PyTelTools: Python scripts and GUI to automate Telemac post-processing tasks

Luc Duron
Engineering department
Compagnie Nationale du Rhône (CNR)
Lyon, France
l.duron@cnr.tm.fr

Yishu Wang
Bioinformatics and Modelling department
Institut National des Sciences Appliquées (INSA)
Lyon, France
yishu.wang@insa-lyon.fr

***Abstract* — PyTelTools is a set of Python scripts recently developed by CNR Engineering department. Developed scripts are released as an open source software, which is intended to be accessible, extensible and customizable. An easy-to-use Python GUI was implemented to facilitate post-processing tasks on 2D or 3D TELEMAC-MASCARET computation results.**

A wide range of tools is offered to the modeller within PyTelTools. These tools include file conversions, interpolations, calculations, statistics and visualization. A workflow interface was developed to chain these unitary tasks. Indeed, after defining graphically a conceptual model representing the tasks nesting, runs can be performed on multiple simulations in parallel.

I. Introduction

A. CNR Engineering

CNR, the first producer of exclusively renewable energy in France, operates 18 hydroelectric facilities on the Rhone River, with an installed capacity of 3000 MW. Under the operation of these facilities, CNR has developed and uses specific hydro-informatics tools tailored to their individual needs, either for studies or for operational applications.

For engineering tasks requiring a fine resolution, TELEMAC-MASCARET is often used in connection with its associated pre and post-processing software. With the multiplicity of processing software along the modelling chain, some tools are required to interface them and to extend their possibilities.

B. Objectives and development guidelines

CNR Engineering recently decided to constitute a set of Python scripts which are intended to be robust, extensible, modular and accessible to the community. This project was named PyTelTools and stands for "Python Telemac Tools". These scripts do not rely on Python scripts provided in the official openTelemac SVN repository, but were developed from scratch and are fully functional with Python 3.

A user documentation [1] and a developer documentation are provided online, with the source code. The developed scripts include some files parsers (Serafin, BlueKenue ...) and a collection of mathematical or spatial calculations based on 2D or 3D Telemac results. Moreover a graphical user interface (GUI) is provided to make them available, easy to use and to facilitate systematic exports and post-processing tasks.

These tools are extensible, thanks to a documentation, snippets and contributing guidelines. Additionally, a customization is possible and allows the user to change default behavior (e.g. the interface, I/O parameters, some calculations assumptions ...).

II. Description and classification of post-processing tools

Main developed tools are described in this section in the different tasks items.

A. Compute variables tasks

"Extract (or Select) variables" is a tool whose aim is to compute derived variables from a Serafin input file. All the potential derived variables are suggested to the user depending on variables available in input file and equations implemented (more than twenty equations are implemented [2]).

Equations can be rather simple, but the tool handles multiple indirect computation of variables. For example, to compute Froude number, the equation states that it requires water depth and velocity magnitude variables. But if these variables are not already present in input Serafin file, they could be computed with other variables, such as bottom, free surface, velocity on X and velocity on Y.

Friction velocity can be recalculated depending on friction law and allows for example to compute Rouse numbers based on user defined settling velocities.

To compute complex and custom equations in 2D, the user has to use the tool named "Variable Calculator". This advanced calculator allows to define successive conditions on variables or on zones defined spatially with polygons.

B. *Compute fluxes tasks*

Computation of fluxes in 2D through an arbitrary section is implemented with a generic integration of a product with one scalar, one vector field and eventually water depth (a particular second scalar). This allows computation of different types of liquid and solid fluxes, depending on variables present in input Serafin file. An example for liquid fluxes is presented in Table 1 (with Telemac 2D variable names). For this example, among all the possibilities, the first computation (with velocities U and V) is more accurate because velocity vectors are projected on a normal vector to the cross-section, and it relies on a quadratic function integration. That quadric function is defined by parts.

(X, Y) Vector field	Height	Scalar
VELOCITY U VELOCITY V	WATER DEPTH	-
FLOWRATE ALONG X FLOWRATE ALONG Y	-	-
-	WATER DEPTH	SCALAR VELOCITY
-	-	SCALAR FLOWRATE

Table 1: Example of flux computation possibilities for liquid fluxes

C. *Compute volume tasks*

Different kind of volumes can be computed in a set of 2D polygons defined by the user. Volumes are basically described by an upper and a lower surface. These surfaces are defined by values at nodes of the 2D mesh and for all frames. Upper values correspond to a scalar variable, whereas lower values can be zero, initial frame of same variable or another scalar variable. Typical applications with their upper and lower values are summarized in Table 2.

Upper values	Lower values	Interpretation
WATER DEPTH	0	Liquid volume
FREE SURFACE	BOTTOM	Liquid volume
BOTTOM	RIGID BED	Solid volume
BOTTOM	Initial values	Volume from beginning

Table 2: Examples of typical volume computations
(with TELEMAC-2D variable names in capital letters)

Positive and negative volumes for each polygon can be computed in addition to the net volume. Exact volumes are computed, including pieces of elements which are cut by a polygon and/or by the lower surface defined by its values at nodes. The underlying theory is established on a document available online [3].

D. *Interpolate and project along lines tasks*

Interpolating a 2D variable along an open line provides a 1D view, with the x-axis being in general the cumulated distance. To have the exact evolution of the variable along the line, the initial line is populated with intersection points between it and the 2D mesh.

For fluvial applications, involving a long river reach, modellers often fall back on a 1D view along the hydraulics axis. Along this specific line, called reference line and drawn within the minor bed, a curvilinear distance (or kilometre point) is defined, starting from the first point.

The tool to project lines on the reference line interpolates variables along a set of lines and performs a projection to determine their curvilinear distance along the reference line. Fig. 1 shows on the left, the definition of the reference line (Line 1) and dykes (Lines 2 to 4) and on the right, the computed longitudinal profile mixing different variables (bottom elevation for dykes and free surface for the reference line).

E. *Compare two 2D results tasks*

Comparison of a variable of 2D results consists of a wide range of methods. Computation of differences between two meshes, which are eventually different, is one of simplest method and gives the spatial distribution of deviation.

For quantified comparison, modellers often compute criteria or turn to statistical analysis. However computation of deviation criteria or statistics are not direct because the triangular mesh is irregular. To overcome that, developed scripts do not rely on an intermediate regular grid, but compute the exact deviation thanks to some volume calculations [3].

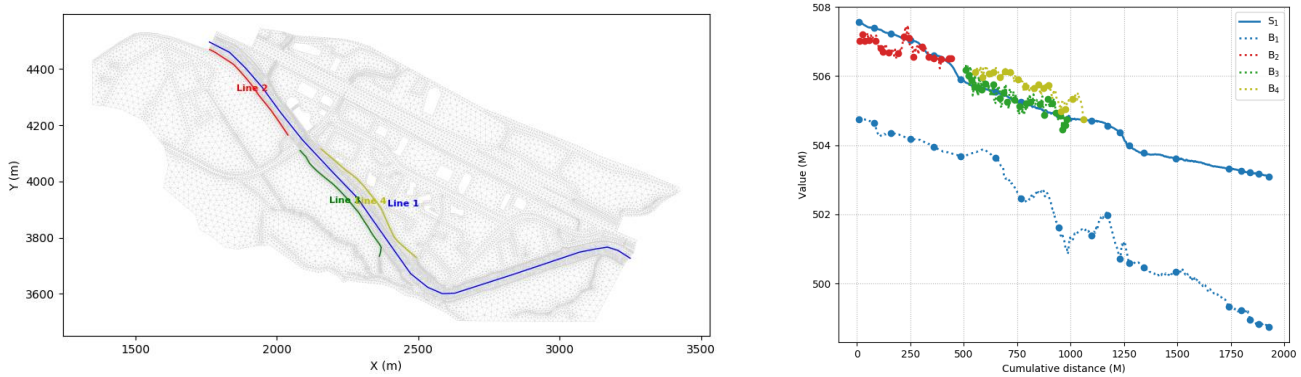

Fig. 1: Example of Project Lines application with the map (left) and longitudinal profile with bottom (dotted lines denoted Bi) and free surface elevation

(continuous line denoted S_1) (right)

Comparison criteria developed in PyTelTools are the following:

- Mean Signed Deviation (MSD)
- Mean Absolute Deviation (MAD)
- Root Mean Square Deviation (RMSD)
- Brier Skill Score (BSS)

The last criterion is mainly used for sediment transport issues, to compare bed evolution during two bathymetric campaigns.

Temporal evolution of criteria is interesting for analyse of the dynamic and can point out impacts (or even divergence) at specific time or during events. Some above-mentioned criteria can be visualized over time, as shown in Fig. 2 for the MAD criteria. This figure simply compares bottom elevation (on the whole domain) between a temporal 2D sediment result and the target (or reference). The reference, which is constant over time, is a bathymetric campaign, which should correspond to the last computed frame. In this example, MAD tends nearly to zero and shows some inflexion points corresponding to flood events.

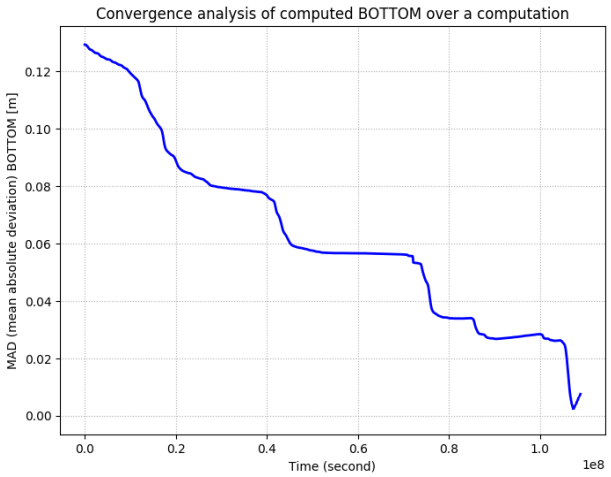

Fig. 2: Evolution of BOTTOM deviation from the target

Moreover, all these criteria can be computed over the whole domain or in a set of polygons to study specific areas of interests.

III. BUILDING A WORKFLOW

A. *Context and notions*

Most post-processing tasks are not isolated, but are rather a part of a chain of operations. Using a tool handling workflow is more convenient and efficient, because these steps can be wrapped into a single operation, which can be modified later or repeated on other input data. This automation of post-processing tasks can save time, efforts, and avoid errors while repeating them manually. Some software already include their own automation tool:

- QGIS has his so-called graphical modeller,
- Paraview allows on-the-fly trace recording [4].

Developed scripts include a tailor-made graphical modeller, which is adapted and optimized to the hydraulic modeller needs. It is independent of any graphical post-processing, GIS or statistical software, but is not intended to replace any of them.

Conceptual Model Principle

PyTelTools offers an easy-to-use drag-and-drop interface that allows users to construct a series of computational tasks that can be chained up by drawing links between them. The resulting graph is a directed acyclic graph (DAG) and each node represents a unitary tool (or task).

The workflow graph not only represents graphically different steps of data manipulation, computation and visualization, it can also be executed within the software which also provides process monitoring in real time.

B. *Provided tools and specificities*

The conceptual model can include the post-processing tasks listed below. This list groups together tools by category.

- Load files
  - Load Serafin 2D or 3D
  - Load Reference Serafin
  - Load 2D Polygons
  - Load 2D Open Polylines
  - Load 2D Points
- Write files
  - Convert to LandXML, shp, vtk
  - Write Serafin 2D or 3D
- Basic operations
  - Select Variables
  - Select Time range or Single Frame
  - Add Rouse Numbers
  - Convert to Single Precision
  - Add Transformation
- Operators
  - Max, Min, Mean, SynchMax
  - Project B on A
  - A Minus B (or inversely)
  - Max(A, B) or Min(A, B)
- Calculations
  - Compute time of arrival and duration
  - Compute Volume
  - Compute Flux
  - Interpolate on Points
  - Interpolate along Lines
  - Project Lines
- Visualization
  - Show Mesh
  - Locate Points, Open Lines or Polygons
  - Visualize Scalars, Vectors
  - Point, Flux or Volume Plot
  - Project Lines Plot
  - Multiple variables or frames Line Plot
  - Vertical 3D Cross-Section
  - Vertical 3D Temporal Profile

C. *Two views for a more complete automation*

Two levels of automations can be performed by PyTelTools workflow.

The first one is intended to process a single simulation and is therefore called "Mono". In the mono view, a run will go through the whole post-processing tasks defined in the conceptual model. It is in this view that the graph is built and each tool configured.

On the other hand, the "Multi" view in the workflow tool works on a level above and will apply a "Mono run" to a set of simulations.

IV. Workflow Examples

A. *Typical Applications*

Common applications of a workflow for Telemac post-processing are mainly:

- sensitivity studies (on parameters, boundary conditions, ...),
- calibration processes,
- batch conversion or simplification of Telemac result files.

B. *Basic 2D hydraulic calibration*

A first application of PyTelTools workflow was led on a simplified 2D hydraulic model (only around 20 000 nodes for 39 000 elements). This example is based on a friction coefficient calibration process, for which 6 different roughness zonings were defined.

This kind of calibration process commonly includes several computations:

A. compute maximum of variables over flood duration,
B. interpolate along lines (and eventually project them on the hydraulic axis) and/or at points located at gauging stations,
C. compute fluxes through some cross-sections.

For each roughness zoning, a simulation was performed and the same post-processing tasks where applied. Post-processing tasks are described in the graph presented on Fig.3 (left panel).

C. *Sedimentary study*

For a study dealing with sediment transport issues on a large scale model, workflow tool was applied profitably. An example of some automated post-processing tasks is presented on Fig. 4. These different tools were defined to:

- compute liquid and solid fluxes,
- compute volumes of erosion and deposition,
- produce maps afterwards from shp and TIN files.

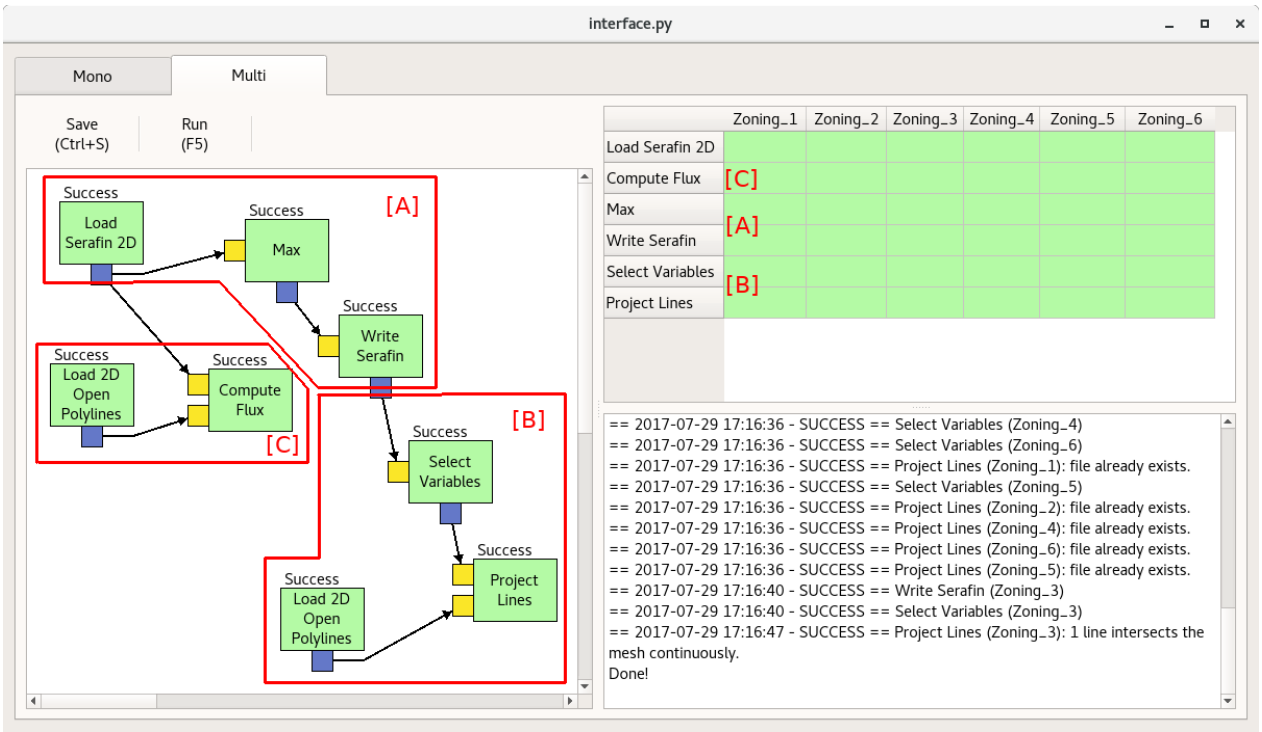

Fig. 3: Workflow graph (left panel) and process status after a successful run (right panel)

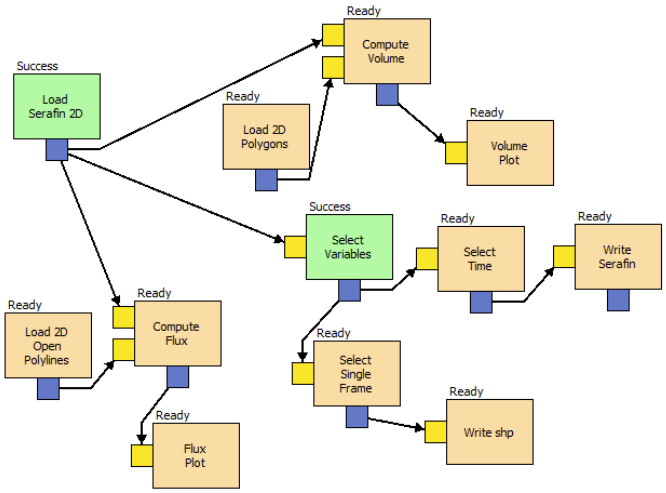

Fig. 4: Workflow graph with running status of post-processing tasks for a sedimentary study

V. 3D Visualization

In addition to writing processed data, PyTelTools provides a set of visualization plots, which can be customized and exported in mass. Two visualization examples are presented hereafter and are dedicated to 3D results analysis on the vertical axis.

The first visualization plot (Fig. 5) shows vertical distribution of mud concentration over the time, at a single point. In this example, during the period representing two tidal cycles, high concentration of mud occurs at low tides.

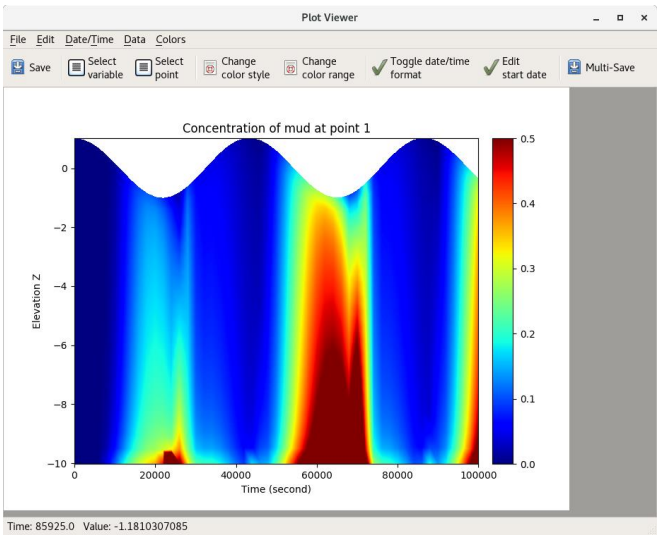

Fig. 5: Temporal vertical profile visualization of mud concentration at a specific location

The second example deals with 3D velocity field through a cross-section. This data is often measured in situ (with a so-called ADCP instrument) and compared to 3D results.

Fig. 6 shows the superimposition of measurements and results on the same cross-section. ADCP measurements were processed with ADCPtool, which is developed by TU Graz, whereas TELEMAC-3D results post-processing was performed with PyTelTools.

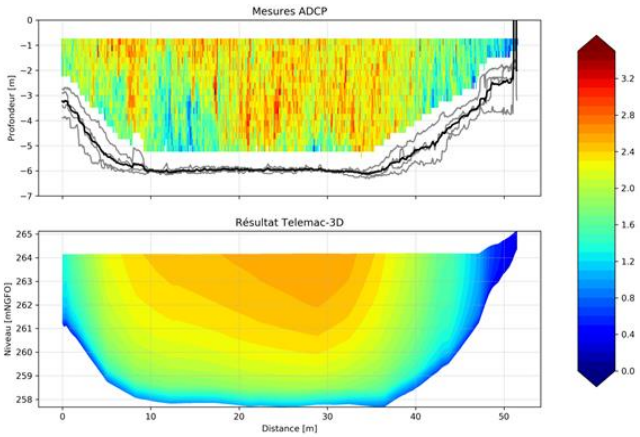

Fig. 6: Comparison of ADCP measurements (upper plot) and TELEMAC-3D results (lower plot) through a cross-section

VI. Multi-Folder Parallel Execution

A. Basic principles

Default Python interpreter was designed with simplicity in mind and has a thread-safe mechanism. However some Python modules exist to spawn multiple subprocesses, and PyTelTools "Multi view" uses the library multiprocessing to distribute tasks on different data to multiple CPU cores.

Some performance tests were carried out in order to validate the proper functioning and efficiency of parallelism in PyTelTools.

B. Performance test

Local workstation

The computer used for the benchmark is a post-processing machine, whose performances are slightly higher than a typical PC workstation. Its characteristics are summarized below:

- Intel(R) Xeon(R) E5-2690, 2.90 GHz, 64 bits (8 cores, 16 threads),
- Windows Server with Python v3.6.1 64 bits

Test case

The example A, described in part IV, was used for the present benchmark. This simple and representative example includes only one bifurcation (just after "Load Serafin 2D" node), which leads to two independent chains of tasks for a single simulation (or folder). In this example, most of the computation time is spent on flux and interpolation calculations, which limit I/O bound state.

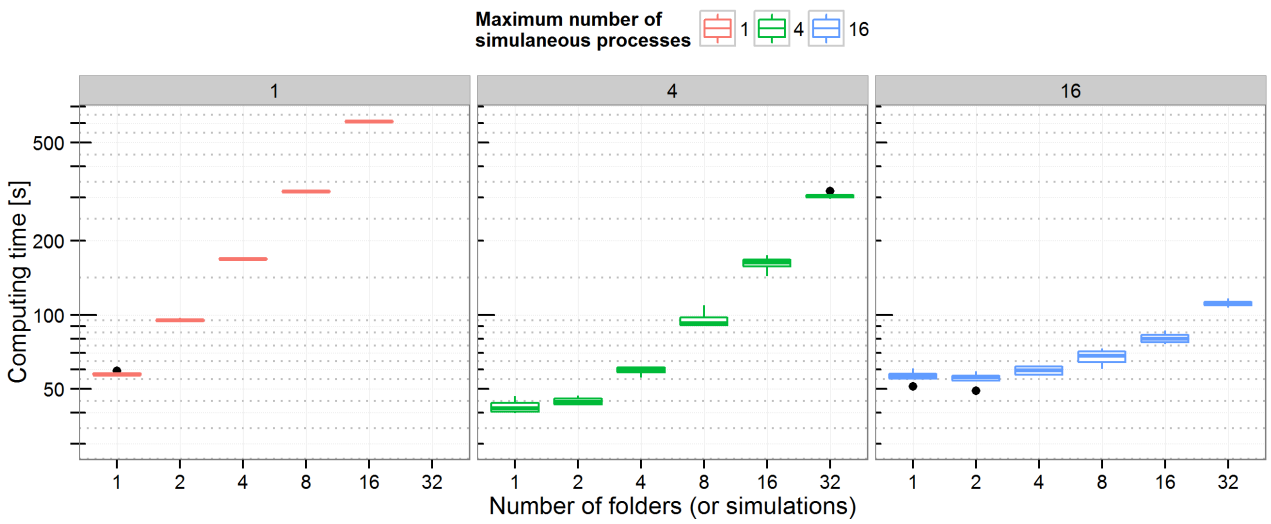

Fig. 5: Computing time benchmark on example A, with different number of folders and CPU cores (1, 4 and 16, which corresponds to machine nproc)

C. *Main outcomes*

The computing benchmark results are presented on Fig. 5 (with logarithmic axis). A sensitivity analysis was performed on two variables: the number of CPU cores (called maximum number of simultaneous processes) and the number of folders. For a given couple of these variables, each boxplot includes at least 5 repetitions.

The different repetitions show that the results are well reproducible. The scaling is coherent with the number of folders and CPU limitations. Indeed, for a number of folders which is large enough (compared to CPU cores), the computing time increases linearly. Moreover, in this case, the slope of the curve is also proportional to the CPU cores number.

Because of the two independent chains of tasks in the graph, it is still interesting to have more CPU cores than the number of folders to process.

REFERENCES

[1] PyTelTools User Documentation (wiki based), https://github.com/CNR-Engineering/PyTelTools/wiki

[2] Equations integrated in PyTelTools, August 2017, https://github.com/CNR-Engineering/PyTelTools_media/raw/master/latex/equations.pdf

[3] Mathematical definitions for PyTelTools, August 2017, https://github.com/CNR-Engineering/PyTelTools_media/raw/master/latex/mathematical_definitions.pdf

[4] U. Ayachit, "The Paraview Guide, A Parallel Visualization Application", Kitware