

# DyMat - User Manual

version 0.3

## Contents

|                                       |          |
|---------------------------------------|----------|
| <b>1 Overview</b>                     | <b>1</b> |
| <b>2 Author, Homepage and License</b> | <b>1</b> |
| <b>3 Dependencies</b>                 | <b>1</b> |
| <b>4 Installation</b>                 | <b>2</b> |
| <b>5 Background</b>                   | <b>2</b> |
| <b>6 Future Plans</b>                 | <b>2</b> |
| <b>7 Export Formats</b>               | <b>2</b> |
| 7.1 CSV                               | 2        |
| 7.2 Gnuplot                           | 3        |
| 7.3 MATLAB                            | 3        |
| 7.4 netCDF                            | 3        |
| <b>8 DyMatExport</b>                  | <b>3</b> |
| 8.1 Exploring a file                  | 3        |
| 8.2 Exporting variables               | 3        |
| <b>9 Python API</b>                   | <b>4</b> |

## 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](#) is in the section [DyMatExport](#).

## 2 Author, Homepage and License

DyMat is developed by Joerg Raedler ([joerg@j-raedler.de](mailto:joerg@j-raedler.de)).

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

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

The code is released under the terms of the [BSD License](#), which allows the free usage, distribution and enhancement of this code. Feel free to send your contributions.

## 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 for 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.

**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 doesn't 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 separated 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

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 -i myfile.mat
```

List all variables in short form:

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

List all variables including block number and description:

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

Show a simple tree view of the variables:

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

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

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

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

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

Get a list of supported formats:

```
$ DyMatExport -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 -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

[... documentation will follow soon, read the source ...]