



Visit the mor4ansys web page

Author: Evgenii Rudnyi, <http://Evgenii.Rudnyi.Ru/>

■ Imtek`Post4MOR`: Manual

The goal of the package is to make transient or harmonic simulations for a system of ordinary differential equations of the first or second order. I have written it to support simulation for a reduced model obtained by mor4ansys.

Examples can be found in two separate notebooks, FirstOrderSystem.nb and SecondOrderSystem.nb. This notebook contains formal description of objects DynamicSystem and SimulationResult as well as simulation and plotting functions.

■ DynamicSystem

The object represent a system of ordinary differential equations in the form as follows

$$\begin{aligned} \frac{dx}{dt} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \quad (1)$$

$$\begin{aligned} E \frac{dx}{dt} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \quad (2)$$

$$\begin{aligned} E \frac{dx}{dt} + Kx &= Bu \\ y &= Cx + Du \end{aligned} \quad (3)$$

$$\begin{aligned} M \frac{d^2 x}{dt^2} + E \frac{dx}{dt} + Kx &= Bu \\ y &= Cx + Du \end{aligned} \quad (4)$$

The object contains the system matrices. Matrices M, E and D are optional, matrices A(K), B and C must be present. In addition the object contains output names for the vector y.

Constructors

<pre>MakeDynamicSystem [matM,matE, matK,matB,matC,namesC,matD] and MakeDynamicSystem [matM,matE,matK,matB,matD]</pre>	<p>give a DynamicSystem object. matM,matE, and matD should be matrices or Null. matK, matB, and matC must be matrices. namesC is a list of strings. matD is optional and by default is Null. The second form makes matC equal to the identity matrix.</p>
<pre>MakeFirstOrderSystem [matA, matB,matC,namesC,matD], MakeFirstOrderSystem [matE, matA,matB,matC,namesC,matD], and MakeFirstOrderSystem [matE,matA,matB,matD]</pre>	<p>give a DynamicSystem object for a first order system. matE, and matD should be matrices or Null. matA, matB, and matC must be matrices. namesC is a list of strings. matD by default is Null. The last form makes matC equal to the identity matrix.</p>
<pre>MakeSecondOrderSystem [matM, matK,matB,matC,namesC,matD], MakeSecondOrderSystem [matM,matE,matK,matB, matC,namesC,matD] and MakeSecondOrderSystem [matM,matE,matK,matB,matD]</pre>	<p>give a DynamicSystem object for a second order system. matM,matE, and matD should be matrices or Null. matK, matB, and matC must be matrices. namesC is a list of strings. matD is optional and by default is Null. The last form makes matC equal to the identity matrix.</p>

MakeFirstOrderSystem follows the convention of Eq (1) and (2) while MakeDynamicSystem and MakeSecondOrderSystem follow the convention of Eq (3) and (4). The difference is in the sign of MatrixK (MatrixA = - MatrixK).

If dimensions of different matrices are not compatible, the constructors throw an exception.

Selectors

MatrixM [sys]	gives system matrix M of sys.
MatrixE [sys]	gives system matrix E of sys.
MatrixA [sys]	gives system matrix A of sys.
MatrixK [sys]	gives system matrix K of sys.
MatrixB [sys]	gives system matrix B of sys.
MatrixC [sys]	gives system matrix C of sys.
MatrixD [sys]	gives system matrix D of sys.
OutputNames [sys]	gives a list of output names of sys.

If a matrix does not present in the system, a selector returns a default matrix: the zero matrix for MatrixD; for a first order system: the zero matrix for MatrixM and the unity matrix for MatrixE; for a second order system: the unity matrix for MatrixM and the zero matrix for MatrixE.

Predicates

MatrixMQ[sys]	gives True if system matrix M is present in sys and False otherwise.
MatrixEQ[sys]	gives True if system matrix E is present in sys and False otherwise.
MatrixDQ[sys]	gives True if system matrix D is present in sys and False otherwise.
FirstOrderSystemQ[sys]	gives True if sys represent a first order system and False otherwise.
SecondOrderSystemQ[sys]	gives True if sys represent a second order system and False otherwise.
ExplicitSystemQ[sys]	gives True if sys represent an explicit first order system and False otherwise.

Transformation Functions

DeleteOutputs[sys, names]	gives a new system when outputs defined in a list of names are deleted.
AddOutputs[sys, matCadd, names]	gives a new system with additional outputs defined by rows of matCadd and a list of names.
DeleteDamping[sys]	gives a new system without system matrix E. sys must be a second order system.
AddDamping[sys, matE]	gives a new system with matrix E increased by matE. sys must be a second order system.
AddDamping[sys, alpha, beta]	gives a new system with matrix E increased by Rayleigh damping $\alpha \cdot \text{matM} + \beta \cdot \text{matK}$. sys must be a second order system.
AddDampingRatio[sys, imatK]	gives a new system with matrix K increased by $I \cdot \text{imatK}$. sys must be a second order system.
AddDampingRatio[sys, dmprat]	gives a new system with matrix K increased by $I^2 \cdot \text{dmprat} \cdot \text{MatrixK}$. sys must be a second order system.
ToFirstOrderSystem[sys]	gives a new system in the form of a first order system.
ToExplicitSystem[sys]	gives a new system in the form of an explicit first order system.
TakeSystem[sys, dim]	gives a new system of dimension dim. This operation makes sense only when an original system has been obtained by iterative model reduction method.

Input-Output

<code>ReadSystem [baseName]</code>	reads <code>DynamicSystem</code> from files with the base name <code>baseName</code> .
<code>WriteSystem [sys,baseName]</code>	writes a system to files with the base name <code>baseName</code> .
<code>ReadSystem [file,type]</code>	reads a system from a file written in the old format of <code>mor4ansys 1.5</code> . <code>type</code> must be a string <code>FirstOrderSystem</code> or <code>SecondOrderSystem</code> .

The external representation of `DynamicSystem` is the same as produced by `mor4ansys` and accepted in Oberwolfach Model Reduction Benchmarks.

■ SimulationResult

The object represent results of transient or harmonic simulation for `DynamicSystem`. We have an X-axis (time or frequency) and results for all outputs for an Y-axis. As such, there are four components: `XSeries`, `XName`, `YSeries`, `YNames`. `XSeries` is a vector and `YSeries` is a matrix. Names are text strings and they allow us to decide if two objects are compatible with each other.

Constructor

<code>MakeSimulationResult [XSeries,XName,YSeries,YNames]</code>	gives a <code>SimulationResult</code> object. <code>XSeries</code> must be a vector, <code>YSeries</code> must be a matrix, <code>XName</code> must be a string and <code>YNames</code> must be a list of strings.
--	--

If dimensions of different components are not compatible, the constructor throws an exception.

Selectors

<code>XSeries [res]</code>	gives <code>XSeries</code> of <code>res</code> .
<code>XName [res]</code>	gives <code>XName</code> of <code>res</code> .
<code>YSeries [res]</code>	gives <code>YSeries</code> of <code>res</code> .
<code>YNames [res]</code>	gives <code>YNames</code> of <code>res</code> .

Predicates

<code>CompatibleResultQ [res1,res2,...]</code>	gives <code>True</code> if all objects are compatible and <code>False</code> otherwise.
--	---

The criterion for compatibility is that all names are the same and in the same order.

Transformation Functions

<code>TransformResult [res, names]</code>	gives a new res whose YSeries correspond to names. res also can be a list of SimulationResult.Option TransformFunction to define fun[y,x].
<code>Difference [res1, {listres}, ops]</code>	gives a list of differences between res1 and each object in listres. All SimulationResult must be compatible. Option is ErrorFunction, by default ErrorFunction→Subtract.
<code>TakeResult [res, num]</code>	takes first num values for results.

You may want to reduce the number of YSeries before plotting or to re-sort them. To this end, use TransformResult. TransformFunction is an option to define a function fun[y, x]. If it is a single function, it is applied to all YSeries. Alternatively, one can specify a list of pairs {name, fun}.

Another operation is to find a difference between different results for a difference plot. You define what the difference means by using an option ErrorFunction. In the case, if XSeries are different, Difference uses Interpolation. You can specify its options as additional arguments to Difference.

Plotting Function

<code>PlotResult [listres, ops]</code>	makes plots for the list of compatible SimulationResult. Options: CommonTitle for a title, FunctionX and FunctionY to modify series values, MultipleListPlot options.
--	---

PlotResult is a front end to MultipleListPlot to make a series of plots. Each plot contains a single output for all SimulationResult in listres.

Use Difference to make a difference plot.

Use options FunctionY and FunctionX to modify values before plotting. By default, they are equal to Identity.

Use MultipleListPlot options to make your plot look nicer.

Input-Output

<code>ReadResult [file]</code>	reads SimulationResult from file in the Matrix Market format.
<code>WriteResult [res, file]</code>	writes res to file in the Matrix Market format.
<code>WriteResult [res, file, "Table "]</code>	writes res to file as space separated values. This is convenient if you would like to use another plotting software.
<code>ReadResult [file, XName]</code>	reads SimulationResult from file in my old format, where XName should be "Time" or "Frequency".
<code>WriteResultOld [res, file]</code>	writes res to file in my old format.
<code>WriteResultAsDirichlet [res, i, file]</code>	writes the i-th set of res to file as ANSYS script to fix values as Dirichlet constraints.

An external representation of SimulationResult is based on the MatrixMarket format. Additionally, file file.names contains XName and YNames.

■ Functions for Transient and Harmonic Simulation

StationorySolution [sys,ops]	gives a stationary solution. Options: InputFunction, LinearSolve options.
HarmonicSolution [freq,sys,ops]	gives SimulationResult for a given list of freq. Options: InputFunction, LinearSolve options.
AnsysTransientSolution [time,sys,ops]	gives SimulationResult for a given list time by using integrators like in ANSYS: backward Euler for a first order system and Newmark for a second order system. Options: InputFunction, InitialState, TINTP. TINTP→ {gamma,alpha,delta,theta} allows us to change integrator behavior (see ANSYS manual).
TransientSolution [time,sys,ops]	gives SimulationResult for a given list time. Options: Verbose, InputFunction, InitialState, NDSolve options. It uses different approaches for different types of DynamicSystem. Use ToFirstOrderSystem or ToExplicitSystem, if TransienSolution fails.
FrequencyConvergence [freq,sys,start,finish] or FrequencyConvergence [freq,sys,start,finish,step]	gives an object for LocalError and LocalErrorIndicator computed for sys with dimensions from start to finish for frequency freq.
LocalError [conv,true,ops]	gives a list of local errors computed from the object conv obtained by FrequencyConvergence and a list of true values for the same frequency. Option is ErrorFunction (by default Log10RelativeError).
LocalErrorIndicator [conv,true,ops]	gives a list of local error estimates computed from the object conv obtained by FrequencyConvergence. Option is ErrorFunction (by default Log10RelativeError).

HarmonicSolution sets XName to "Frequency". TransientSolution sets XName to "Time".

By default, an input function is a unit step function for each input. You can modify this, by using an option InputFunction. For HarmonicSolution, InputFunction must be a list of numerical values. For TransientSolution, InputFunction can be a list of numerical values or a list of functions in two arguments: time and a state vector. InitialState by default is zero.

HarmonicSolution uses LinearSolve and you can use its options as additional arguments. TransientSolution uses NDSolve and you can use its options. AnsysTransientSolution implements integrators from ANSYS, use TINTP to set integrator constants.

NDSolve in TransientSolution uses only the last value in list time. It chooses integration points by itself adaptively. Use Verbose->True to see how many timesteps NDSolve has made. AnsysTransientSolution makes timesteps according to list time. As a result, AnsysTransientSolution is much faster but TransientSolution gives more accurate result.

Functions LocalError and LocalErrorIndicator can be used to choose an optimal dimension for the reduced model as described in paper Error indicators for fully automatic extraction of heat-transfer macromodels for MEMS.