Contents

About this file

Nearly everything you might need to know for using plotXY satisfactorily is introduced in PlotXY tutorial.
This document is a reference document, to understand more in depth how different file formats are managed and how different names are converted from one format to another.
This might be useful in case you have troubles with one specific file format.

Types of pl4 files and issues with pl4 format

Originally, PlotXY was created to plot data created using the well-known simulation program Alternative Transients Program (ATP). The binary output of ATP is written in files that traditionally have as extension “atp”.

PlotXY was originally structured to be compatible to all versions of the pl4 files. Unfortunately, over the years, modifications on how pl4 files are created have ben introduces, and the modifications are not well documented. In practices there are cases for which nearly nobody is able to tell what the actual file structure is.

Presently (2014) there is the following situation:

- Pl4 files created earlier than 2003 should be read satisfactorily: however, the correct identification of power and energies might be impossible because of the limited information stored on files
- Pl4 files created using the following settings NEWPL4=2 or NOPISA=0 should be correctly read and interpreted.

Settings such as NEWPL4 or NOPISA are in an ATP file called STARTUP

Some old files containing frequency-scan runs, created using NEWPL4=2, do not correspond to the file format specification, and therefore cannot be correctly read by PlotXY.
It is not clear whether this issue is still present in the ATP code. In case you have troubles with a pl4 file coded with the NEWPL4=2 (or NOPISA=0) directive, contact m.ceraolo@ing.unipi.it. An attempt will be made to contact ATP programmers to try to solve the issue.

To verify how PlotXY is able to read atp binary output files the following small testfiles are supplied:
- file **rad2.pl4** that is created using the older format. This is acknowledged by PlotXY that issues a warning message only once per session whenever such very old files are loaded

- file **type2.pl4**. This is created using NEWPL4=2. It is perfectly interpreted by PlotXY

The "adf" file structure definition

In addition to pl4 files, the program is able to read generic Ascii Data Files, whose extension must be ADF.
This is particularly useful for comparing ATP simulation outputs with lab results or other programs' outputs.

For the maximum flexibility of this feature, a very simple file structure is required. The file must be composed by:
Header. It is constituted by two lines.

The first header line has the following format:
[*step* [*x_variable name*]] [*comment*]
in which fields between squares [] are optional[1], and the components can be separated by one or more blanks.
Description of the meaning of the components:

- *step* is a number;
  When this field is specified, it is taken as a constant step for building the x-axis variable: a variable is automatically generated having values 0, *step*, 2* *step,* 3**step,* ... . This is a useful option to reduce the ADF file size and interpretation time when a constant time-step is involved.
  If no step is specified, the first variable present in the body of the file is intended to be the default x-axis.
- *x_variable name* is a string representing the name of the automatically generated x-axis variable when *step* is specified. If this name begins with the character 't', it is interpreted as the time, expressed in seconds, by the automatic labelling algorithm of the program[2]. If *x_variable name* is missing, the name "`X_(auto)`" is assumed for the automatically generated x-axis variable.
- *comment* is a comment string the first characters of which are two slashes[3].

The second header line will contain the names of the variables, separated by spaces and/or tab characters[4].

Body. It is composed by values of the variables, in written ASCII, in the form of a matrix (a column for each variable). This way each row refers to a particular x-axis (normally time) value.
If no step is specified in the first header line, the first variable in the body of the file is intended to be the default x-axis. It is therefore required that the corresponding values be monotonically increasing; they are NOT required to be equally spaced (i.e., variable-sampling x-axis data are allowed). The numbers in a row are to be separated by spaces and/or tab characters[10]. The decimal separator **must be** "." no matter of the value of it set in the OS (in Windows: Control Panel "International" section).

Two simple examples will clarify:

```
******** FILE SAMPLE1.ADF ********


time sin(W*t) sin(W*t-240°) sin(Om*t+240^)
0.00000e+00 0.00000e+00 8.66022e-01 -8.66022e-01
5.00000e-04 1.56429e-01 7.77145e-01 -9.33577e-01
1.00000e-03 3.09008e-01 6.69133e-01 -9.78144e-01
1.50000e-03 4.53978e-01 5.44646e-01 -9.98628e-01
2.00000e-03 5.87770e-01 4.06749e-01 -9.94524e-01
2.50000e-03 7.07090e-01 2.58836e-01 -9.65933e-01
3.00000e-03 8.09000e-01 1.04551e-01 -9.13558e-01
3.50000e-03 8.90991e-01 -5.23084e-02 -8.38690e-01
```

```
******** FILE SAMPLE2.ADF ********
5.e-4 time //this is a comment
sin(W*t) sin(W*t-240°) sin(W*t+240^)
0.00000e+00 8.66022e-01 -8.66022e-01
1.56429e-01 7.77145e-01 -9.33577e-01
3.09008e-01 6.69133e-01 -9.78144e-01
4.53978e-01 5.44646e-01 -9.98628e-01
5.87770e-01 4.06749e-01 -9.94524e-01
7.07090e-01 2.58836e-01 -9.65933e-01
8.09000e-01 1.04551e-01 -9.13558e-01
8.90991e-01 -5.23084e-02 -8.38690e-01
```

1 therefore *step* can be present or missing; if present can be followed by *x_variable name*; *comment* can be missing. The row can even be empty.
2 Also variables beginning with "%t" are considered as being time. This has been added for compatibility with MATLAB M-files.
3 Comments can also begin with the character '%'. This has been added for compatibility with MATLAB M-files.
4 The program can accept separators between names and numbers containing commas (','). To obtain this, check the "Commas are separators in ADF files" option in the **input** section of the **Program Options** dialog box.

To verify how PlotXY is able to read atp binary output files the following small testfiles are supplied:

- file **sample1.adf** a simple version with no automatic time nor any comment
- file **sample2.adf** a version containing automatic time generation and a comment
- file **sample3.adf** a further version with a comment but without automatic time generation

Reading "csv" files

PlotXY reads csv files having the following format:

The first row containing, separated by commas, the names of the variables
All the other rows contain numerical values, separated by commas.

In practice for PlotXY a csv file is an "adf" file written with the option "commas are separators" active, and from which the first row has been deleted.
The user, to keep displayed names shorter, can choose in "Program Options" window to select the option "Trim quotes at beginning and end of CSV names".

If this is done, for instance
"time"
will be displayed as
time

An example of valid csv file is supplied in the data folder, whose name is "bounging.csv".

This format is very important for those that use OpenModelica as a simulation environment, since it is compatible with the csv output from OpenModelica.

The Comtrade file format

PlotXY has some capability to read Comtrade-formatted files.
The Comtrade interpretation routine has been creating according to the instructions available in the following document:

IEEE Std C37.111-1999 "IEEE Standard Common Format for transient Data Exchange (CMTRADE) for Power Systems.

Please note that:

- the quoted document describes two different types of comtrade files: ASCII and Binary data files. PlotXY is able to read both of them, although only some features have been actually tested;
- After I wrote the Comtrade interpretation code I could just made a few simple tests on comtrade files. I cannot be sure that it works correctly in all cases.

In case you have a Comtrade file that PlotXY is not able to read satisfactorily, you are recommended to send a small report on this at the address: m.ceraolo@ing.unipi.it. Don't forget to describe exactly what the problem is and to enclose a small data case causing the trouble remember that Comtrade format defines two different files for each data set: one having ".CFG" as extension, and another, with the same name having extension ".DAT".

To verify how PlotXY works along with Comtrade sets, you can load any of the supplied files:

- **170900.CFG** (loads 170900 that is ASCII Comtrade)
- **Binary.CFG** (loads Binary.DAT that is a binary Comtrade data file)

The LVM file format

PlotXY allows reading data also from LabView measurement files. The file description used to create dthe interpreter is the document
Specification for the LabVIEW Measurement file (.lvm) from National Instruments. (Document Type: Tutorial
NI Supported: Yes; Publish Date: Jul 07, 2010)

Please note that the interpreter has been checked only in a limited number of cases, and therefore might not work properly when asked to interpret a proper lvm file.

In case you have a LVM file that PlotXY is not able to read satisfactorily, you are recommended to send a small report on this at the address: m.ceraolo@ing.unipi.it. Don't forget to describe exactly what the problem is and to enclose a small data case causing the trouble.

To verify how PlotXY works along with LVM files, you can load any of the following:

- **oneX.LVM** (contains a single X, i.e. a unique column for time)
- **multiX.LVM** (is a multiple- X file, i.e. each signals has its own time)

Matlab format info

PlotXY can be useful for viewing MATLAB/SIMULINK outputs in some cases, e.g.:

- when one wants to share his outputs with someone that does not have a Matlab copy of his own
- when one wants to effectively export plots into other programs by means of the Windows Clipboard. In fact PlotXY have, with this respect, important advantages over MATLAB:
  - it automatically eliminates visually-redundant points. In normal use this can result in output plot size reductions by factors of 100 and over
  - if the plot is zoomed PlotXY export just the useful part of the plots, while MATLAB exports also all the plotting points outside the zoom window.

Matlab files of the old type V4 can be managed by the program effectively. Newer versions are read, under some constraints. The most limiting one is that the matlab file must not be compressed.
This limitation is programmed to be be dropped in the future.
Note matlab programs allow to write backward-compatible .mat files, for instance using the "-V4" flag in the save format.

Different kinds of arrays can be stored in a Matlab file: matrices of integers, floats, text strings.
However PlotXY is intended for dealing with files containing output of either measures or simulations. Therefore the files managed by the program must always contain a "time" (i.e., a quantity monotonically increasing) and some "signals" associated to that time (therefore all the signals should have a number of point equal to that of time).
As a consequence of these characteristics, to be understood by the program, a mat file has to comply with some specific requirements:
- it must contain only matrices of floats sharing the same number of rows
- it has to contain a "time" variable. If there is in the file a single column variable having as name "t" it is assumed to be time. Otherwise, the first column of the first variable in the file is assumed to be time.

If some of the variables in the file has multiple columns it stores multiple signals. In this case the program adds a unique suffix to the different columns, constituted by a progressive number between parentheses.

NOTE. The best way to exploit PlotXY as a SIMULINK graphical post-processor is as follows:

- use for any variable sent to workspace the same value of the "Decimation" parameter
- send data to workspace using "To workspace" blocks instead of the "Save data to workspace" feature of Scopes (this way replications of time vectors is avoided);
- send the simulation time to workspace just once by means of the "Save to workspace" feature of the Simulation|Parameters|Workspace I/O dialogue box.

To verify how PlotXY is able to read mat binary output files the following small testfiles are supplied:

- file **rad2.mat** obtained starting from the file (also supplied) rad2.pl4), using the naming conversion rues discussed in the section of this document immediately below here; This file is written in matlab binary file version V4
- file **V6rad2.mat** containing the same content but written in Matlab binary form V6 (uncompressed)

Naming conversion rules from PL4 to ADF and MAT

ADF and MATLAB names are generated by pl42mat according to the following rules:

0. Time is simply indicated as "**t**"
1. All the .PL4 node names and TACS, MODELS, Universal Machine and Synchronous Machine variable names are converted into lowercase, except the first characters that are converted into uppercase
2. If the names contain dashes or plus signs or embedded blanks, they are converted into underscores ('_'); blanks at beginning or end of names are discarded

3.

- The names for node voltages are obtained adding at the beginning of the node names as they are after step 2 the character '**v**'
- The names for branch voltages and currents are obtained combining the two node names into a unique name and then adding at the beginning of the resulting string the character '**v**' or '**i**' respectively; if one of the nodes is " ", it is changed into "**Terra** "
- Names of TACS or MODELS variables are obtained adding at the beginning of the names as they are after step 1 the characters '**t**' or '**m**' respectively"
- Names of Universal Machine and Synchronous Machine variables are obtained adding at the beginning of the names as they are after step 1 the characters '**u**' or '**s**' respectively, followed by one or two digit(s) indicating the corresponding machine number (but up to 9 SMs are supported).

Examples:
(from DC5)
Node of Voltage **' GENT ': vGent**
Voltage Difference between **'TRANFF'** and **'OPEN ': iTranffOpen**
Current between **'GEN '** and **' GENT ': iGenGent**
Current between **'LOAD '** and **' ': iLoad**
(from DCn12)
UM variable **'UM-1 ' - 'TQGEN': u1Tqgen**
(other)
Voltage Difference between **'UMPOS '** and **' ': vUmposTerra**
Voltage Difference between **' '** and **'UMNEG ': vTerraUmneg**
SM variable **'MACH 1' - 'ID ': s1Id**
SM variable **'MACH 1' - 'TQ GEN': s1Tq_gen**
SM variable **'MACH 1' - 'ANG 1 ': s1Ang_1**
SM variable **'MACH 2' - 'ANG 1 ': s2Ang_1**
MODELS variable **'MODELS' - 'SOC ': mSoc**
MODELS variable **'MODELS' - 'Iw ':**