



**HAL**  
open science

## Post-processing: coming next in SALOME

C. Aguerre, A. Bruneton, C. Genrault

► **To cite this version:**

C. Aguerre, A. Bruneton, C. Genrault. Post-processing: coming next in SALOME. SALOME User Day 2016/ EDF Lab Paris-Saclay, Dec 2016, Palaiseau, France. cea-02435103

**HAL Id: cea-02435103**

**<https://cea.hal.science/cea-02435103>**

Submitted on 10 Jan 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



# Post-processing: coming next in SALOME

Cédric Aguerre – EDF Lab Paris-Saclay  
Adrien Bruneton – CEA Saclay  
Clarisse Genrault – CEA Saclay

SALOME User Day 2016  
EDF Lab Paris-Saclay

December 9<sup>th</sup> 2016



# AGENDA

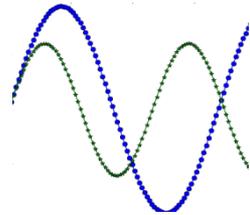
## 1. MEDCALC

A FIRST LEVEL ACCESS TO PROCESSING AND VISUALIZATION FEATURES OF MESHES AND FIELDS IN SALOME



## 2. CURVEPLOT

HIGH LEVEL MATPLOTLIB ACCESS INTO SALOME



# MEDCALC



# MEDCALC? A WORK IN PROGRESS



- **Load, process and visualize meshes and fields from the same SALOME module**
  - Integrate PARAVIS viewers w/o exposing the full ParaView GUI
  - Take advantage of the power of MEDCoupling and PARAVIS combination
  - Provides arithmetic functions on fields



- **« Easy to use »**
  - GUI exposing essential functions with PARAVIS look and feel
  - A dedicated Python API: you can dump!
- **Finally, need advanced operations?**
  - Switch to PARAVIS in a second!
  - The pipeline is already there!



# MEDCALC: GUI OVERVIEW



The screenshot displays the SALOME 8.1.0 interface. The top menu bar includes File, Edit, View, Presentations, Processing, Tools, Window, Writers, CoProcessing, and Help. The toolbar contains various icons for file operations and simulation controls. The Object Browser on the left shows a tree structure with objects like PRESSION\_ELEM\_DOM, VITESSE\_ELEM\_DOM, and Slices (24). The Workspace on the left shows a table with columns for Name and Value. The Presentation parameters section for Slices (24) shows settings for Range, Color map, Displayed component, Number of slices (max), and Slice orientation. The Python Console at the bottom shows a script for updating slices. The main 3D view shows a cylinder mesh with a color map and a slice orientation.

Object Browser

- PRESSION\_ELEM\_DOM (ON\_CE)
  - Scalar map (7)
- VITESSE\_ELEM\_DOM (ON\_CELL)
  - Scalar map (8)
- VITESSE\_SOM\_DOM (ON\_NODE)
  - Mesh view (4)
- INTERFACES
- Demo\_2.med
- MAIL
- MODES\_DEPL
  - Scalar map (23)
  - Slices (24)

Workspace

Name	Value
Mesh_qu...	No value
Mesh...	No value
it ...	No value

Presentation parameters

Slices (24)

Range: All timesteps

Color map: Blue to red rainbow

Displayed component: Euclidean norm

Number of slices (max): 5

Slice orientation: Normal to Z

Python Console

```
24
>>> params = medcalc.GetSlicesParameters(24)
>>> params.nbSlices = 5
>>> medcalc.UpdateSlices(24, params)
>>> params = medcalc.GetSlicesParameters(24)
>>> params.orientation = MEDCALC.SLICE_NORMAL_TO_Z
>>> medcalc.UpdateSlices(24, params)
>>>
```

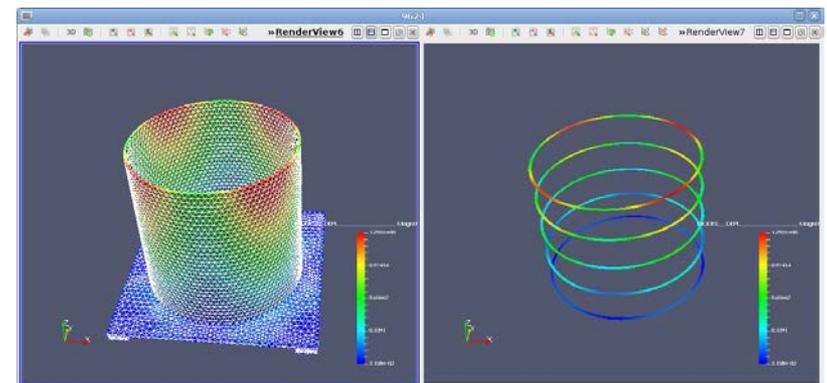
# MEDCALC: PYTHON SAMPLE



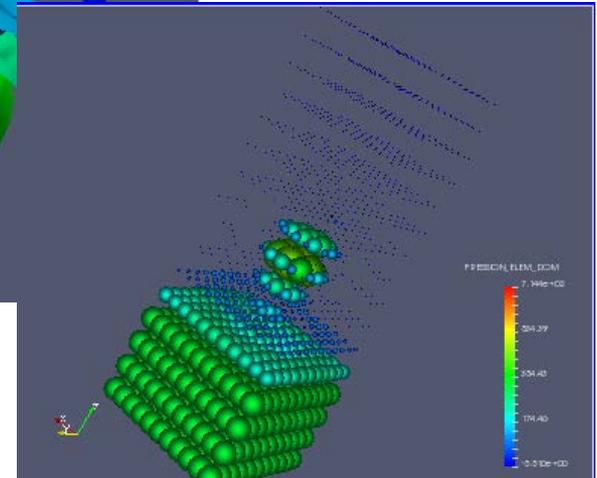
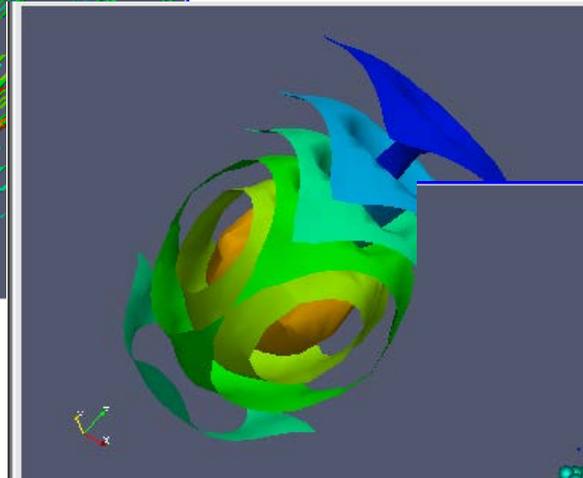
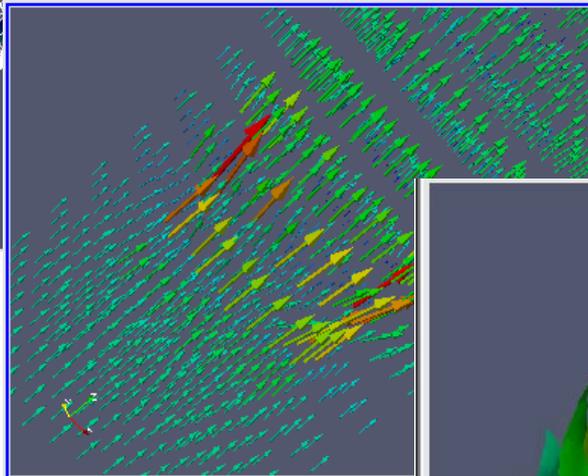
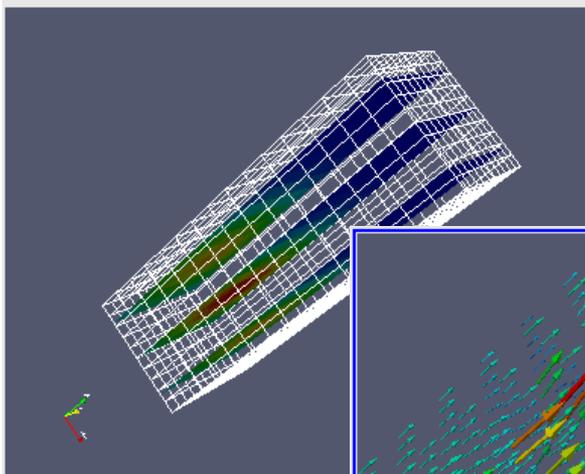
```
>>> source_id = medcalc.LoadDataSource('/local00/home/F62173/testfiles/Demo_2.med')
>>> presentation_id = medcalc.MakeMeshView(medcalc.GetFirstMeshFromDataSource(source_id),
                                           viewMode=MEDCALC.VIEW_MODE_NEW_LAYOUT)
>>> presentation_id = medcalc.MakeScalarMap(accessField(80),
                                           viewMode=MEDCALC.VIEW_MODE_OVERLAP)
>>> presentation_id = medcalc.MakeSlices(accessField(80),
                                         viewMode=MEDCALC.VIEW_MODE_SPLIT_VIEW)
>>> params = medcalc.GetSlicesParameters(presentation_id)
>>> params.orientation = MEDCALC.SLICE_NORMAL_TO_Z
>>> params.nbSlices = 5
>>> medcalc.UpdateSlices(22, params)
```



Each GUI action triggers  
a python function call



# MEDCALC: PRESENTATIONS



# MEDCALC! THERE IS WORK TO BE DONE!



- **Still a prototype to be beta-tested (SALOME 8.2)**

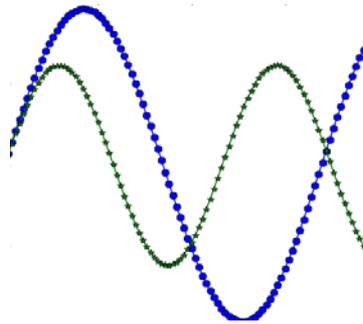
- Presentations with minimal settings
- Modes and animations
- Memory-passed objects
- ParaView pipeline – switch to PARAVIS
- GUI and scripting

- **Roadmap 2017**

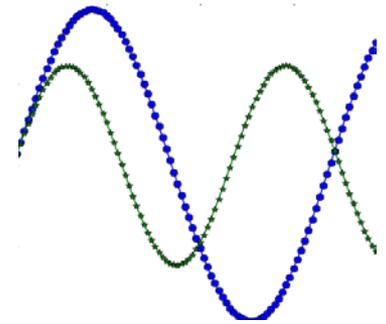
- Concept validation thanks to user feedback
- Dataspace / workspace interaction
- MEDCoupling visibility enhancement



# CURVEPLOT



# CURVEPLOT

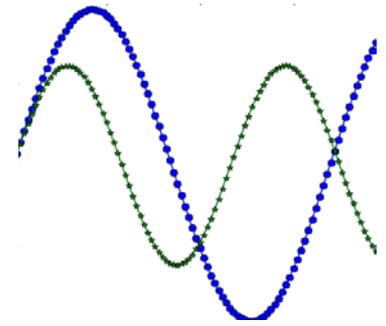


## Some history and background

- At CEA, several business projects use the “old” C++ Plot2D, based on Qwt
- Problem: some divergence/forks in the code!
  - New/modified functionalities/API
  - Not reversed in the main stream
- In general, current tool is not obvious to maintain and to change
- Hazardous mix between Qwt API and pure PLOT2D API
- From recent SALOME versions, we benefit from the *de-facto* integration of *matplotlib*
  - *matplotlib* provides an easy to manipulate API and most wanted services

**matplotlib**

# SPECIFICATIONS

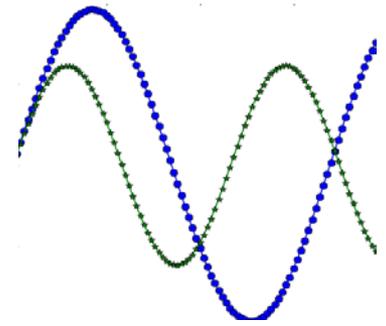


## Main needs around a plotting tool

- Obviously, plotting 2D curves!
  
- No 3D: ParaVis is already there! And MED visualization is almost ready too.
  
- Curve related functionalities
  - Overlapping curves
  - Various representation modes (log, color, etc ...)
  
- Why not use Paraview's capabilities ?
  - Not user-friendly enough for business module developer
  - Too heavy: requires ParaView loading!
  - Harder to interface with custom file formats (XML, .data, ...)

Start minimal and grow only if needed! (similar to MEDCalc)

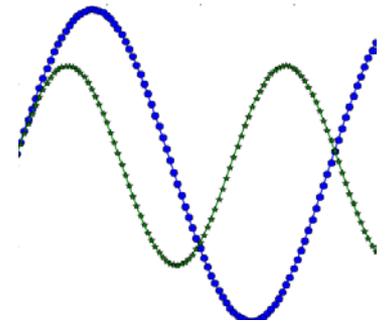
# IMPLEMENTATION & API



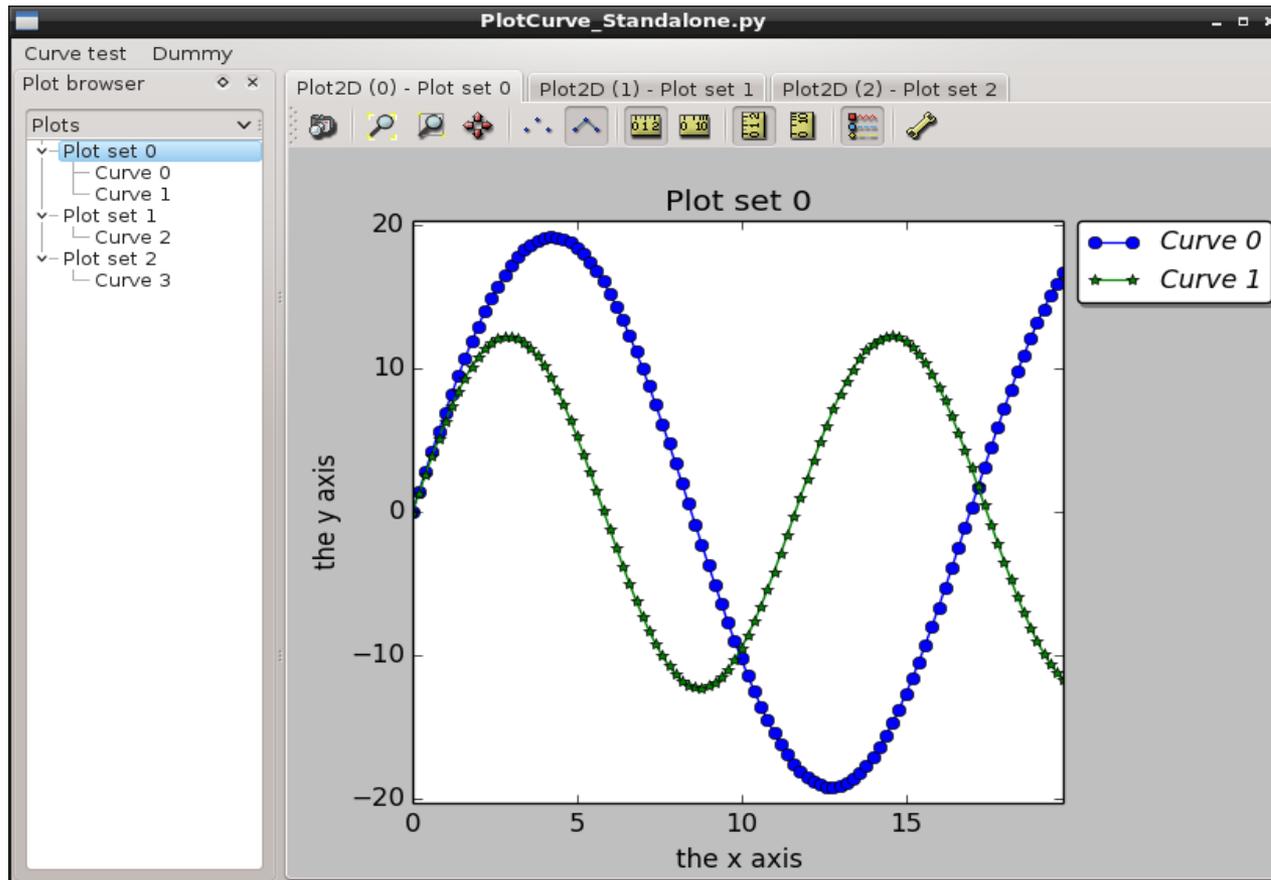
## Implementation

- Full Python (like *matplotlib*), with PyQt4 backend
  - With a C++ wrapping (using internal SALOME interpreter)
  
- Public API does not expose MVC design. Only simple commands
  - `AddCurve`, `DeleteCurve`
  - `AddPlotSet`, `DeletePlotSet`
  - `SetCurveMarker`, `SetXLog`, `SetCurveLabel`
  
- Main parameters: unique curve identifier (=an integer), and a unique plot set identifier
  
- Testing part: ~45 unit tests with screenshot comparison

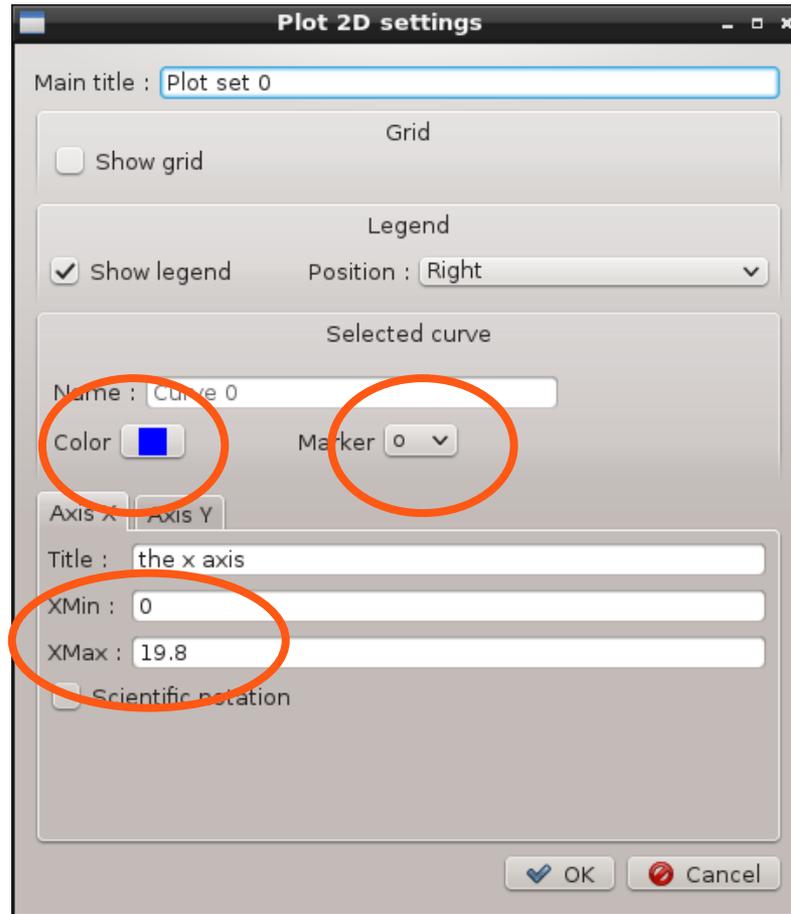
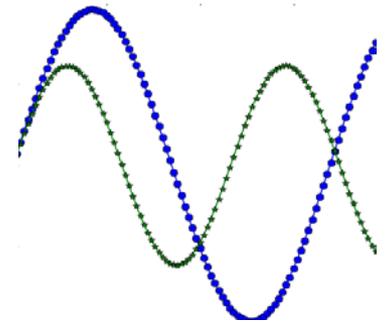
# SOME SCREENSHOTS (1/2)



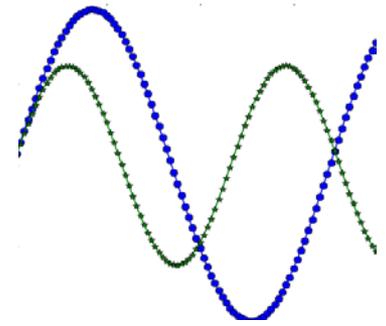
## Illustrative standalone application



# SOME SCREENSHOTS (2/2)



# CONCLUSION & TODO LIST



## Conclusion

- Simple interface to most commonly requested plotting facilities
- *matplotlib* completely encapsulated
- New functionalities: finer handling of curves (changing colors, markers, etc ...)

## On-going work

- Have a full dedicated SALOME module using the tool
  - Loading files in a table, etc ...
  - Potentially create/modify columns, using PANDAS for example
  - And obviously plot the data loaded this way
- Major expected improvements
  - Double Y axis (was in former Plot2D and is needed at CEA)
  - Performance optimization: not as efficient as pure *matplotlib* yet