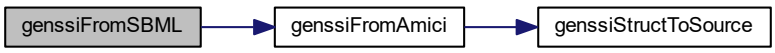
modelNameIn	name of the SBML model (string)
modelNameOut	name of the GenSSI model (string)

Return values

modelNameOut	void

Definition at line 17 of file genssiFromSBML.m.

Here is the call graph for this function:



6.6 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 varargin)

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

6.6.1 Function Documentation

6.6.1.1 mlhsInnerSubst< matlabtypesubstitute > genssiMain (matlabtypesubstitute *varargin*)

Parameters

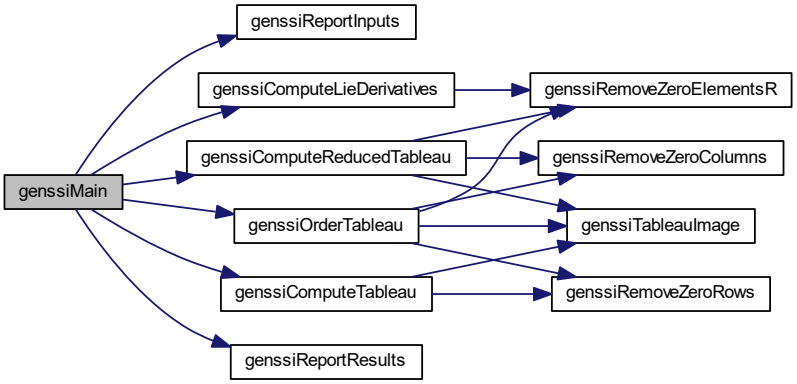
varargin	generic input arguments `1 genssiMain ( modelName, fileFormat, model, mat )` *Required Parameters for varargin:* • modelName the name of the model to be analyzed (a string) • fileFormat the format of the model file • model (default) if the model is a function file (e.g. Goodwin.m) • mat if the model is a Matlab file (e.g. Goodwin.mat)

Return values

varargout	generic output arguments
options	struct containing options

Definition at line 17 of file genssiMain.m.

Here is the call graph for this function:



6.7 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 > > 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 > > 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

6.7.1 Function Documentation

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

Parameters

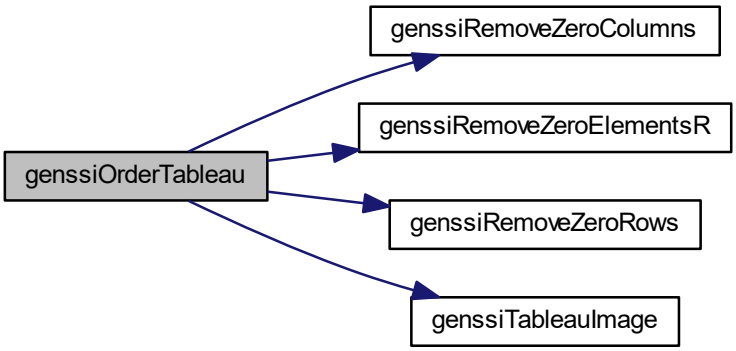
model	model definition (struct)
results	results of previous steps (struct)
RJacParam01	reduced tableau, i.e. binary form of jacobian of the Lie derivatives with respect to the parameters (binary matrix)
ECC	equations (symbolic array)
rParam	reduced list of parameters (symbolic array)
options	options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

Return values

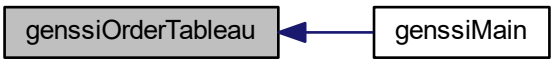
options	options (struct)
results	results of previous steps (struct)

Definition at line 17 of file genssiOrderTableau.m.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8 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

6.8.1 Function Documentation

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

Parameters

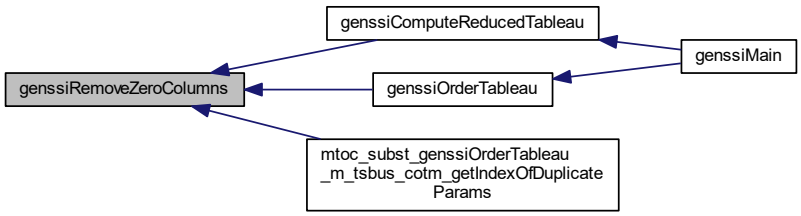
matrixIn	input (matrix)

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:



6.9 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

6.9.1 Function Documentation

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

Parameters

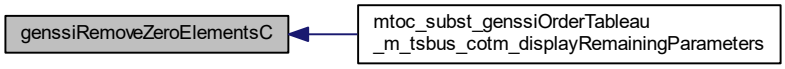
vectorIn	input (array)

Return values

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

Definition at line 17 of file genssiRemoveZeroElementsC.m.

Here is the caller graph for this function:



6.10 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

6.10.1 Function Documentation

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

Parameters

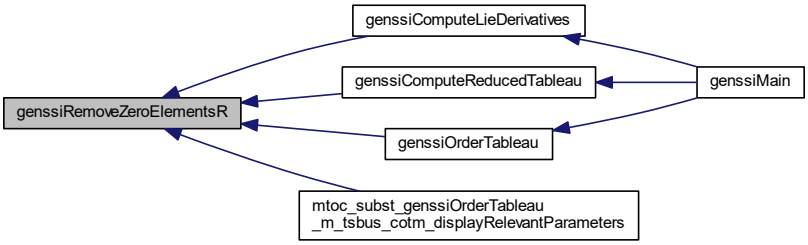
vectorIn	input (array)

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:



6.11 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

6.11.1 Function Documentation

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

Parameters

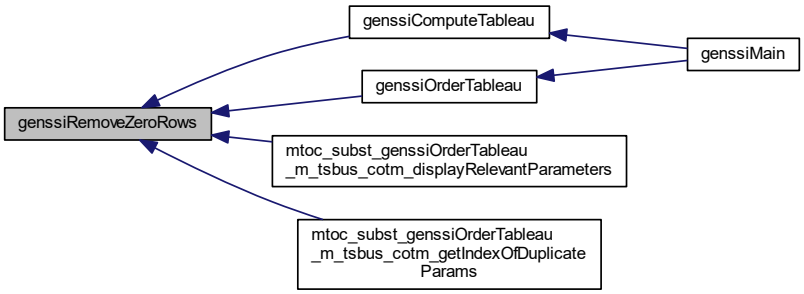
matrixIn	input (matrix)

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:



6.12 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.

6.12.1 Function Documentation

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

Parameters

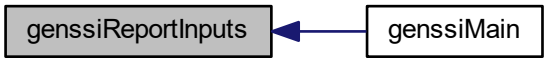
model	model definition (struct)
options	options (struct)

Return values

options	options (struct)

Definition at line 17 of file genssiReportInputs.m.

Here is the caller graph for this function:



6.13 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.

6.13.1 Function Documentation

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

Parameters

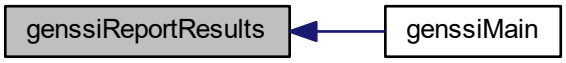
model	model definition (struct)
results	results of previous steps (struct)
options	options (struct)

Return values

options	options (struct)

Definition at line 17 of file genssiReportResults.m.

Here is the caller graph for this function:



6.14 genssiStartup.m File Reference

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

Functions

- noret::substitute genssiStartup ()

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

6.15 genssiStructToSource.m File Reference

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

Functions

- noret::substitute genssiStructToSource (matlabtypesubstitute model)

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

6.15.1 Function Documentation

6.15.1.1 noret::substitute genssiStructToSource (matlabtypesubstitute *model*)

Parameters

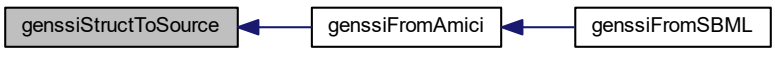
model	model definition (struct)

Return values

model	void

Definition at line 17 of file genssiStructToSource.m.

Here is the caller graph for this function:



6.16 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

6.16.1 Function Documentation

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

Parameters

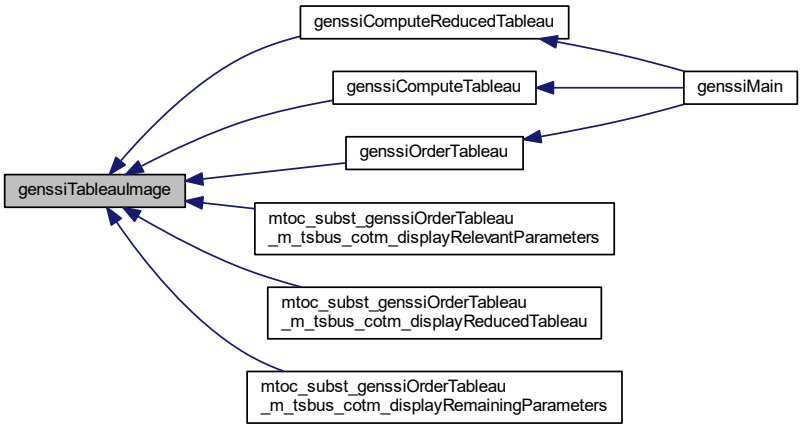
figNum	figure number
tabMat	matrix containing tableau
paramDisplay	parameter vector
options	options

Return values

options	void

Definition at line 17 of file genssiTableauImage.m.

Here is the caller graph for this function:



6.17 genssiToAmici.m File Reference

GenSsiToAmici converts a GenSSI model to AMICI model format and saves the results in the examples directory.

Functions

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

  GenSsiToAmici converts a GenSSI model to AMICI model format and saves the results in the examples directory.

6.17.1 Function Documentation

6.17.1.1 mlhsInnerSubst< matlabtypesubstitute > genssiToAmici (matlabtypesubstitute *modelNameIn,* matlabtypesubstitute *modelNameOut*)

Parameters

modelNameIn	name of the GenSSI model (string)
modelNameOut	name of the AMICI model (string)

Return values

modelNameOut	void

Definition at line 17 of file genssiToAmici.m.

6.18 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.

6.18.1 Function Documentation

6.18.1.1 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.

Index