Tutorial on dymmat toolbox

About

High-level matlab scripts for processing and manipulating Dymola simulations (e.g. Within an optimization). The toolbox works with and requires the matlab files distributed with Dymola. The toolbox is particularly useful with large-scale simulations subject to computational operations e.g.: identifiably analysis or optimization

Author

Atiyah Elsheikh, Energy Department, Austrian Institute of Technology GmbH, Vienna, Autstria

Email

Atiyah.Elsheikh-at-ait.ac.at

A.M.G.Elsheikh-at-gmail.com

Homepage

https://www.researchgate.net/profile/Atiyah_Elsheikh

License

Release Notes

- V1.0 : first version (August/2014)

Contents

bin/

Code for producing a user-defined timed simulation of dymola executables, where a simulation can last for a

Very useful for large-scale simulations where optimization algorithms may suggest infeasible parameter values, that cause a very slow simulation.

Currently not distributed within this version. Contact the author if you may need it.

Examples/

Some Modelica models for this tutorial

mfiles/

Matlab source code. You can put it with /dymola/path/mfiles directly or in a seperate directory as well. For the included matlab files, see dymmat/Contents.m , or

```
>> help dymmat/Contents.m
```

mfiles/mymfiles

Seperate directory for user-implemented matlab files

mfiles/mymfiles/traj

Contains a modified version of the dymosim.m Dymola code.

It is not distributed as an open-source code. Contact the author if you may need it

mfiles/mymfiles/dymmat

The main matlab scripts within the dymmat toolbox

```
>> help dymmat/Contents.m
```

Initial step

1. Go to /path/to/dymmat/mfiles and edit add_extra_to_start_up.m, this adds all matlab scripts to the path variable as well as Dymola mfiles
2. You need to modify variable dymhome to the path to the dymola mfiles

```
dymhome = '/path/to/dymola/mfiles';
...

>> cd /path/to/dymmat/mfiles
>> add_extra_to_start_up
```

Setting a Dymola simulation record

```
>> cd /path/to/dymmat/examples/circuit
>> ls
      ChuaCircuit.mo . ..
```

This a copy of an electrical circuit example from the Modelica standard library. You need to translate it using Dymola to produce the required files. You can do that within Matlab as well, using provided mfiles from Dymola.

```
>> ls
      ChuaCircuit.mo dymosim.exe dsin.txt . ..
```

The first step is to create a Dymola simulation record using the setSimulation routine. The routine has the following signature:

```
[simpleSimOp] = setSimulations([simpleSimOp,]
                        'Property1',value1,
                        'Property2',value2,...)
```

The properties describe essential fields in the simulation record that charachterizes the simulation. Some of the properties can be set by the user (otherwise default values are considered). Some other properties are computed automatically. For detailed description of properties, see help.

Demo

```
>>  setSimulation()
arguments :
SimPar : vector of 6/7 values, see help dymosim
x0values : vector of length = length(x0), dymosim(simpar,x0,p)
pvalues : vector of length = length(p), dymosim(simpar,x0,p)
varName : cell array of regular expressions/strings
activePar : cell array of regular expressions/strings
newSim : true or false
saveAllRes : true or false
SimDir : a path to the directory of the original model as a string

>> simOpt = setSimulation()
simOpt =

       activepar: []
    activeparind: []
          alllab: []
          allres: []
          newsim: 1
         plabels: {12x1 cell}
         pvalues: [12x1 double]
      saveallres: 0
          simdir: 'D:\work\mydymola\dymmat\examples\circuit'
          simpar: [0 1 0.0100 0 1.0000e-004 8]
         succeed: []
            time: []
          varind: []
         varname: []
               x: []
        x0labels: {3x1 cell}
        x0values: [3x1 double]
```

Now, a simulation record is produced including some basic editable default values for some properties such as

- plabels: parameter labels
- pvalues : the corresponding parameter values
- x0labels and x0values similarly
- simpar: default simulation parameters required by the dymola routine dymosim, see help dymosim

- simdir: the directory of the simulator

Emphasizing interesting model parameters

While the properties plabels and pvalues provide a complete access to all model parameters, there are situations where only a subset of model parameters need to be emphasized for particular computational methods, e.g. subject to modification and tuning by an optimization or for which parameter sensitivities need to be computed.

The property name activepar is used for separating interesting parameters that are e.g. one can insert parameters iteratively one by one:

```
>> simOpt.plabels

ans =

    'L.L'
    'Ro.R'
    'Ro.T_ref'
    'Ro.alpha'
    'G.G'
    'G.T_ref'
    'G.alpha'
    'C1.C'
    'C2.C'
    'Nr.Ga'
    'Nr.Gb'
    'Nr.Ve'
```

A better way to list parameters:

```
>> listPars(simOpt)
    1                                   L.L  = 18.000000
    2                                  Ro.R  = 0.012500
    3                              Ro.T_ref  = 300.150000
    4                              Ro.alpha  = 0.000000
    5                                   G.G  = 0.565000
    6                               G.T_ref  = 300.150000
    7                               G.alpha  = 0.000000
    8                                  C1.C  = 10.000000
    9                                  C2.C  = 100.000000
```

```
   10                                                  Nr.Ga  = -0.757576
   11                                                  Nr.Gb  = -0.409091
   12                                                  Nr.Ve  = 1.000000

>> simOpt = setSimulation(simOpt,'activePar',{'Nr.Ga'});
>> simOpt = setSimulation(simOpt,'activePar',{'L.L'});
>> simOpt.activepar

ans =

    'L.L'
    'Nr.Ga'
```

or alltogether

```
>> simOpt = setSimulation(simOpt,'activePar',{'L.L','Nr.Ga'});
```

and even using regular expressions:

```
>> listParsReg(simOpt,'^G.*');
    5                                                     G.G  = 0.565000
    6                                                 G.T_ref  = 300.150000
    7                                                 G.alpha  = 0.000000

>> simOpt = setSimulation(simOpt,'activePar',{'^G.*'});
```

The indices are automatically computed

```
>> simOpt.activeparind

ans =

     1
     5
     6
     7
    10

>> listPars(simOpt,simOpt.activeparind)
    1                                                   L.L  = 18.000000
```

```
    5                                                  G.G  = 0.565000
    6                                              G.T_ref  = 300.150000
    7                                              G.alpha  = 0.000000
   10                                                Nr.Ga  = -0.757576
```

A warning is issued if the user sets a name of a parameter(s) that does not exist:

```
>> simOpt = setSimulation(simOpt,'activePar',{'^HHH.*'})
Warning: regular expression :^HHH.* not matchable
...
```

Emphasizing interesting model variables

Similar to emphasizing parameter names, user needs to emphasize the variable names for which their simulation results are interesting. The variable names and their indicies will be first checked and instantiated first when performing the first simulation by the routine loadSimulation. The loaded results will be stored in simOpt.x only for specified variable names.

```
>> simOpt = setSimulation(simOpt,'varName',{'L.i','Ro.i'});
```

The same operations with activepar are also supported for the property varname.

Running simulations

Using the created dymola simulation structure, simulations can be easily done:

```
>> simOpt = loadSimulation(simOpt);
```

The results will be stored in simOpt.x

Having produced One can also simulate in a different directory.

```
>> cd(/other/directory);
>> simOpt = loadSimulation(simOpt);
```

This is very useful when dealing with multiple simulations, e.g. Model simplifications etc.

Getting and setting specific parameter by name:

```
>> simOpt = setSimPar(simOpt,'L.L',17.1);
>> getSimPar(simOpt,'L.L')

ans =

   17.1000
```

Setting parameters (specific names, or matched by regular expressions) to a specific value or randomly according to a uniform distribution:

```
>> listParsReg(simOpt,'.*C$');
    8                                                  C1.C  = 50.000000
    9                                                  C2.C  = 50.000000
>> [simOpt,ncnt] = setSimParReg(simOpt,{'.*C$'},50.0);
>> listParsReg(simOpt,'.*C$');
    8                                                  C1.C  = 50.000000
    9                                                  C2.C  = 50.000000
>> ncnt

ncnt =

     2

>> [simOpt,ncnt] = setSimParRegUniform(simOpt,{'.*C$'},1,100.0);
>> listParsReg(simOpt,'.*C$');
    8                                                  C1.C  = 81.657645
    9                                                  C2.C  = 90.673402
```

See help dymmat/Contents and see what is the purpose of other functions.