MELCOR post-processing using open source tools

Petr Vokáč, vok@ujv.cz

NRI Řež plc

3$^{\text{nd}}$ EMUG meeting, 11/12 April 2011, Bologna

Overview

- Motivation
- Overview of software used
- Presentation of tools:
  - `readptf` FORTRAN utility
  - GNUPlot (2D value vs. time plots)
    - use of `readptf` and GNUPlot
    - `browseptf` Python GUI for GNUPlot
    - use of Python scripts with GNUPlot
  - PyX (state snapshots)
    - Python scripts for creating PyX charts
    - Example of PyX output
    - Animations
- Documentation, Installation, Configuration
- Content of CD

This presentation is an update of a talk given at CSARP/MCAP 2009.

Motivation

- I am using Linux (64-bit)
- There is no post-processing tool for Linux in the MELCOR 1.8.6 distribution
- HISPLT tool from MELCOR 1.8.5 can be used, however :
  - it is difficult to compile
    (we have 32-bit executable once compiled with Lahey Fortran,
    attempt to compile HISPLT with Intel Fortan failed . . .)
  - it is difficult to get the data out of HISPLT
    (the “`datout`” file has to be converted for using the data with other programs)
- MELCOR plotfile is binary compatible for Linux and Windows, therefore I can use post-processing tools distributed for Windows, but . . .

⇒ decided to develop tools for MELCOR post-processing with following requirements:

- to be simple
- to use existing proven applications and OS capabilities to limit my own programming effort
- separate data processing and graphical tools
- output should be portable vector graphics (eps, **pdf**, emf, svg, . . .)

Overview of software used

- Operating systems:
  - Gentoo Linux (www.gentoo.org) + Gnome
    - a free operating system based on either Linux or FreeBSD
    - Portage — package building and installation system. It provides information of version dependencies of applications ported for the system.
    - Linux is “mostly” POSIX-compliant system (see http://en.wikipedia.org/wiki/POSIX)
  - Mac OS X + Darwin (www.macports.org)
    - MacPorts is based on FreeBSD
    - Portfile — MacPorts equivalent of Gentoo portage
    - Mac OS X is fully POSIX-compliant system
- Applications:
  - GNUPlot - GNU plotting tool, www.gnuplot.info
  - PdfTK - the pdf toolkit, www.pdflabs.com
  - Xpdf - A PDF Viewer for X, www.foolabs.com/xpdf/home.html
  - Python 2.x, www.python.org, with additional modules:
    - PyGTK - python interface to GTK+, www.pygtk.org
    - PyX - Python graphics package, pyx.sourceforge.net (requires TeX/LaTeX)
    - SciPy - Scientific Tools for Python, www.scipy.org
  - LaTeX - A document preparation system, www.latex-project.org, www.tug.org

readptf FORTRAN utility

- Command line utility
  - written in FORTRAN, using parts of the MELCOR source code
  - it allows us to retrieve the list of plot variables, various nodalization information, data for each plot variable
  - simple use to get the data as a text in a space separated columns format:

    ```
    readptf.exe MELPTF variable > text output
    ```
  - text output can be directly used by other postprocessing programs such as GNUPlot, SCILab, AcGrace, . . . or even MS Excel.
  - using a `pipe`, there is no need to store data in temporary text file,
    e.g. in GNUPlot:

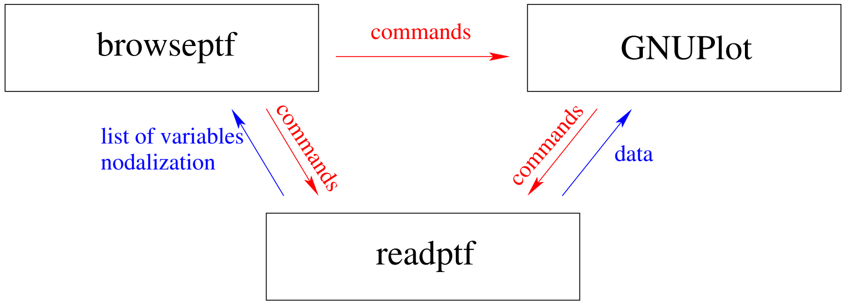
    ```
    plot "< readptf.exe MELPTF CVH-P" using 1:3 title "cv020" with lines
    ```

    or

    ```
    plot "< readptf.exe MELPTF CVH-P.020" using 1:2 title "cv020" with lines
    ```
- Portability: tested with Intel Fortran and g95 under Linux, Mac OS X and with g95 under Windows
  - an executable created with g95 has much worse performance on Linux or Mac, because it does not use the disk cache to store MELPTF for subsequent run
  - the code can not be compiled with gfortran
- Status: final version, no further development expected
  (except for an improvement of portability and elimination of possible bugs)

browseptf Python GUI

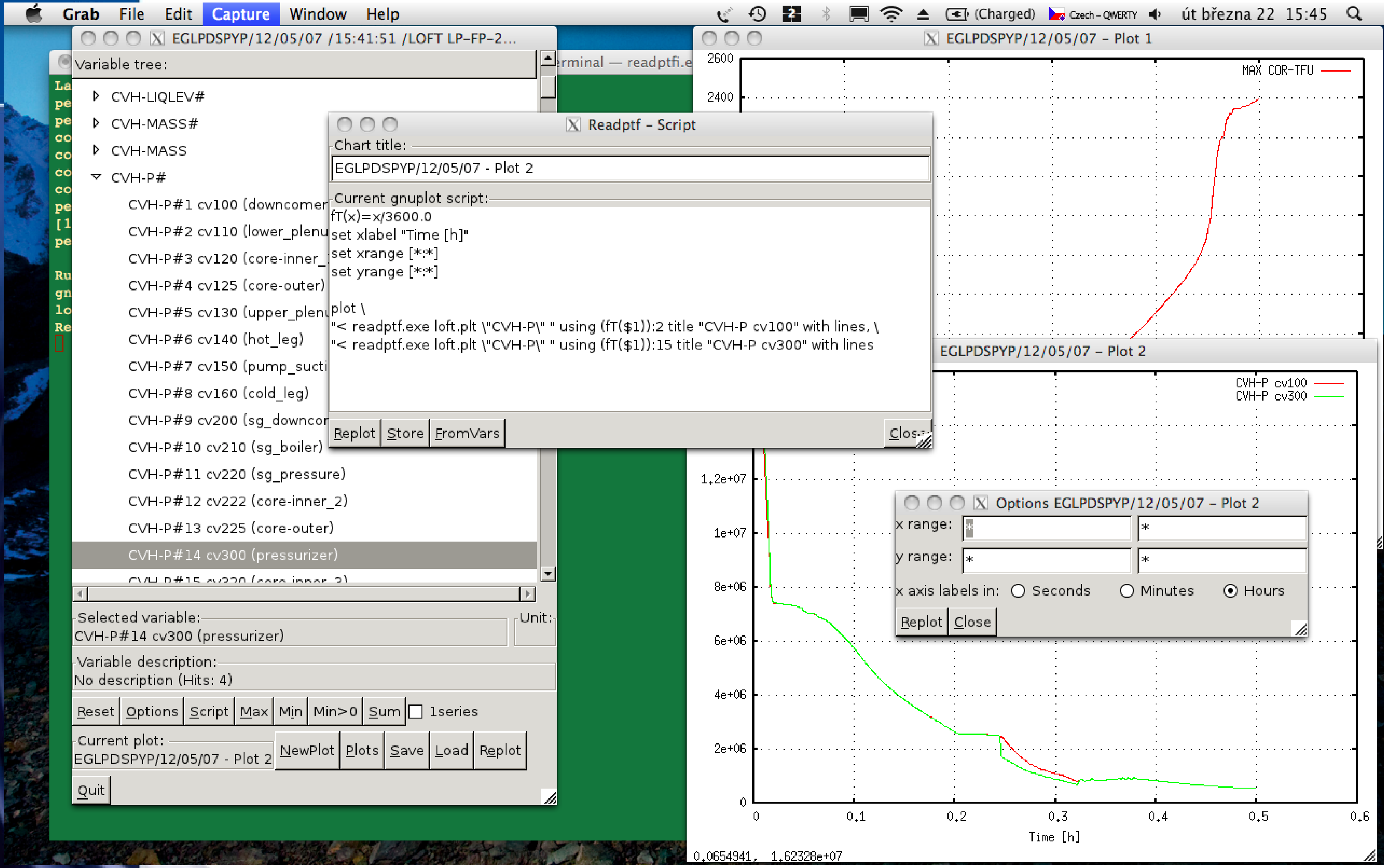
- Simple GUI to allow quick look at the calculation results:
  - PyGTK library is used for GUI
  - pipes are used for communication between programs,
    data flow scheme :



  - it is possible to store GNUPlot scripts via clipboard (it was the second main objective for programming this GUI — to speed-up preparation of scripts for GNUPlot)
  - GNUPlot script can be changed
  - it is possible to save the configuration to replot the same charts again
- Limitations:
  - only one MELCOR plotfile
    (through GUI, but the GNUPlot script can be changed to read data from another output)
- Portability: Linux, Mac and, with recent GNUPlot version, also Windows
- Status: final version, no further development of the code expected, except for an improvement of portability and elimination of possible bugs.

  Description of indexes is missing for many variables and it is being added step by step.

browseptf - example screenshot



Grab File Edit Capture Window Help
(Charged) Czech - QWERTY út března 22 15:45
erminal — readptfi.e
Reset Options Script Max Min Min>0 Sum
NewPlot Plots Save Load Replot
Quit
Replot Store FromVars Close
Replot Close
MAX COR-TFU
2600
2400
CVH-P cv100
CVH-P cv300
1.2e+07
1e+07
8e+06
6e+06
4e+06
2e+06
0
0 0.1 0.2 0.3 0.4 0.5 0.6
Time [h]
0.0654941, 1.62328e+07

EGLPDSPYP/12/05/07 /15:41:51 /LOFT LP-FP-2...

Variable tree:

- CVH-LIQLEV#
- CVH-MASS#
- CVH-MASS
- CVH-P#
  - CVH-P#1 cv100 (downcomer
  - CVH-P#2 cv110 (lower_plenu
  - CVH-P#3 cv120 (core-inner_
  - CVH-P#4 cv125 (core-outer)
  - CVH-P#5 cv130 (upper_plenu
  - CVH-P#6 cv140 (hot_leg)
  - CVH-P#7 cv150 (pump_sucti
  - CVH-P#8 cv160 (cold_leg)
  - CVH-P#9 cv200 (sg_downcor
  - CVH-P#10 cv210 (sg_boiler)
  - CVH-P#11 cv220 (sg_pressure)
  - CVH-P#12 cv222 (core-inner_2)
  - CVH-P#13 cv225 (core-outer)
  - CVH-P#14 cv300 (pressurizer)
  - CVH-P#15 cv320 (core-inner_3)

Selected variable: CVH-P#14 cv300 (pressurizer)

Unit:

Variable description: No description (Hits: 4)

☐ 1series

Current plot: EGLPDSPYP/12/05/07 - Plot 2

Readptf – Script

Chart title: EGLPDSPYP/12/05/07 - Plot 2

Current gnuplot script:

```
fT(x)=x/3600.0
set xlabel "Time [h]"
set xrange [*:*]
set yrange [*:*]

plot \
"< readptf.exe loft.plt \"CVH-P\" " using (fT($1)):2 title "CVH-P cv100" with lines, \
"< readptf.exe loft.plt \"CVH-P\" " using (fT($1)):15 title "CVH-P cv300" with lines
```

EGLPDSPYP/12/05/07 – Plot 1

EGLPDSPYP/12/05/07 – Plot 2

Options EGLPDSPYP/12/05/07 – Plot 2

x range: * *

y range: * *

x axis labels in: ○ Seconds ○ Minutes ● Hours

Python and GNUPlot for more complex data evaluation

- Concept:
  - retrieve data with `readptf`
  - analyze data in Python
  - plot results of analysis with GNUPlot
- Approach:
  - create Python module to standardize data retrieval and repeatedly used evaluations (`pvmisc.py`)
  - write simple Python scripts callable from GNUPlot
- Examples:
  - calculate sum, max, . . . for parts of the core
    (e.g.: VVER-440: lower plenum — followers region — main core region)
  - calculate total mass of RN deposited on HSs and contained in CVH fluids for specified set of HSs and CVHs
  - compare calculation results with experimental data
  - . . .

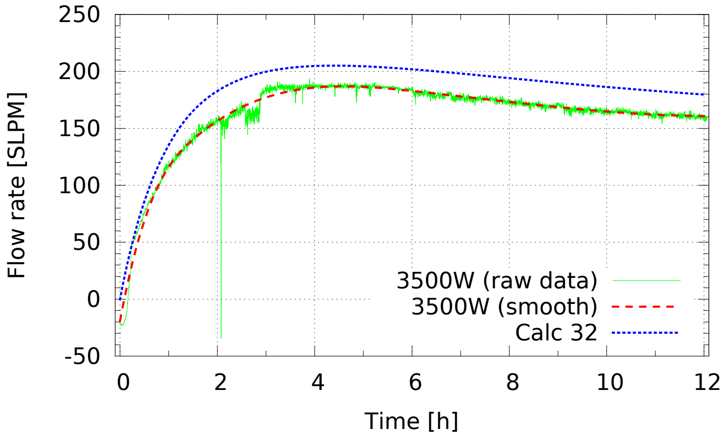
Python and GNUPlot for more complex data evaluation - Example

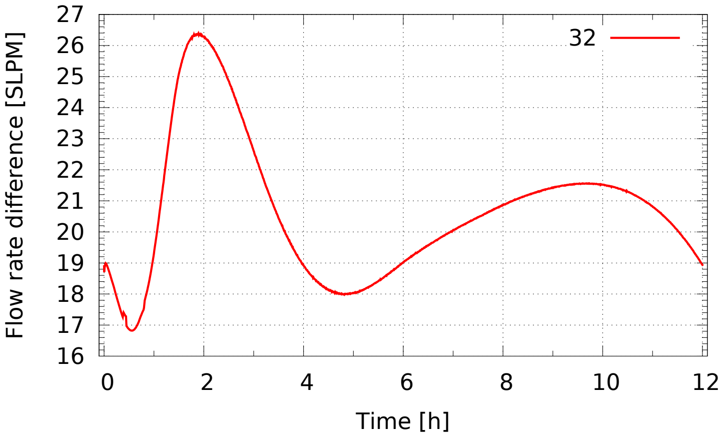
Listing of `fl-mflow.py`

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys
import scipy.interpolate as scii
import pvmisc
import sfplib
# read commandline arguments
sptf = sys.argv[1] # melcor plot file name
sexp = sys.argv[2] # experiment "3500W_Pre-Ignition_10-18-10"
# read experimental data
lt,ly = sfplib.fTSI_HotWire_1("../data01/txt/",sexp)
# read melcor results (columns in lists including time)
lvm=pvmisc.fReadVarC(2,"FL-MFLOW.001",0,sptf,"1")
# recalculate mass flow rate to std. liter per min
sfplib.fSlpml(lvm[1])
# interpolate experimental data to melcor times
spline = scii.UnivariateSpline(lt, ly, k=3, s=float(len(ly))*30.0)
ley = spline(lvm[0])
# output results : time - flowrate melcor - flowrate exp. - difference
for i in range(len(lvm[0])) :
 print "%f %f %f %f" % (lvm[0][i],lvm[1][i],ley[i],lvm[1][i]-ley[i])
pass
```

Listing of `fl-mflow32.gpl`

```
load "settings.inc"

system "fl-mflow.py melptf32 3500W_Pre-Ignition_10-18-10 > ../txt/fl-mflow-3-2.txt"

set xrange [-0.1:12.1]
set key bottom right

set ylabel "Flow rate [SLPM]"
sout="fl-mflow-slpm-32-01"
set output fOut(sout)
plot \
"< TSIHotWire1.py 3500W_Pre-Ignition_10-18-10 " using (fT($1)):($2) title "3500W (raw data)" with lines lw 1 lc 2, \
"../txt/fl-mflow-3-2.txt" using (fT($1)):3 title "3500W (smooth)" with lines lw 4 lc 1, \
"../txt/fl-mflow-3-2.txt" using (fT($1)):2 title "Calc 32" with lines lw 4

set key top right
set ylabel "Flow rate difference [SLPM]"
sout="fl-mflow-slpm-32-02"
set output fOut(sout)
plot \
"../txt/fl-mflow-3-2.txt" using (fT($1)):4 title "32" with lines lw 4

system "rm ../txt/fl-mflow-3-2.txt"
```





some hints to use GNUPlot

- Easy change of graphical output format (terminal in GNUPlot terminology)

  MS Word users can choose EMF terminal and link or include *.emf files to their document:

```
set terminal emf color dashed font "Times-Roman,12"
set output "test.emf"
# ...
```

- Single pdf file with all the figures can be simply created, e.g.:

```
set terminal pdf font "Times-Roman,8" dashed enhanced
#...
# plot figures to hs-temp32*.pdf in ../pdf1
# ...
unset output
system "pdftk ../pdf1/hs-temp32*.pdf output ../pdf/hs-temp32.pdf"
system "echo \"Figures joined to ../pdf/hs-temp32.pdf\""
```

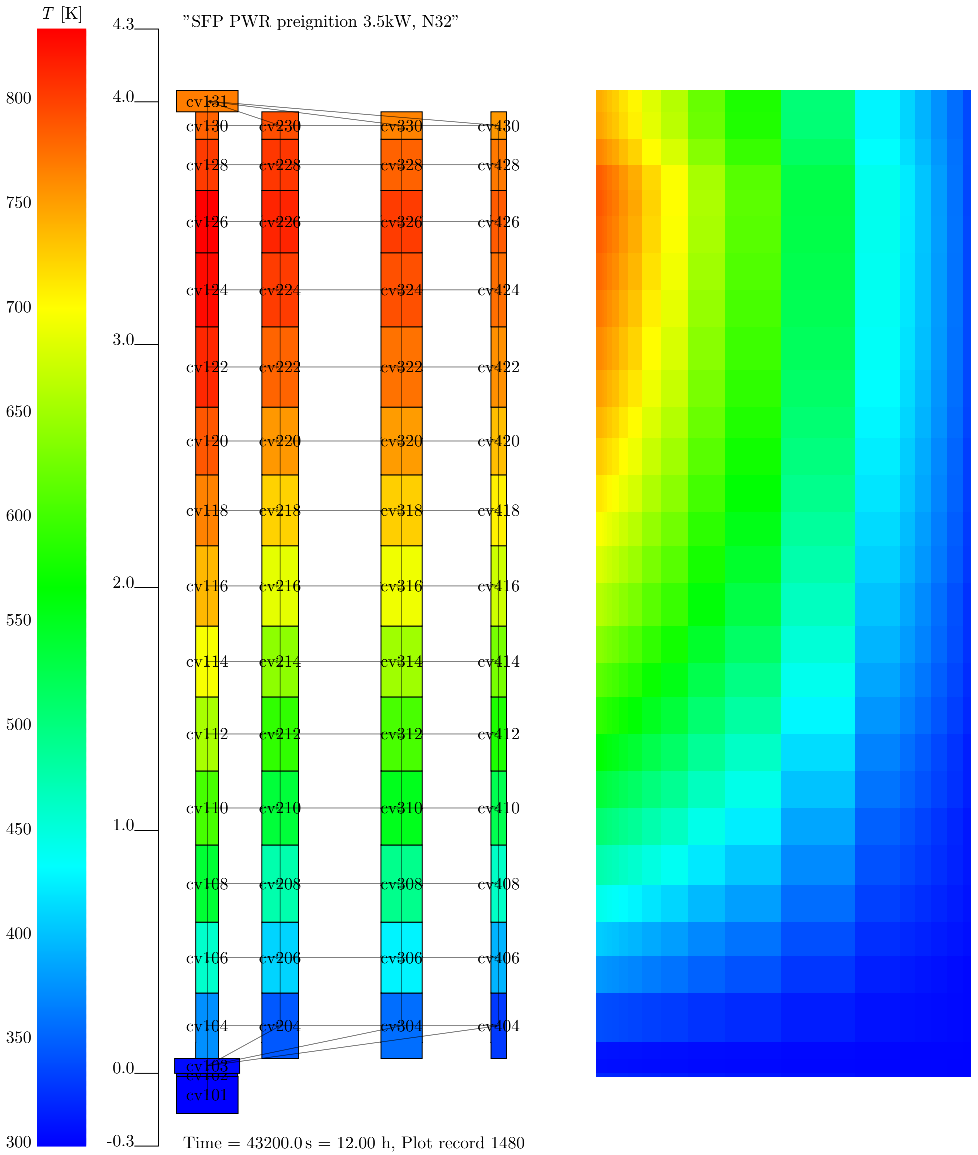
State snapshots using PyX

- Common configuration file format for all the plots
- Frequently used function available in modules (`pvmisc.py, pvpyx.py, tzmod.py`)
- Currently available plots:
  - core volume fractions (`cor-volf.py, colors.py`)
  - temperatures of lower head (`cor-tlh.py`)
  - temperature vs. elevation charts (`cor-tz.py`)
  - liquid levels in CVH volumes (`cvh-liql.py`)
  - temperatures of gas in CVH and temperatures in HS (`cvh-t.py`)
- Tested with:
  - VVER-440/213 with input models for: full power, IVR and shutdown
  - Grand Gulf sample input
  - TMI2 sample input
  - LOFT-LP-FP2 sample input
  - OECD-SFP simulations
- Animation

  Script `animpdf.py` implements simple GUI to :
  - navigate simultaneously in several pdf documents
  - run “slideshow”

  using Xpdf.

State snapshots using PyX - cvh-t example for OECD-SFP



Documentation, Installation, Configuration

- Documentation
  - `readptf` — run `readptf.exe` without any argument to see options or read the source
  - Documentation of Python scripts and modules was created using pydoc. See `index.html` in the `meltools` folder.
- Installation of required libraries and applications
  - Linux

    use appropriate installation tool for your Linux distribution
    (usually it takes care about version dependencies)
  - Mac

    use MacPorts;
    in `/usr/bin` make a backup copy of Apple's `python` and create link:
    `python` → `/opt/local/bin/python`
    (otherwise Apple's `python` is run instead of that installed from macports and libraries are not found)
  - Windows

    all the libraries and tools should be downloaded and installed individually — but versions should be compatible
- Configuration of "meltools" :
  - `readptf.exe` and "executable" Python scripts should be on system `PATH` variable
  - Python modules should be on system `PYTHONPATH` variable

Content of CD

- 🗀 meltools
  - 🗀 readptf — readptf source, makefile and executable for Linux, Mac and Windows
  - 🗀 browseptf — browseptf.py script and support modules
  - 🗀 pyc — library modules: pvmisc.py, pypyx.py, . . .
  - 🗀 melpyx — scripts for generation of snapshots using PyX
- 🗀 melpytests
  - 🗀 GrandGulf
  - 🗀 loft-lp-fp2
  - 🗀 tmi2
- 🗀 Windows — installation of python, pygtk and gnuplot for win32

I would like to invite MELCOR users to use presented post-processing tools and participate in their further development

Thank you for your attention