

CS CAPSTONE WINTER TERM PROGRESS REPORT

MARCH 17, 2020

CONVERTING FORTRAN TO PYTHON

PREPARED FOR

GENEVIEVE SEGOL

PREPARED BY

GROUP40

BRIAN HAMBLETON

DAKOTA ALTON

IAN BAND

IVAN HALIM

Abstract

This document is an overview of work completed, tasks left to be completed and problems encountered from January 7th to March 15th.

CONTENTS

1	Project Recap	2
2	Where We Stand	2
3	What Remains	2
4	Problems Encountered	3
5	Screenshots	3

1 PROJECT RECAP

Specialized software used in research is often found to be outdated due to its complexity and the limited number of people with expertise on the relevant subject who are also able to maintain the software. This can cause such programs, written in older languages, to lose support over time and become unusable by researchers. The purpose of this capstone project is to modernize one such piece of software to make it more readily available and accessible for researchers. The software in question, originally written in Fortran, performs analysis of flow and mass transport in unsaturated, partially saturated and fully saturated flow regions to be used for the evaluation of sites for waste disposal. The software would still be useful today if it were accessible to researchers, however, the mainframes that the original program was written for are no longer in use. The software in its current state must be modernized in order to be useful.

The goals of the project are as follows:

- Produce an updated version of the code that can be run on modern consumer hardware with minimal setup required.
- Increase the ease of data input to the program.
- Rewrite the data visualization portion of the project with modern graphics tools in Python 3.

2 WHERE WE STAND

We have achieved beta functionality for each of the previously specified goals. We have a version of the original Fortran code, ported to C, that can be run on modern systems. We have created a GUI that allows users to setup their scenario through manual input or via file upload. The parameters are then saved to data files to be read by the simulation. Lastly, we have a section of the GUI meant for visualizing the state of the mesh being evaluated in the simulation.

The updated version of the simulation, written in C, is capable of being run as a standalone application, but it can also be run from within the GUI we have developed. Calling it from the GUI is the primary method for running the simulation.

The GUI we have created has inputs for all fields required by the simulation. Additionally, it has sections for users to preview the initial state of the mesh being evaluated as well as a section for viewing the output of the simulation as text.

3 WHAT REMAINS

There are two remaining tasks to complete before the spring expo. We need to debug the simulation and we need to add callbacks to the main loop of the simulation that will allow us to view the mesh throughout the simulation's run time.

We suspect that the primary bug with our port of the simulation has to do with creating the cubic splines used to interpolate values. We are working on narrowing down the issue, but are fairly confident that we will be able to resolve the issue.

As for getting run time updates on the mesh, we have a robust system in the C code that will give the GUI a copy

of the mesh being evaluated. So seeing the mesh change through run time will be a matter of utilizing this system at the correct times. Accomplishing this task will pretty much fallout of debugging the simulation.

4 PROBLEMS ENCOUNTERED

The main issue we encountered with our design had to do with the version of Fortran used to write the original program. Our initial research into how we might solve the problem lead us to a tool known as F2PY which supports interfacing with Fortran 77 code. This tool should have been able to convert the original Fortran code into an intermediary format which would have allowed us to call the Fortran code from Python. This option did not pan out as it was determined the original code was written in Fortran IV (66), not Fortran 77 as the original program's documentation states, which is not supported by F2PY.

We overcame this issue by porting the Fortran code to C and wrapping the resulting C library in Python. There are three reasons for porting to C. First, speed gains are not lost when calling C from Python. Second, parameters to subroutines in Fortran are all by reference, meaning the subroutine can modify them and the change can be seen outside of the subroutine. C allows for this behavior where Python doesn't without writing hacky code. The third reason is that the GNU foundation offers a C library called GSL which we were able to use to mimic the functionality of the IMSL library used by the fortran code.

5 SCREENSHOTS

Basic Parameters				
Number of Nodes	52	Number of Elements	25	
Number of Material Groups	1	Number of Seepage Faces	0	
Half-Bandwidth (Flow)	4	Half-Bandwidth (Mass Transport)	4	
Pressure change criterion	1.000	Modified coefficient of fluid compressibility	0.000	
Initial Time Step (hours)	0.100	Time Step Multiplier	2.000	
Finite difference scheme	Implicit	Mass Transport Solutions	Transient	
Minimum Pressure Head	-260.000	Molecular diffusion constant	0.000	
Max Infiltration Rate	0.002	Convergence criteria for iteration	1.000	
Max Time Steps	28	Max Iterations Per Time Step	10	
Flow Solutions	Transient	Number of Mass Transport Solutions	1	
Time steps between changes in DELT	50			

Fig. 1: Basic Parameters Screen

Figure 1 shows the screen a user is met with when they first launch the application. This screen allows the user to input data that dictates how the simulation will run. That is it defines meta data for the simulation.

The Node Data Screen is titled "GS2" and has tabs for "Parameters", "Mesh", and "Simulation". The "Parameters" tab is active. On the left, there is a sidebar with buttons: "Import", "Export", "Basic Parameters", "Multipliers", "Nodes", "Node Types", "Elements", "Element Properties", "Material Data Points", and "Seepage Faces". The "Nodes" button is selected. The main area displays a table with the following columns: "Boundary Type", "X-Coordinate", "Y-Coordinate", "Initial Pressure", and "Initial Concentration". The table contains 15 rows, each with a dropdown menu for "Boundary Type" and input fields for the other four columns, all currently showing "0.000".

Fig. 2: Node Data Screen

Figure 2 shows a table that lets the user input data for nodes that model the transport problem being simulated.

The Element Incidence Screen is titled "GS2" and has tabs for "Parameters", "Mesh", and "Simulation". The "Parameters" tab is active. On the left, there is a sidebar with buttons: "Import", "Export", "Basic Parameters", "Multipliers", "Nodes", "Node Types", "Elements", "Element Properties", "Material Data Points", and "Seepage Faces". The "Elements" button is selected. The main area is divided into two sections. The left section, titled "Select Element", shows a list of elements: 1, 2, and 3, with element 2 selected. The right section, titled "Element 2", shows a "Material Group" dropdown set to "1". Below this, there is a section titled "Element Incidences" with a grid of 12 input fields labeled "Incidence 1" through "Incidence 12", all currently showing "0".

Fig. 3: Element Incidence Screen

Figure 3 shows the screen where a user can specify elements in the model. These elements have different properties that affects how analysis is done in different sections of the model.

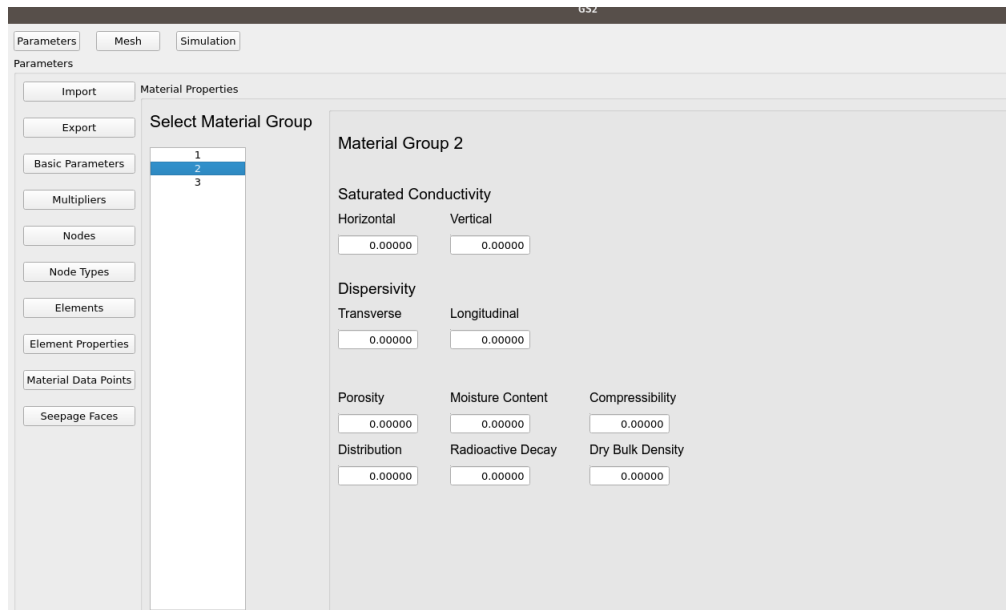


Fig. 4: Element Material Properties

Figure 4 shows the screen where a user can specify material group properties for different elements.

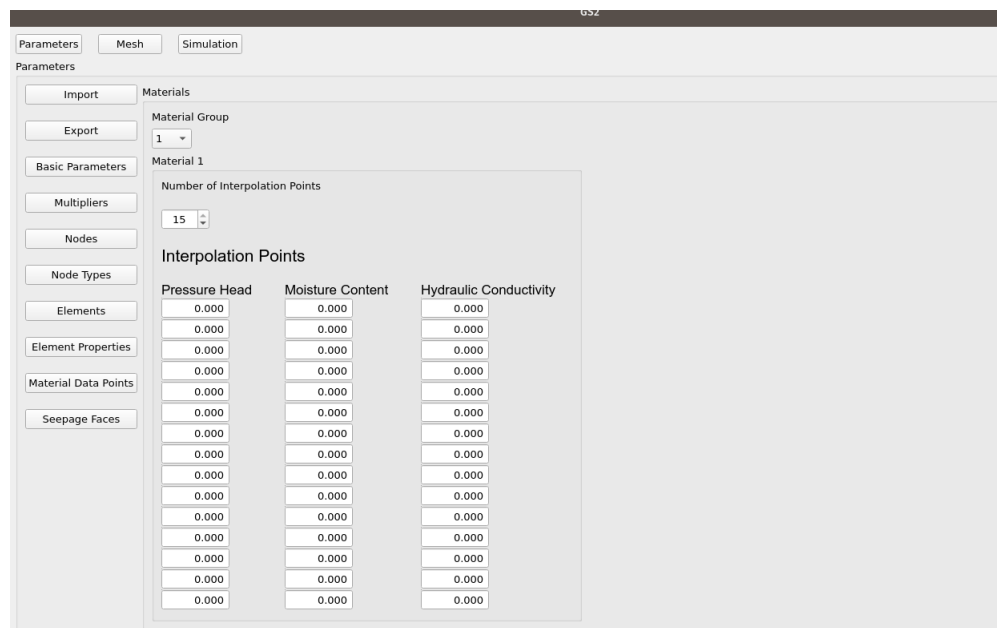


Fig. 5: Material Properties Screen

Figure 5 shows the screen where a user can specify material properties.

The screenshot shows the 'Multipliers' screen in a software application. The interface has a top bar with 'Parameters', 'Mesh', and 'Simulation' tabs. Below this is a sidebar with buttons for 'Import', 'Export', 'Basic Parameters', 'Multipliers', 'Nodes', 'Node Types', 'Elements', 'Element Properties', 'Material Data Points', and 'Seepage Faces'. The main area is titled 'Multipliers' and contains several input fields for various parameters, all set to 1.0000. The parameters are organized into groups: X-Coordinate, Y-Coordinate, Initial Pressure Head, Initial Concentration, Saturated Moisture Content, Medium Compressibility, Distribution Coefficient, Radioactive Decay Constant, Porosity, Density, Saturated Hydraulic Conductivity (Horizontal, Vertical), and Dispersivity (Transverse, Longitudinal).

Parameter	Value
X-Coordinate	1.0000
Y-Coordinate	1.0000
Initial Pressure Head	1.0000
Initial Concentration	1.0000
Saturated Moisture Content	1.0000
Medium Compressibility	1.0000
Distribution Coefficient	1.0000
Radioactive Decay Constant	1.0000
Porosity	1.0000
Density	1.0000
Saturated Hydraulic Conductivity (Horizontal)	1.0000
Saturated Hydraulic Conductivity (Vertical)	1.0000
Dispersivity (Transverse)	1.0000
Dispersivity (Longitudinal)	1.0000

Fig. 6: Multipliers Screen

Figure 6 shows the screen where the user can input multipliers for the simulation. These values affect how data is processed in the simulation.

The screenshot shows the 'Seepage Face' screen in the software application. The interface has a top bar with 'Parameters', 'Mesh', and 'Simulation' tabs. Below this is a sidebar with buttons for 'Import', 'Export', 'Basic Parameters', 'Multipliers', 'Nodes', 'Node Types', 'Elements', 'Element Properties', 'Material Data Points', and 'Seepage Faces'. The main area is titled 'Seepage Face' and contains input fields for 'Seepage Face ID' (set to 1) and 'Seepage Face 1'. The 'Seepage Face 1' section has two columns: 'Dirichlet Node Count' and 'Nuemann Node Count'. The 'Dirichlet Node Count' column has 8 rows, with the first row set to 8 and the others to 0. The 'Nuemann Node Count' column has 4 rows, with the first row set to 4 and the others to 0.

Dirichlet Node Count	Nuemann Node Count
8	4
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Fig. 7: Seepage Face Screen

Figure 7 shows the screen that allows the user to specify seepage faces for the model. Seepage faces affect how integration is computed.

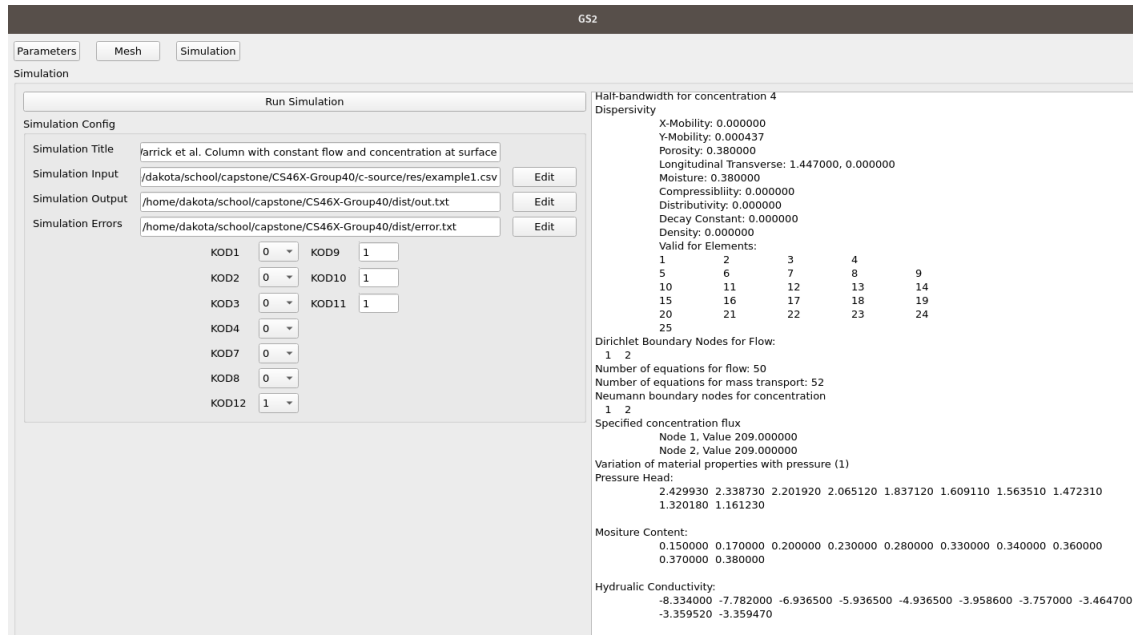


Fig. 8: Run Simulation Screen

Figure 8 allows the user to run the simulation. Additionally it has inputs that allows the user to control IO for the simulation. That is, from this screen the user can decide which data file to read into the simulation, where the output should be written, and where the simulation should log errors to.

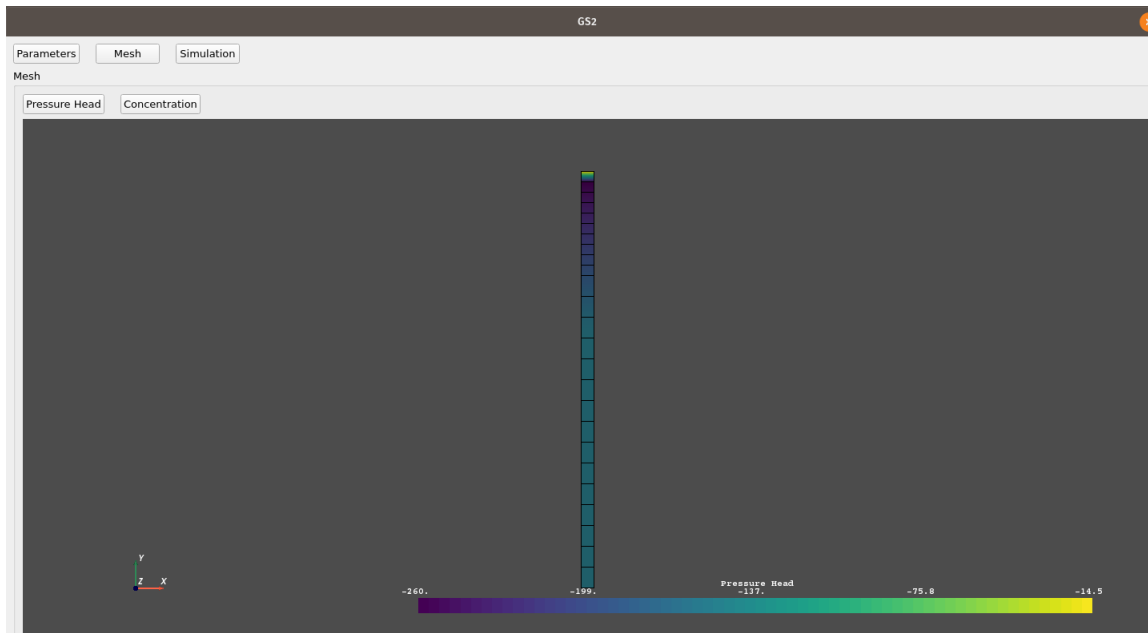


Fig. 9: Mesh Viewing Screen

Figure 9 shows the mesh of the model that the simulation is evaluating. In our applications current state, the mesh shown represents the initial state of the model.