QGIS as a pre- and post-processor for TELEMAC: mesh generation and output visualization

P. Prodanovic
Hydrotechnical Engineer
Riggs Engineering Ltd.
London, Ontario, Canada
pprodanovic@riggsengineering.com

Abstract— **This paper presents a summary of scripts named pputils that link QGIS to tasks common in numerical modeling of free surface flows, such mesh generation and visualization of model output. QGIS is an open source Geographic Information System under active development and supported under all major platforms. The scripts in pputils are written in the Python programming language relying on libraries Matplotlib, Numpy, and the Python parser scripts that are part of the TELEMAC source code. Mesh generation is accomplished by developing skeleton geometry within the QGIS environment (model boundary, constraint lines, islands, nodes, etc.) and exporting it to a WKT (well known text) format. The WKT format is then used by pputils to generate steering files for Triangle and Gmsh mesh generation programs. The meshing programs are then executed, and produce a mesh respecting user specified constraints. The bottom elevation and spacially varying friction attributes in the generated mesh are created and final output saved for further TELEMAC simulations. After the simulations are complete, the scripts in pputils take the TELEMAC output files and generate a set of gridded files (with a user specified resolution), thus allowing snapshots of the model output to be visualized within the QGIS environment. The same also applies to display of vector variables. Having model output available in the QGIS environment allows the user to create publication quality output of the TELEMAC simulation results.**

I. INTRODUTION

Increasing development of open source Geographic Information Systems (GIS) has had a marked impact in how spacial data is managed. By being open source, current GIS applications provide individual users with a real alternative to commercially available GIS packages. In recent years open source GIS has matured that it now allows users to perform a wide variety of spacial data management tasks using both vector and raster data. For example, open source GIS applications provide numerical modeling specialist a useful set of tools for vector based geometry manipulation (such as importing and editing shoreline features, creating model boundaries, adding constraint lines, islands, re-sampling polylines, etc.). Tasks that in the past would require a Computer Aided Design (CAD) packages, nowadays need only an open source GIS package. The open source GIS package relied upon in this work is QGIS [1]. Alternative open source GIS applications include SAGA [2] and GRASS GIS [3], GDAL [4] although there are others as well.

A typical free surface flow modeling project requires the user to collect, assemble, merge, and edit geometric data like topographic and bathymetric surveys, lidar data (masspoints and breaklines), digital elevation models, etc. Free surface modeling projects are defined here as those that study river or coastal hydraulics, sediment transport, wave climate analysis, water quality assessments, and others using 2D or 3D numerical modeling codes. Typical projects of this kind also require one to manage large data sets like aerial images, land use data, and other spacial databases. Such data sets are used in numerical modeling projects where the user is required to define model boundary, construct internal constraint lines, delineate islands or holes, and include other geometric features in the domain. Following geometrical edits, the next step is to apply a meshing algorithm to: i) construct terrain models (or digital surfaces) that are used as the basis for interpolating or assigning elevations, and ii) construct a quality model mesh for use in numerical simulations. After the input meshes are assembled, and bottom elevations and friction or other attributes are assigned to the mesh, the numerical simulations take place. Following completion of the numerical simulations, results of the models need to be conveyed, often to those not familiar with intricacies of numerical analysis. High quality graphical outputs are thus required to include in reports and provide the reader with a graphical summary of simulated behaviour under study.

Given the rapid development of open source GIS applications (QGIS in particular), the near future will likely allow all tasks typical in free surface modeling projects to be completed within a GIS environment. This means it would likely be possible to open a GIS package, import the necessary topographic and bathymetric data, build terrain models to represent surfaces, produce a quality mesh of a domain for use in simulations, interpolate the quality mesh from the generated surface, write model steering files, execute the numerical simulations, view model outputs, and prepare publication ready figures of the desired output.

No doubt, creating an interface envisioned above will require significant effort by many in the user community, but tools are now available that make this possible. There has been progress to date in the regard. Existing developments include the following projects:

- Lutra Constulting's Crayfish project [5], which allows visualization of hydraulic model output within the QGIS environment,
- ETH Zurich's BASEMENT hydraulic modeling system [6], and in particular its BASEmesh QGIS plugin that allows the user to create terrain surfaces, develop quality model meshes, perform necessary data interpolations, and prepare model geometry files all within QGIS,
- Uwe Merkel's TELEMAC Selafin Reader for QGIS [7], that allows visualization of TELEMAC model output within QGIS.

Each of the above projects has advanced use of using open source software in free surface modeling projects. They have served as both an inspiration and motivation for the development of pputils that are the focus of this paper. The main objective of pputils is to continue the trend in using open source software as both pre- and post-processors for use in free surface flow modeling projects.

One of the guiding principles that lead to the development of pputils was driven by the need to efficiently complete tasks typical in environmental flow modeling projects, and the desire to do so entirely using open source software. Use of commercially available software, or software that is free but not in open source, was not further considered.

The guiding criteria that was set when developing pputils were the following:

- All code must be entirely in open source,
- It must work on all common platforms,
- It must have absolute minimal dependencies and easy installation,
- It must be computationally efficient with minimal execution times, and
- All scripts must be executed using standard command line.

It is believed that if the above criteria are met, the tools developed could easily be incorporated in a future graphical user interface, and be integrated within an open source GIS environment like QGIS.

A. Scope of paper

This paper presents a summary of command line tools collectively named pputils that provide its users with an ability to complete all aspects of a typical environmental flow modeling project while using only open source software. Background information is provided on the open source software used, including QGIS, Triangle [8,9] and Gmsh [10] meshing programs, as well as Numpy [11] and Matplotlib [12] libraries part of the Python programming language. An illustration of the process used in the construction of meshes used in numerical simulations is presented, and includes geometric input preparation using QGIS (boundary definition, specifying mesh constraints, re-sampling polylines, etc.), development of Triangulated Irregular Networks (TIN) for terrain surfaces using Triangle, quality mesh generation using Gmsh, interpolation of quality mesh from a previously developed TIN, and creation of Selafin files for use in simulations using the TELEMAC modeling system. Visualization of the TELEMAC model output within QGIS is also illustrated, using both field and vector variables.

II. BACKGROUND

This section of the paper presents a brief overview of the tools relied by pputils. For example, open source QGIS application is required with which the user performs all geometrical edits, and prepare input files used by scripts in pputils. In this regard, QGIS is not used out of necessity, but simply out of convenience. It is possible for the user to create manually (in a simple text editor) all of the inputs required for say, mesh generation using Triangle and Gmsh. However, tasks like drawing and re-sampling polylines, joining, breaking, merging and otherwise editing polygons can be accomplished with relative ease using QGIS, that it becomes simply easier to use a graphical user interface than a text editor. As noted earlier, QGIS is used in this work, although no doubt the same tasks could be carried out using other GIS packages. Given the wide spread development of open source software, other users may find ways to accomplish same tasks differently, and perhaps more efficiently than is presented in this paper.

A. Open source GIS

The focus on this paper is on using QGIS on pre- and post-processing tasks associated with typical free surface flow modeling projects using the TELEMAC modeling system. QGIS is a free and open source Geographic Information System application that, in the most general sense, provides its users with editing, viewing and analysis of spacial data. QGIS has reached a mature status in its evolution, having a large number of volunteer developers who provide regular updates and bug fixes to the program. The application has been translated in approximately 50 languages, and is freely available on Windows, Mac and Linux operating systems.

QGIS also interfaces with other open source GIS packages, including GRASS, SAGA, GDAL and others, and thus provides its users with access to a wide range of geospacial tools all within one application. QGIS Plugins, which are commonly written in the Python programming language act to further customize and extend capabilities of QGIS. The Crayfish, BASEmesh and Selafin reader for QGIS projects are all plugins written to work inside QGIS.

B. Mesh generation for TELEMAC

Mesh generation for use in the TELEMAC modeling system requires high quality triangular meshes. A number of tools currently exist that accomplish this task. Perhaps the most popular in the TELEMAC user community is the BlueKenue application [13], developed by researchers at the National Research Council in Ottawa, Canada. BlueKenue is a pre- and post-processor for TELEMAC, with features to generate and interpolate meshes, and read and view model output. BlueKenue is free, but not in open source. It is available only on the Windows platform.

Another set of pre- and post-processors for TELEMAC (and other models) are Gismo, Janet, and Davit, developed by Smile Consult GmbH [14]. Gismo, Janet, and Davit allow their users highly advanced pre- and post- processing capabilities. The programs from Smile Consult are commercial applications and are available for Windows, Mac, and Linux platforms.

Open source mesh generation for TELEMAC is perhaps not as common as above applications. Triangle [8, 9] mesh generator, developed by JR Shewchuk at the University of California at Berkley, is available in its entirety as C source code from the author's website. Further, description of Triangle is given in [8]:

"Triangle is a C program for two-dimensional mesh generation and construction of Delaunay triangulations, constrained Delaunay triangulations, and Voronoï diagrams. Triangle is fast, memory-efficient, and robust; it computes Delaunay triangulations and constrained Delaunay triangulations exactly. ... Features [of Triangle] include user-specified constraints on angles and triangle areas, user-specified holes and concavities, and the economical use of exact arithmetic to improve robustness" p. 203.

After compiling Triangle, the program is executed from the command line, with the user specifying a number of parameters and input files. The BASEmesh QGIS plugin, is one variant of pre- and post-processor to the Triangle mesh generator. Another is pputils script (gis2triangle.py), described in the subsequent section of this paper.

Another open source triangular mesh generator is Gmsh [10], developed by Christophe Geuzaine and Jean-François Remacle, at the Université de Liège, Belguim. Gmsh is an all purpose 2D and 3D finite element mesh generator (more than just triangular meshes are included), with a built-in GUI CAD engine for pre- and post-processing. Gmsh's GUI is developed using FLTK GUI toolkit, making it extremely fast, light while at the same time providing its users advanced graphical input and visualization features. The Gmsh program is available as open source, and is supported in Windows, Mac and Linux. The GUI has four different modules: geometry, mesh, solver and post-processing. Geometry for Gmsh can be generated interactively using its GUI, or be imported from external files using a number of different formats. Geometry can also be developed using Gmsh's text based steering files.

Previously, Dr. Olivier Gourgue at the Flanders Hydraulic Research developed a set of post-processing scripts using Matlab named PUG [15] that are able to convert Gmsh output to the Selafin format for use in TELEMAC simulations. The PUG scripts also produce the TELEMAC *.cli boundary conditions files.

C. Python programming language

In order to create a link between QGIS and mesh generation programs Triangle and Gmsh, a script is needed to convert GIS geometry and create a steering file understood by each respective meshing algorithm. Further, scripts are also needed to extract TELEMAC simulation output and port it back to QGIS.

Package Numpy, used for scientific computing on the Python language was heavily used in the development of pputils scripts. Numpy's vectorized implementation of common functions ensured that tasks involving numerical calculations are executed extremely efficiently, with minimal waiting times for the user. Using Numpy, rectangular grids can be generated using tens of millions of cells in a matter of seconds using just today's desktop computers.

The Python library Matplotlib was also heavily relied upon in pputils, specifically its triangulation and gridding algorithms. Matplotlib is a plotting library and is a numerical extension of Numpy. Using Matplotlib triangulation algorithms allowed scripts in pputils to carry out conversion tasks (such as converting TELEMAC's simulation output to a gridded format used by QGIS).

Python provides functionality of a general scripting language with excellent libraries used for numerical analysis, thereby making it an extremely useful tool for general scientific analysis of data. Since Python programming language is used by both QGIS and TELEMAC, it seems natural that it also be used as a scripting language to link QGIS and TELEMAC.

III. MESH GENERATION

There are at least two different types of meshes commonly used in free surface flow modeling projects. One deals with the generation of digital terrain surfaces or Triangulated Irregular Networks (TINs) from topographic, bathymetric, lidar and other digital data. TINs are used to represent digital surfaces of table lands, rivers or sea beds, and are used for, among other things, as a basis for interpolating (and assigning elevations) to the quality mesh to be used in numerical simulations. The other kind of mesh is the quality mesh on which numerical simulations are carried out. Regardless of the type of mesh the user wishes to generate, input data preparation is very similar. A skeleton geometry (meaning boundary polygon, constraint lines, islands or holes, and/or embedded nodes) must be prepared first.

A. Input preparation using QGIS

This section illustrates how to use the open source QGIS application to develop skeleton geometry used for mesh generation. The scripts in pputils will take the skeleton

geometry, do some format conversions, and then generate steering files for both Triangle and Gmsh mesh generation programs. It will be up to the user to launch these programs, and generate triangular meshes. More on this process is provided below.

The logic in how data management is used in pputils parallels the data management structure used by the BASEmesh QGIS plugin. There are some differences, of course. First, the user is required to define a polyline representing the boundary of the mesh domain. A required condition is that the boundary polyline be a closed shape, implying that the same coordinate be used as starting and ending point. Second and optional, internal constraint lines are developed, and can be either closed or open polylines. Third and optional, islands (or holes) in the domain are defined using closed polylines. Finally, the user must define a master nodes file, which contains vertices of all of the boundary, constraint, and island (if any) files. Creation of the master nodes files in QGIS is rather simple, as the user is required to merge all of the polylines, extract their vertices as individual nodes, and save the file as a text based xyz format. If there are to be embedded nodes in the mesh (as would be the case in the generation of digital surfaces or TINs), the nodes file must include these, in addition to the vertices of the boundary polyline, lines and islands. The inherent assumption in the above procedure is that all of the polyline vertices must snap (within a reasonable horizontal tolerance) to the coordinates of the master nodes file. Note that only the master nodes file contain xyz attributes), while the boundary, line constraints, and holes files contain shapeid,x,y attributes. The scripts that generate the mesh generation steering files for Triangle and Gmsh use sophisticated searching algorithms that look up the z value from the master nodes file for all boundary, lines, and island files (if any).

The user is required to save within QGIS individual files, according to the formats specified. For the master nodes file, an xyz, comma separated file suffices. For the boundaries, lines, and hole files, the easiest is to save each as a WKT (well known text) format within QGIS. A script within the pputils named wkt2csv.py takes the files in WKT format, and converts them to a shapeid,x,y comma separated format used by pputils.

In summary, before going to mesh generation, the user prepares nodes.csv (required), boundary.csv (required), lines.csv (optional) and holes.csv (optional) files.

B. Triangulated Irregular Network (TIN) using Triangle mesh generator

As a note of completeness, pputils use a slightly different format for specification of holes for use in the Triangle mesh generator. Simply by virtue of the requirements of the Triangle's steering file, the boundary of the holes should be included as closed lines in the lines file, and point coordinates (holeid,x,y) within hole boundary must be included in the holes file. Examples provided with the pputils source code explain this further.

In order for the user to generate a TIN surface using the Triangle mesh generator, the following files are required:

1. nodes.csv (containing a list of all nodes in xyz format, comma separated),
2. boundary.csv (containing comma separated node listings of mesh boundary, specified as shapeid,x,y),
3. lines.csv (optional, containing comma separated constraint lines or breaklines, specified as shapeid,x,y). If there are holes, they should be specified as closed lines in the lines file.
4. holes.csv (optional, containing comma separated point file with holeid,x,y attributes). x,y coordinates should be placed inside the hole closed polyline.

After creating the above files, the user creates the Triangle steering file by executing the following Python script:

```
python gis2triangle.py -n nodes.csv -b
boundary.csv -l lines.csv -h holes.csv -o
out.poly
```

If there are no lines or holes files needed, the user simply enters 'none', without the quotes as the -l and -h argument to the script.

The file out.poly is generated that is a steering file for Triangle. To generate the TIN mesh, the user executes the Triangle mesh generator using previously compiled binary program:

```
triangle_64 out.poly
```

Where triangle_64 is the Linux 64 bit version of the Triangle mesh generator. The compiled binaries in pputils also provide triangle_32 and triangle_32.exe, and represent Linux 32 bit and Windows 32 binaries.

The above command generates out.1.node, out.1.ele and out.1.poly text based files. A script in pputils takes these and creates an Adcirc based mesh:

```
python triangle2adcirc.py -n out.1.node -e
out.1.ele -o out.grd
```

Where the out.grd is the TIN in Adcirc mesh format. The Adcirc format was selected for use in pputils as it is a simple text based mesh file.

Suppose the user wishes to convert the out.grd TIN file to a regular *.asc gridded file (often referred to as the digital elevation model or DEM), the following would be executed:

```
python adcirc2asc.py -i out.grd -s 10 -o
out.asc
```

Where the out.grd is the Adcirc TIN file generated above, -s parameter is the output grid spacing in meters (10 m in above example), and -o parameter is the resulting output DEM file. The Matplotlib library is used to read the triangulation from the TIN file, and create a gridded DEM file. The *.asc file can easily be loaded into QGIS, or be used in input to other gridded based numerical simulation applications (like the SWAN wave model for example).

Testing was done by reducing the grid resolution and producing a DEM with tens million grid points. The processing for this task took in the order of seconds on a desktop computer due to vectorized Numpy and Matplotlib functions.

Lastly, should the user wish to display the mesh within the QGIS environment, the following script should be used:

```
python adcirc2wkt.py -i out.grd -o
outWKT_e.csv outWKT_n.csv
```

Where the outWKT_e.csv and out WKT_n.csv files are WKT (well known text) format output of the elements (as polygons) and nodes (as points) that can be easily loaded into QGIS.

Alternatively, the following script could be used to create a *.dxf file of the Adcirc file:

```
python adcirc2dxf.py -i out.grd -o out.dxf
```

Where -i represents the input Adcirc file and -o is the output file written in *.dxf format.

An example output of the TIN model of the bathymetry of Lake Manitouwabing, located in northern Ontario, Canada is shown in Figure 1 (global view) and Figure 2 (zoomed in view).

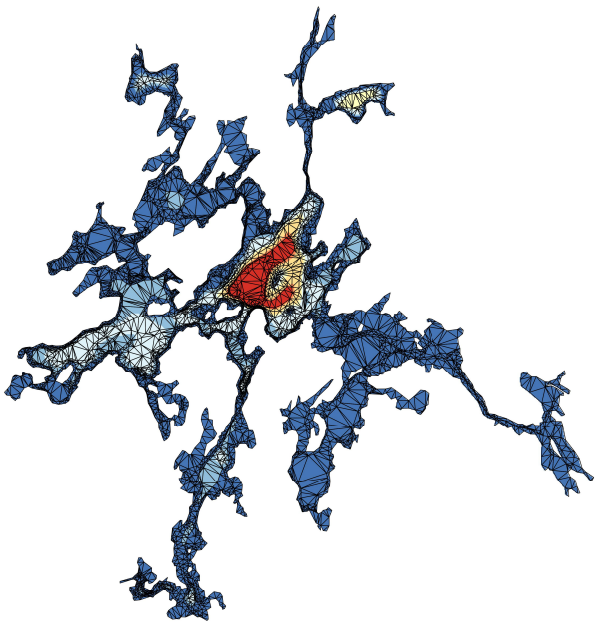

Figure 1: Lake Manitouwabing TIN generated by Triangle

Even though this section uses the Triangle mesh generator to generate a TIN, Triangle can also be used to create a quality mesh for use in numerical simulations. The interested user is thus directed to documentation of Triangle which covers command line flags and input parameters used to produce a quality based mesh. Note the same steering file generated by gis2triangle.py would be used as a starting point towards this task.

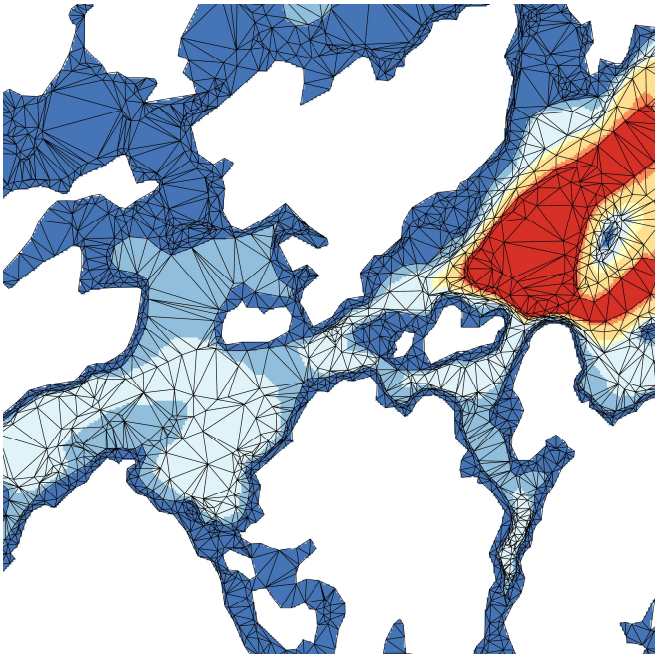

Figure 2: Close up of Lake Manitouwabing TIN

C. Quality mesh generation using Gmsh mesh generator

As an alternative to Triangle, the user is also given the option to use the Gmsh mesh generator. In the same way as Triangle, the user is expected to prepare the following input files:

1. nodes.csv (containing a list of all nodes in xyz format, comma separated),
2. boundary.csv (containing comma separated node listings of mesh boundary, specified as shapeid,x,y),
3. lines.csv (optional, containing comma separated constraint lines or breaklines, specified as shapeid,x,y). The lines can be either open or closed lines,
4. holes.csv (optional, containing comma separated hole or island closed polylines specified as holeid,x,y).

To generate a steering file for use in Gmsh, the user would use the following script:

```
python gis2gmsh.py -n nodes.csv -b
boundary.csv -l lines.csv -h holes.csv -o
out.geo
```

As before, if there are no lines or holes files the user simply enters 'none' without the quotes as the -l and -h flags in the script. pputils assumes the refinement of the Gmsh mesh (i.e., how mesh grows from small to large elements) is controlled by the spacing of the nodes in the boundary, lines and/or hole files. This is only one way of specifying mesh growth in Gmsh. There are others as well. The interested user is directed to the Gmsh model documentation for more information on this subject.

The file out.geo is a steering file for Gmsh. To generate the quality mesh using the command line, the user could execute the following:

```
gmsh -2 out.geo
```

Where the -2 option specifies that the 2d triangular mesh is to be generated. Alternatively, the user is also given the option to launch the Gmsh GUI (which is available in Linux, Mac and Windows) and open the out.geo steering file. Immediately upon opening the Gmsh GUI, the user will see the skeleton geometry (that was originally created in QGIS). Once in the Gmsh GUI, the user can further edit the meshing parameters (select how mesh grows, place attractors, etc.). Please refer to the Gmsh user documentation for further details.

Execution of the Gmsh program produces the out.msh file, which is a Gmsh formatted mesh file. The following script in pputils converts it to the Adcirc mesh file:

```
gmsh2adcirc.py -i out.msh -o out_gmsh.grd
```

Where out.msh is the Gmsh generated mesh file, and the out_gmsh.grd is the same mesh in the Adcirc format.

Once the mesh is in the Adcirc format, the user can use adcirc2wkt.py script (see above) to create WKT format of the element polygons and node points for viewing the mesh within QGIS. Alternatively, the user can convert the Adcirc format to a *.dxf file using adcirc2dxf.py for mesh visualization using existing CAD based packages.

An example of using Gmsh is shown in Figure 3, where the generated mesh was used in the simulations nearshore sediment transport at Wheatley Harbour, Lake Erie.

D. Interpolation of quality mesh from a TIN

Suppose now that we have a TIN file (generated by Triangle), and converted to an Adcirc format (tin.grd) and also the quality mesh generated by Gmsh, also in Adcirc format (mesh.grd). The task now is to assign elevations to every node of the mesh.grd file from the tin.grd file.

The following script in pputils does just that:

```
python interp.py -t tin.grd -m mesh.grd -o
mesh_interp.grd
```

Where the mesh_interp.grd is the quality mesh with node z values interpolated from the TIN file. Matplotlib library is used to recreate the triangulation of the TIN, and assign the z values to quality mesh. Note that the mesh must entirely be within the boundary of the TIN. If this is not the case, a warning message is displayed at the prompt informing the user of this fact.

Of course, the user could easily have used Fudaa [16] or BlueKenue to carry out the task of interpolation as well.

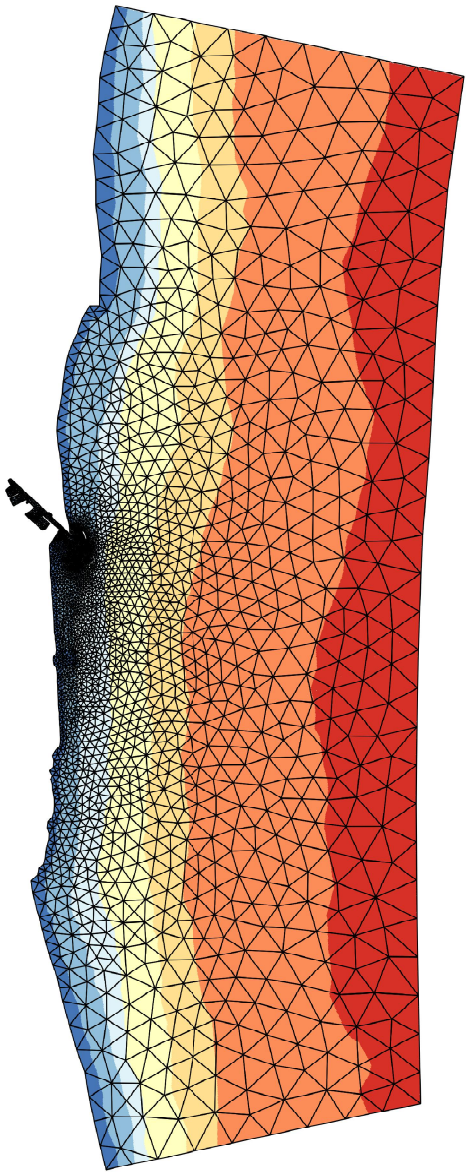

Figure 3: Nearshore mesh around Wheatley Harbour, Lake Erie, Ontario, Canada

E. Creation of Selafin files for use in TELEMAC simulations

Once the mesh_interp.grd Adcirc file is generated, the last step in the procedure is for the user to convert the Adcirc file to the Selafin format. There are at least three of the existing tools available to the TELEMAC user community that will do this. The user can:

1. Use Fudaa pre-processor, and convert Adcirc to Selafin mesh,
2. Use BlueKenue to import the Adcirc mesh, and save the imported mesh to Selafin format.
3. Use STBTEL program (part of the TELEMAC source code) to convert Adcirc to Selafin format.

There is an example in the validation cases on how to convert Adcirc mesh to Selafin mesh. After generating the Selafin files, the user then proceeds with numerical simulations using the TELEMAC modeling system.

IV. TELEMAC OUTPUT VISUALIZATION USING PPUTILS

Following completion of the numerical simulations using the TELEMAC system the user can port the output to QGIS using scripts in pputils. Please note however, that pputils output visualization is never intended to replace visualization that is typically done with BlueKenue, Fudaa, Davit, etc. At present time scripts in pputils are only meant to take select TELEMAC output (at key time steps), and generate publication style graphical output for use within the QGIS environment. Simulation output can then be overlaid with aerial photos, and annotated with labels, arrows, etc.

In order to get the TELEMAC output to QGIS, pputils relies on the Python parser scripts that are already part of the TELEMAC source code. For easy portability, the Python parser scripts (used for reading and writing TELEMAC data) have been copied and are included in the pputils distribution. The disadvantage of this is that a TELEMAC user will have these scripts in two places (one part of TELEMAC and one part of pputils). However, the advantage of including the Python parser scripts provides for easy installation, allows use of pputils without the need to update system path variables. A further advantage of including a copy of the Python parser scripts is that pputils can act as a standalone set of utilities, and could be open to more than just TELEMAC users. For example, those using Adcirc and/or SWAN could also benefit from them as well.

A. Displaying field variables

Displaying TELEMAC field variable (such as depths, velocity magnitudes, wave heights, etc.) with pputils is achieved by first probing the TELEMAC result file (assumed as result.slf) with the probe.py script, as follows:

```
python probe.py -i result.slf
```

Where -i represents the input file to probe. The output of the probe.py script simply tells the user what variables are saved in the result file, and what time steps are included. Most importantly, the probe.py script outputs the index of the variables and index of the time steps in the result file. An example of execution of the probe.py script on an existing output of the Telemac-2d simulation would be as follows:

```
The input file being probed: result.slf
Variables in result.slf are:
--------------------------------
     v     variable name
--------------------------------
0 --> VELOCITY U
1 --> VELOCITY V
2 --> WATER DEPTH
3 --> FREE SURFACE
4 --> BOTTOM
5 --> WIND ALONG X
6 --> WIND ALONG Y
7 --> COURANT NUMBER
number of records in input file : 25
--------------------------------
t        time (s)
--------------------------------
0 -->       0.0
1 -->    3600.0
2 -->    7200.0
3 -->   10800.0
     ......
24-->   86400.0
```

Suppose that the user wishes to display in QGIS the field variable free surface (-v index of 3) one hour into the simulation (-t index of 1 corresponding to simulation time 3600 s). The following pputils script would be executed:

```
python sel2asc.py -i result.slf -v 3 -t 1 -s
2.0 -o output.asc
```

Where -s parameter represents the grid spacing in meters (2 m grid spacing in above example). The sel2asc.py script parallels the adcirc2asc.py script, where Matplotlib reads the triangulation from the TELEMAC output file, and creates a gridded DEM file of the specified output variable for a specified time step using specified grid spacing.

B. Displaying vector variables

In order to display the vector variables within QIGS, the Field Renderer plugin [17] developed by Chris Crook out of New Zealand is used. The Field Renderer plugin requires the position (x and y) along with u- and v- components of the vector variable to be loaded as points within QGIS. In other words, it needs x, y, u, v points file. The pputils script extract.py is used to write this data from the TELEMAC results file. Suppose the user wishes to display within QGIS the velocity vectors for time step 24 (corresponding to simulation time 86,400 s), the following would be required:

```
python extract.py -i result.slf -v 0 1 -t 24
-o output.txt
```

Where -i is the Selafin result file to extract from, -v 0 1 are the indexes of the u- and v- component of the velocity vector (see output of probe.py above), -t is the time step to extract, and -o is the resulting text based output file containing x, y, u, v for each node in the Selafin file of the model results. The output.txt file is then loaded into QGIS, and the Field Renderer plugin is used to display the vector field within the QGIS.

An example of using the sel2asc.py and QGIS Field Renderer is shown in Figure 4 to display flood depths and velocity vectors during for a simulation of a flood wave of an urban area in London, Ontario, Canada using Telemac-2d.

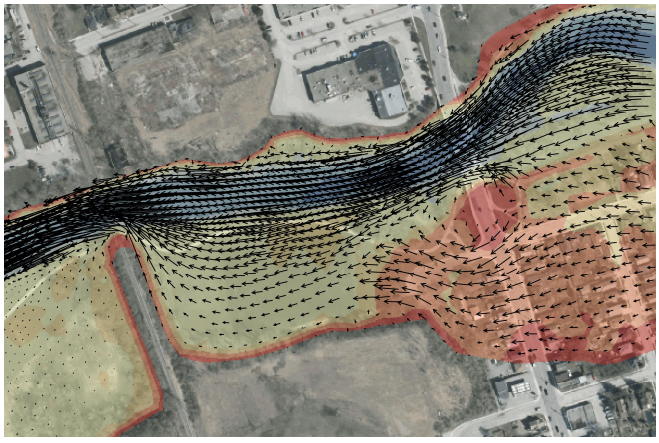

Figure 4: Flood depths and velocity vectors of an urban flood simulation using Telemac-2d at London, Ontario, Canada

C. *Future developments*

The post-processing of TELEMAC output using pputils are able to display single snapshots of field and vector variables for use in QGIS. At the present time, the intent of the post-processing features within pputils is to facilitate select graphical output for in QGIS for the preparation of reports and publication quality figures. To that end, having a small number of plots, overlaid with aerial photos and annotated with text, suffices.

Future developments may include different ways of visualizing 2d simulation model output in QGIS, and could include a full fledge post-processor with full animation control, extraction of cross sections, time series displays at select nodes, etc.

Note that pputils is work in progress, and will continually be refined and updated. The files included in the distribution also include a number of examples, which will assist the user in applying the code to their domains. pputils can be downloaded from the following links:

https://drive.google.com/open?id=0B7ZAQzSQW0q-NDlwbXBnbm1weUU

V. CONCLUSIONS

This paper presents a set of tools named pputils intended to be used for pre- and post-processing used in typical free surface flow modeling projects (mesh generation, digital surface creation, interpolation of mesh, display of model output). The main goal of the pputils project is to present the interested user with a set of tools that will allow the completion of an entire free surface flow modeling project from start to finish using only open source software. With this respect, pputils are able to achieve just that.

REFERENCES

[1] QGIS Development Team. QGIS Geographic Information System. Open Source Geospatial Foundation Project, 2015. http://qgis.osgeo.org.

[2] O. Conrad et al. “System for Automated Geoscientific Analyses (SAGA) v. 2.1.4”, Geoscientific Model Development, vol 8, pp. 1991-2007.

[3] GRASS Development Team, Geographic Resources Analysis Support System (GRASS) Software. Open Source Geospatial Foundation Project, 2015. http://grass.osgeo.org.

[4] GDAL. Geospatial Data Abstraction Library: Version 2.0.0, Open Source Geospatial Foundation, 2015. http://gdal.osgeo.org.

[5] Lutra Consulting Development Team, Crayfish QGIS plugin, 2015. http://www.lutraconsulting.co.uk/products/crayfish.

[6] D. Vetsch et al. “System Manuals of BASEMENT, Version 2.5”. Laboratory of Hydraulics, Glaciology and Hydrology (VAW), ETH Zurich, 2015. http://www.basement.ethz.ch.

[7] Uwe Merkel Consulting Engineers, TELEMAC Selafin Reader for QGIS, 2015. http://www.uwe-merkel.com/wordpress.

[8] J.R. Shewchuk, “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator”, in “Applied Computational Geometry: Towards Geometric Engineering” (M.C. Lin and D. Manocha, editors), volume 1148 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1996, pp. 203-222.

[9] S.W. Cheng, T.K. Dey, and J.R. Shewchuk, Delaunay mesh generation, CRC Press, 2012.

[10] C. Geuzaine and J.F. Remacle, “Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities”, International Journal for Numerical Methods in Engineering, 2009, vol 79, no 11, pp. 1309-1331.

[11] S. van der Walt, S.C. Colbert and G. Varoquaux, “The NumPy Array: A Structure for Efficient Numerical Computation”, Computing in Science & Engineering, 2011, vol 13, pp. 22-30.

[12] J.D. Hunter, “Matplotlib: A 2D Graphics Environment”, Computing in Science & Engineering, 2007, vol 9, pp. 90-95.

[13] National Research Council of Canada, BlueKenue: software tool for hydraulic modelers, Ottawa, Ontario, Canada, 2015, http://www.nrc-cnrc.gc.ca/eng/solutions/advisory/blue_kenue_index.html.

[14] Smile Consult GmbH, Janet, Gismo and Davit pre- and post processors for free surface numerical models, 2015, http://smileconsult.de.

[15] O. Gourgue, Pre- and post-processing of Unstructured Grids (PUG), 2015, http://www.oliviergourgue.net/pug.

[16] Fudaa, Object Oriented and Distributed Integration Platform For Scientific Codes, 2015, http://sourceforge.net/projects/fudaa.

[17] C. Crook, QGIS Vector Field Renderer plugin, 2013, https://github.com/ccrook/QGIS-VectorFieldRenderer-Plugin.