DyMat - User Manual

version 0.3

Contents

1 Overview

This package contains some modules to read and process the result files from Dymola and OpenModelica with Python. A script is included that will help you to browse and export the contents of these files to other formats.

Both simulation systems save their results in regular mat-files, but use a special variable structure to store the data efficiently. An easy way to access the data is MATLAB, but some people (like me) don't have it or don't want to use it.

If you want to use this package in your own python scripts you should read the section Python API.

The documentation on the provided script DyMatExport.py is in the section DyMatExport.py.

2 Author, Homepage and License

DyMat is developed by Joerg Raedler (joerg@j-raedler.de).

Homepage: http://www.j-raedler.de/projects/DyMat

Development hosting: https://github.com/jraedler/DyMat

3 Dependencies

1. Python 2.x
2. SciPy
3. argparse for Python < 2.7

For MS Windows you may consider using the excellent distribution python(x,y), which includes almost everything you will ever need for scientific computing with python.

4 Installation

The installation process uses the distutils package which is included in standard python distribution. In the base folder of this package call:

```
$ python setup.py install
```

to install everything. Have a look at the distutils documentation for more options.

5 Background

If you do simulations with Modelica usually a lot of variables (names) share the same values because the different parts of the model are connected to each other. Dymola and OpenModelica will store those values only once in a mat-file and let different variable names point to the same data (possibly with a negative sign). Because of this structure accessing the values of a variable is not straightforward (that's the reason fro writing this package!).

If you export all variables the amount of data may explode because the same values are written a lot of times. It's possible that a mat-file of 1 MB results in a new file of several hundred MB. That's why there is no simple export-all option in DyMatExport.py.

You have been warned: only export the variables you really need!

The variables are stored as time rows. The number of elements in a row may be different. If a variable doesnt change, only the initial and the last value are stored. Other variables may change with every time step. That's why different data blocks with a different shape are contained in the same file. Each block has its own abscissa (time values). For some export formats you must not mix variables from different blocks!

6 Future Plans

- Implement some plot functions with matplotlib, qwt or gnuplot.
- Implement additional options for the export backend (like delimiter for CSV).
- Split the loading of the header data and other data from the mat-file. This will speedup some of the functions but needs scipy 0.10 to work.
- Improve the tree view which is just a simple hack.
- Fix all remaining bugs, do testing on different platforms

7 Export Formats

7.1 CSV

This is a well-known exchange format for tabular data. Variables are saved in columns with a header. Multi-block data will simply be appended.

7.2 Gnuplot

Is a special form of CSV with fields sparated by whitespace. A header is included to show the column numbers. You can plot those files directly:

```
gnuplot> plot "myfile" using 1:4 # plots column 4 over column 1
```

7.3 MATLAB

Every variable is stored as a single 1D matrix. This results in a simplified form of the original file, but in a bigger file with less metadata.

7.4 netCDF

netCDF is a highly efficient binary format for structured multi-dimensional data. You may use this option to prepare files for the ncDataReader2. This way you can use the results of simulations as (interpolated) input data for other simulations.

8 DyMatExport.py

This script is mainly a command-line interface for the python functions in this package. Every option has a long (*--help*) and a short (*-h*) form. Use this option to get help on the other options.

8.1 Exploring a file

General info on blocks and variables:

```
$ DyMatExport.py -i myfile.mat
```

List all variables in short form:

```
$ DyMatExport.py -l myfile.mat
```

List all variables including block number and description:

```
$ DyMatExport.py -d myfile.mat
```

Show a simple tree view of the variables:

```
$ DyMatExport.py -t myfile.mat
```

Show other variables which share data with the variable *foo.bar*:

```
$ DyMatExport.py -s foo.bar myfile.mat
```

8.2 Exporting variables

You need to specify a list of variables, either on the command line or read from a file. Variable names on the command line are comma-separated like this:

```
$ DyMatExport.py -e "foo.bar,baz,a.b.c" myfile.mat
```

This will export the variables *foo.bar*, *baz* and *a.b.c*. For more than just a couple of names you should consider using a file with names instead:

```
$ DyMatExport.py -x vars.txt myfile.mat
```

You may use the output from -l or -d to produce such files and just delete lines you don't want:

```
$ DyMatExport.py -d myfile.mat > vars.txt
$ $EDITOR vars.txt   # delete unwanted variables
$ DyMatExport.py -x vars.txt myfile.mat
```

The output format defaults to *CSV*, you may specify other formats like this:

```
$ DyMatExport.py -f netCDF -x vars.txt myfile.mat
```

Get a list of supported formats:

```
$ DyMatExport.py -m myfile.mat
```

The output file name defaults to the name of the mat-file with a format-specific suffix. You may specify an alternative name:

```
$ DyMatExport.py -o mynewfile.nc -f netCDF -x vars.txt myfile.mat
```

This will write all variables named in the file *vars.txt* to a file *mynewfile.nc* in the netCDF format.

9 Python API

9.1 DyMatFile

This class is the main part of the package, it reads a mat-file and provides access to its data. The class can be imported directly from DyMat. The constructor needs a filename as argument:

```
>>> import DyMat
>>> d = DyMat.DyMatFile('myfile.mat')
```

9.1.1 DyMatFile.blocks()

Returns the numbers of all data blocks.

Arguments:
- None

Returns:
- sequence of integers

9.1.2 DyMatFile.names(|block=None)

Returns the names of all variables. If block is given, only variables of this block are listed.

Arguments:
- optional block: integer

Returns:
- sequence of strings

9.1.3 DyMatFile.data(varName)

Return the values of the variable.

Arguments:
- varName: string

Returns:
- numpy.ndarray with the values

9.1.4 DyMatFile[varName]

The same as *DyMatFile.data(varName)*.

9.1.5 DyMatFile.block(varName)

Returns the block number of the variable.

Arguments:
- varName: string

Returns:
- integer

9.1.6 DyMatFile.description(varName)

Returns the description string of the variable.

Arguments:
- varName: string

Returns:

- string

9.1.7 DyMatFile.sharedData(varName)

Return variables which share data with this variable, possibly with a different sign.

Arguments:

- varName: string

Returns:

- sequence of tuples, each containing a string (name) and a number (sign)

9.1.8 DyMatFile.size(blockOrName)

Return the number of rows (time steps) of a variable or a block.

Arguments:

- integer (block) or string (variable name): blockOrName

Returns:

- integer

9.1.9 DyMatFile.abscissa(blockOrName |, valuesOnly=False)

Return the values, name and description of the abscissa that belongs to a variable or block. If valuesOnly is true, only the values are returned.

Arguments:

- integer (block) or string (variable name): blockOrName
- optional bool: valuesOnly

Returns:

- numpy.ndarray: values or
- tuple of numpy.ndarray (values), string (name), string (description)

9.1.10 DyMatFile.sortByBlocks(varList)

Sort a list of variables by the block number, return a dictionary whose keys are the block numbers and the values are lists of names. All variables in one list will have the same number of values.

Arguments:

- sequence of strings: varList

Returns:

- dictionary with integer keys and string lists as values

9.1.11 DyMatFile.nameTree()

Return a tree of all variable names with respect to the path names. Path elements are separated by dots. The tree will represent the structure of the Modelica models. The tree is returned as a dictionary of dictionaries. The keys are the path elements, values are sub-dictionaries or variable names.

Arguments:

- None

Returns:

- dictionary

9.1.12 DyMatFile.getVarArray(varNames |, withAbscissa=True)

Return the values of all variables in varNames combined as a 2d-array. If withAbscissa is True, include abscissa's values first. **All variables must share the same block!**

Arguments:

- sequence of strings: varNames
- optional bool: withAbscissa

Returns:

- numpy.ndarray

9.1.13 DyMatFile.writeVar(varName)

Write the values of the abscissa and the variabale to stdout. The text format is compatible with gnuplot. For more options use DyMat.Export instead.

Arguments:

- string: varName

Returns:

- None

9.2 DyMat.Export

This module contains a generic export function and a dictionary of available export formats (*formats*).

9.2.1 Export.export(fmt, dm, varList |, fileName=None, formatOptions=None)

Export the data of the DyMatFile object *dm* to a data file. *fmt* is the format string, *varList* the list of variables to export. If no *fileName* is given, it will be derived from the mat file name. *formatOptions* will be used in later versions.

Arguments:

- string: fmt
- DyMolaMat object: dm
- sequence of strings: varList
- optional string: fileName
- optional dictionary: formatOptions

Returns:

- None

There are sub-modules for every format. Each module contains a function called *export* which has the same interface as the above function but without the first argument. You may import one of these modules or use the generic function above.