a place of mind
THE UNIVERSITY OF BRITISH COLUMBIA

Earth, Ocean and Atmospheric Sciences
2020 – 2207 Main Mall
Vancouver, BC, V6T 1Z4 Canada

EOSC 213
Computational methods in geological engineering

Project rubric

Learning goals

- To develop a computational analysis of interest to geological engineering using the principles developed in EOSC 213.
- Be able to conceptualize a problem so that it becomes amenable to computational analysis.
- To be able to create or select appropriate algorithms and computational methods to solve the problem.
- To be able to implement the analysis / algorithm in an appropriately structured and documented python code and jupyter notebook.
- To present the results in a jupyter notebook.
- Where appropriate, utilize visualize to present results/data.

Project timelines (revised)

Tuesday March 5	Submit one-page proposal to Canvas (instructions to be posted to canvas)
Thursday March 7	Feedback on project proposals
~~Tuesday March 19~~	~~Submit progress report, including draft notebook~~
Thursday April 4	Submit final project as zip file containing notebook and supporting files (details to follow).
Thursday April 4	In class presentation of each project 4 minutes each MAX. (details to follow)
Thursday April 4	Submit self-assessment questions.

Project quality indicators

The table below lists good practice and qualitative criteria that we will be considering when we evaluate your projects.

Adapted from: Developing a project-based computational physics course grounded in expert practice https://aapt.scitation.org/doi/10.1119/1.4975381

Competency	Indicators
Physical Transcription	Where possible, analytical methods are first employed to understand the problem to the greatest extent possible, including identification of symmetries, length scales and timescales. The purpose of the calculation and desired results are clearly articulated.
Planning	The program to be written is broken into modules and functions that can be designed, tested and debugged independently. A suitable and efficient representation of the data, such as classes and data structures, is chosen appropriately for the algorithm. Relevant libraries, software packages and existing code are identified.
Implementation	The code can be easily understood and convinces the reader it works through careful commenting, descriptive variable and function names and validation of input. Coding standards are developed and obeyed amongst the implementation team. Comments document the physical principles, are in proportion to the complexity of the section, and identify input and output to functions.
Testing	The program is verified on test cases with known solutions identified in the planning process. Visualization is used to provide insight into whether the algorithm is working.
Running	Initial conditions are chosen judiciously. Output is organized and labeled and input parameters used in each run are recorded. Multiple runs, if necessary, are automated efficiently through scripts.
Visualization	Visualization is used to gain intuition regarding the output and to present final results in a compelling way.
Numerical Analysis	The source and nature of all approximations made are identified and their impact on the result discussed. The most significant sources of error are carefully analyzed and estimates of the error are given; ideally these are used to guide the algorithm, e.g., in refining the discrete representation.
Physical Analysis	Adherence to physical constraints (e.g., energy conservation) is verified. Possible improvements or alternate implementations are identified.

Self-assessment questions to be submitted with final project
from [http://dx.doi.org/10.1119/1.4975381]

(1) Describe your contribution to the project. Identify things that you yourself did.

(2) Overall, what grade would you give to your own contribution to the project? (*A—Mastery. I think I did this to a professional level; B—Solid understanding. I got this, though there may be still residual mistakes; C— Progress. I'm still working on learning this.*)
(3) How well did your team achieve the goals of the project? Explain briefly each member's contribution. Identify any challenges your team faced and how you overcame them.
(4) Overall, what grade to your team's project submission as a whole? (*A—Mastery. I think I did this to a professional level; B—Solid understanding. I got this, though there may be still residual mistakes; C—Progress. I'm still working on learning this.*)
(5) Did your team do anything over and above that required in the project description?
(6) If you have other comments on your group's project, please write them here.

Project grading rubric

a place of mind
THE UNIVERSITY OF BRITISH COLUMBIA

Earth, Ocean and Atmospheric Sciences
2020 – 2207 Main Mall
Vancouver, BC, V6T 1Z4 Canada

Overall project rubric				
Project depth / scope	Challenging topic using advanced methods that greatly exceed the scope of material presented in course notebooks.	Uses methods that somewhat exceed the scope of material presented in course notebooks, or apply the methods in novel ways.	Uses methods presented in the course notebooks to a new problem.	Incorrectly uses methods presented in the course to a trivial problem.
Applies an appropriate computational tool	Clearly communicates method / algorithm independent of programming software; selection of the most appropriate methods; implements the method independently	Can communicate the method in one context, but struggles to place it in a more general context; implements the code independently	Able to modify an existing code to address the solution to a similar problem.	Unable to determine how to solve the problem numerically; requires detailed and explicit direction.
Communicates the results of analysis in a meaningful way	Selects appropriate formats, figures, equations and animations (if appropriate) to clearly	Creates figures, tables, equations, but with some errors, ambiguities, labeling problems.	Able to create some rudimentary figures, tables, equations, but with poor reasoning to explain choice of data or presentation.	Does not create legible or properly labeled figures, tables or equations.

	communicate the result.				
Characteristic	**Outstanding**	**Above Average**	**Average**	**Below Average**	**Does not meet expectations**
Code / Notebook Grading Rubric					
Meets Computational Specifications	The program meets all of the computational specifications	The program produces the correct results and displays them correctly for almost all computational specifications	The program produces correct results for most computational specs, has a few bugs	The program is produces incorrect results, has several bugs	The program is does not work or has many bugs
Displays Output Correctly	The program displays results very clearly and intuitively, and meets all display specifications	The program displays results clearly and meets most of the display specifications	The program displays results clearly and meets many of the display specifications	The program does not display results clearly or does not meet most display specs	The program does not display results correctly and does not meet most display specs
Error Handling	The program checks for all error conditions and handles them appropriately	The program checks for most error conditions and handles them appropriately	The program checks for some error conditions and handles them appropriately	The program checks for few error conditions and doesn't handle them appropriately	The program does not check error conditions
Readability	The code / notebook is well organized and very easy to understand,	The code / notebook is pretty well organized, fairly easy to read,	The code / notebook has some organization, is a challenge to read,	The code / notebook is readable only by someone who	The code / notebook is poorly organized and very

	with clear comments both in-line and in headers	and has good comments	and has minimal comments	knows what it is supposed to do, has few comments	difficult to read, with no comments
Reusability	The code could be reused as a whole and each routine could be reused	Most of the code could be reused in other programs	Some parts of the code could be reused in other programs	A few parts of the code could be reused in other programs	The code is not organized for reusability
Documentation	Documentation is clear and well written, and clearly explains what the code does and how. It includes how to configure the system and how to use it correctly	Documentation is reasonably clear and mostly complete, and is useful in understanding the system and how to configure and use it correctly	Documentation is adequate, but not well written or thorough; configuration and user information is minimal	Documentation is does not explain the purpose or methods well, and does not help the reader understand the program or system; configuration and user documentation is inadequate	No separate documentation is provided
Testing	Test cases are thorough and systematic, well documented with expected and actual output	Test cases are thorough and systematic, known bugs are documented	Tests cover most representative cases, tests and known bugs are adequately documented	Test cases miss significant scenarios, and are poorly documented; bugs are poorly documented	Test cases are absent or very few, and are poorly documented or undocumented ; bugs not documented
Efficiency and Performance	The code is very efficient, system meets or exceeds	The code is fairly efficient, system	The code is naïve or brute force, system meets most	The code is brute force and unnecessarily long,	The code is huge and grossly inefficient, system

	all performance requirements	meets performance requirements	performance requirements	system meets some performance requirements	meets few or no performance requirements