2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023
The Harbor of Engineering Education for 130 Years
ASEE
Paper ID #38413

Work-in-Progress: A Pedagogical Unboxing of Reservoir Simulation with Python — Backward Design of Course Contents, Assessment, and Pedagogy (CAP)

Dr. Olatunde Olu Mosobalaje, Covenant University

Dr. Olatunde Mosobalaje holds a Chemical Engineering Bachelor degree from Ladoke Akintola University of Technology, Ogbomoso. He is an alumnus of the World Bank-funded African University of Science and Technology, Abuja, where he bagged a Petroleum Engineering MS degree in 2011. In 2019, he completed his Petroleum Engineering PhD program at Covenant University, Ota. He has been a faculty member at the Petroleum Engineering Department of Covenant University since February, 2013. In addition to being a registered engineer (COREN R68878), he is also a member of the Nigerian Society of Engineers, NSE (33597) as well as the Society of Petroleum Engineers, SPE (3495171).

In teaching petroleum engineering course modules, Dr. Mosobalaje adopts a balanced blend of analogical reasoning, concept visualization, field application and workflow coding as a pedagogy style. His recent enrolment in and completion of dozens of online courses (MOOC), delivered by world-class universities, has broaden his view of state-of-the-art teaching methods. As a testimonial of his pursuit of excellence in teaching, he recently received an award as the best teacher in the department from the Dean of Engineering, Covenant University.

Currently, Dr. Mosobalaje's research interest is in petroleum data analytics (PDA) as well as the deployment of machine learning (ML) tools to petroleum engineering applications. In research (and teaching, too), he leverages his proficiencies in open source platforms such as R and Python and associated libraries (ggplot, gstat, dplyr, scipy, numpy, matplotlib etc). In a modest way, his research products have helped to extend the functionality of some existing geostatistical routines. For his efforts, he recently received the Best Paper award in the 2020 International Conference on Applied Informatics, sponsored by Springer and featuring over 100 authours from 17 countries. Dr. Mosobalaje is open to post-doctoral fellowship/internship opportunities, especially in petroleum data analytics as well as engineering education.

Moses Olayemi, Purdue University, West Lafayette

Moses Olayemi is a Doctoral Candidate and Bilsland Dissertation Fellow in the School of Engineering Education at Purdue University. His research interests revolve around the professional development of engineering educators in low resource/post conflict settings and the design and contextualization of instruments to measure the impact of educational interventions. Research projects on these topics have and are currently being conducted in Nigeria, South Sudan, Iraq, Jordan, Kenya, and the US. His dissertation focuses on understanding the nuances and affordances of culturally relevant engineering education in Nigeria and the United States using a comparative case study methodology.

WIP: A Pedagogical Unboxing of Reservoir Simulation with Python: Backward Design of Course Contents, Assessment, and Pedagogy (CAP)

Abstract

Reservoir simulation is a state-of-the-art tool for reservoir performance prediction and remains an essential part of chemical and petroleum engineering undergraduate and post-graduate curricula. While the science of reservoir simulation is considered well-taught in academic programs, the literature suggests that students are still unaware of the foundational coding processes behind reservoir simulation software packages. Very little teaching attention has been given to the coding of the governing models and solutions to make these software packages, making reservoir simulation appear like a black box to students. Yet, the coding is indisputably the link between the science and the art. This paper stems from an ongoing project called Pedagogical Unboxing of Reservoir Simulation with Python (PURSIM-Py). This paper presents a proposed classroom adaptation of the project at a private University in Nigeria. Using backward design, in this paper, we present an alignment of the proposed course contents, assessment, and pedagogy (CAP) elements of the course. We propose this alignment be implemented in classes either as a stand-alone course or an accompanying lab to help students unbox reservoir simulation.

Introduction

Background

The need to minimize risks and maximize returns associated with alternative petroleum reservoir development options is the compelling motivation for the task of reservoir performance prediction. Such predictions are necessary for investment and operational decisions. Reservoir simulation is a state-of-the-art performance prediction tool that deploys physics, mathematics, programming, and reservoir engineering to formulate and implement reservoir fluid flow models as a computer program [1]. Thus, students are typically expected to have some command of programming, numerical methods, and reservoir engineering to be able to successfully understand reservoir simulations. Naturally, reservoir simulation is an essential part of the undergraduate and post-graduate curricula of petroleum engineering programs.

The science of reservoir simulation (i.e., the governing models and solutions) is well established [2] and considered well-taught in academic programs and specific training institutes around the world [3]. However, the literature surrounding workplace learning and onboarding reveals that companies continue to invest heavily in on-the-job learning and development for new and existing workers [4]. A recent report suggests that programming and coding are some of the most important skills for students to have in engineering in 2023 [5]. Yet, coding is arguably one of the most important links between the science and the art of reservoir simulation. Regrettably, this missing link has made reservoir simulation appear like an opaque black box to students.

This paper sets out as advocacy for an ongoing project (PURSIM-Py) to pedagogically unbox reservoir simulation via an interactive Python script that completely implements the workflow of a simple 3D oil reservoir simulator. Specifically, this paper presents a proposed Content, Assessment, and Pedagogy (CAP) alignment for an undergraduate classroom adaptation of the

project. This is also in an attempt to redesign the course modules of an undergraduate reservoir simulation class at a private University in the Federal Republic of Nigeria. This initiative is timely considering the recent developments in Nigeria's Core Curriculum and Minimum Academic Standards, CCMAS [6] which allows institutions to independently design 30% of their academic curricula.

The engineering profession is known for its problem-solving orientation [7]. Today, a modern approach to solving digital engineering problems systematically is to implement established workflows [8], [9]. These workflows may be expressed as algorithms. In practice, these algorithms are often coded as computer programs for subsequent implementation on modern computers. Incorporating computer coding of workflows into relevant engineering course modules is therefore of immense benefits. Apart from strengthening their problem-solving skills, exposing engineering students to such coding experience confers attributes of systems thinking, creativity and deeper understanding of processes on students [10]. Additionally, such exposure enhances research capabilities of graduate students as it offers the opportunity to experiment new ideas. Pedagogically, computer program scripts written for teaching and learning purposes could be deployed as tools to engage learners in simulation-based reflection on their performance in manual computations [11]. The teaching and learning of reservoir simulation, as a petroleum engineering course module, stands to benefit a lot from this workflow coding approach. The reservoir simulation body of knowledge is naturally presented in academic texts as a workflow of steps to be taken in developing a simulator computer program [1].

Motivation

The first author's classroom experiences in teaching Reservoir Modelling and Simulation (PET 524) as a final-year undergraduate course module at Covenant University provides a strong motivation to undertake this project. Over the past eight academic sessions, instructors have taught and assessed over 400 final-year undergraduate students. Historically, the delivery of the course module has focused primarily on the science of reservoir simulation. However, we have had a few instances of observing the performances of some students on the art of using commercial reservoir simulator software to execute their research projects. In those instances, we have observed students' lack of understanding of the inner workings of these software packages as well as their inability to interpret results therefrom. It is our considered opinion that this shortfall in understanding is a skill-gap and is arguably attributable to the fact that the students are not exposed to the underlying computer codes in those software packages.

Additionally, we are motivated to embark on this project as we observed the dearth of teaching resources and research interests in the pedagogical use of computer coding to enhance learners' understanding of reservoir simulation. While materials on general applications of computer coding (and machine learning) to petroleum engineering have been published in recent years [12], [13], there is a dearth of materials on specific topical applications such as is being advocated in this paper. Our ongoing efforts to pedagogically unbox reservoir simulation using the Python programming language is therefore in response to the foregoing. This paper advocates the initiative of that project and also presents the alignment of the content, assessment, and pedagogy (CAP) being proposed for the classroom adaptation of this initiative.

Project Overview

The tentative product of the ongoing PURSIM-Py project is an interactive Python script that completely implements the workflow of a simple 3D oil reservoir simulator. In the script, the reservoir fluid flow model is presented in a format that is deemed programmatically convenient. Thereafter, the entire simulation workflow is depicted as a flowchart. For pedagogical purposes, the script is presented in sequential modules (algorithms and codes). The modules address different sub-routines of the workflow such as data file preparation, import and formatting; reservoir discretization; gridblock ordering; preliminary computations; gridblock categorization; gridblock-level modeling; volumetric computations; and sensitivity analysis. Where necessary, the fine details of the sub-routines are further presented as flowcharts. The Python script is enriched with visual representations of some model parameters and intermediate results. Additionally, interactive web apps that would enable students to visualize the effects of model parameters on simulation results would be provided as supplements to the script. All materials pertaining to this project (scripts, data files, graphical objects, and web apps) would be available in an open-access online repository: https://github.com/TTOWG/Unboxing_Reservoir_Simulation_with_Python.

Overview of CAP Alignment

The content, assessment, and pedagogy alignment proposed for this course follows the recommendations of Streveler and Smith [14] and the stipulations of backward design by Wiggins and colleagues [15]. The authors proposed approaching course design from the perspective of the student, with a focus first on the intended learning outcomes of the course - essentially, focusing on what course instructors would like students to be able to know or do at the end of the course. By focusing on the intended learning outcomes, instructors can perform a backward design by identifying acceptable evidence before planning instruction. Thus, instructors are able to identify which concepts are enduring outcomes, important to know, or good to be familiar with (contents). Backward design also enables instructors to intentionally approach how they would achieve their learning outcomes in choosing appropriate pedagogies and assessments that determine if indeed students have achieved the intended learning outcomes.

Thus, the CAP alignment follows the order prescribed above. First, we introduce the contents of the course, beginning with a distinction between course objectives and learning objectives/outcomes. In this paper, we refer to course objectives as stipulated course goals, presented from the perspective of the instructor and the institution. These are the pedagogical actions that will be taken in the class and facilitated by the instructor to achieve the learning outcomes for the students. Conversely, learning outcomes or learning objectives are described from students' perspectives. These are the skills and knowledge that students are expected to achieve at the end of the course. After presenting the learning objectives, we proceed to categorize them in order of importance into curricular priorities. Fundamentally, enduring outcomes refer to concepts and skills that we would like our students to possess many years after the course is over. Simply put, we would like students of the PET 524 course to be able to know these concepts and perform these skills if they took nothing else from the course. Consequently, the important-to-know and good-to-be-familiar-with concepts are of lesser importance but also valuable.

Next, we present a mapping of the learning objectives to the curricular priorities and proceed to present the taxonomy of said objectives. For this task, we applied Bloom's taxonomy framework modified by Anderson and Krathwohl [16]. This presents the learning objectives separated into

cognitive process dimensions and cognitive knowledge complexities. For the assessment, we developed a worksheet for the learning objectives using backward design principles. Each learning objective is mapped to an assessment task to be performed by the students. Acceptable evidence is also listed in the sections. The goal of this exercise is to create rubrics for the students to understand specifically what is expected of them in the assessments [17]. Finally, the pedagogy section details the steps to be taken by the instructor to achieve the intended learning outcomes. Thus, this section presents information about the learning environment, guidance in terms of course objectives (from the instructor's perspective and supported by the institution), learning sequence, learning resources, and instructional strategies. We invite comments and suggestions on the appropriateness of this CAP alignment.

Content

The "C" in the CAP model refers to the course contents. We begin by first identifying the learning objectives of the course. Essentially, these consist of the intended skills and knowledge that students of the course are expected to be able to perform and know at the end of the course. Thus, we interpret these as the expected skills and knowledge to be gained by the students.

Learning Objectives

It is expected that at the end of the course, students should be able to:

1. Outline the ordered computational tasks that constitute the operation of a reservoir simulator software.
2. Deploy Python's functionalities to automate the tasks of reservoir discretization and gridblock ordering.
3. Validate the discretization parameters on the visual depiction of the discretized reservoir model.
4. Develop Python scripts to compute preliminary simulation parameters.
5. Create a custom Python function that can classify gridblocks with respect to their interactions with reservoir boundaries.
6. Set up nested repetitive loops in Python to transverse gridlocks in all relevant axes of the discretized reservoir.
7. Develop Python scripts to execute core simulation tasks such as inter-block flow, boundary conditions, well model, coefficient matrix, RHS vector and matrix solution to linear models.
8. Develop a Python script to execute the MBE-based volumetric computation workflow.
9. Relate changes in reservoir model parameters (inputs) to changes in reservoir performance parameters (outputs).

Curricular Priorities

Following the recommendations of backward design, we separated these objectives in order of importance or curricular priorities. The rationale for ordering curricular priorities is to emphasize what knowledge or skills are essential, important, or peripherally good to be familiar with. Thus, these objectives are listed as follows curricular priorities listed as enduring outcomes, important-to-know, and good-to-be-familiar-with concepts. A summary of the curricular priorities is provided in figure 1.

Enduring Outcomes (EO)

- EO1: The logical flow of the sequence of computational tasks of a reservoir simulator software.
- EO2: The use of Python's functionalities to implement the reservoir simulation workflow.
- EO3: The sensitivity of reservoir performance predictions to values of reservoir model parameters.

Important to Know (ITK)

- ITK1: Python implementation of reservoir discretization and gridblock ordering

- ITK2: Computation of preliminary simulation parameters, in Python.
- ITK3: Use of Python's looping structure to traverse gridblocks.
- ITK4: Use of Python to implement block-level modelling.
- ITK5: Use of Python to execute MBE-based volumetric computations.

Good to be familiar with (BFW)

BFW1: Flowchart of a reservoir simulator's operations

BFW2: Visualization of the discretized reservoir model.

BFW3: Gridblock categorization algorithm and implementation in Python.

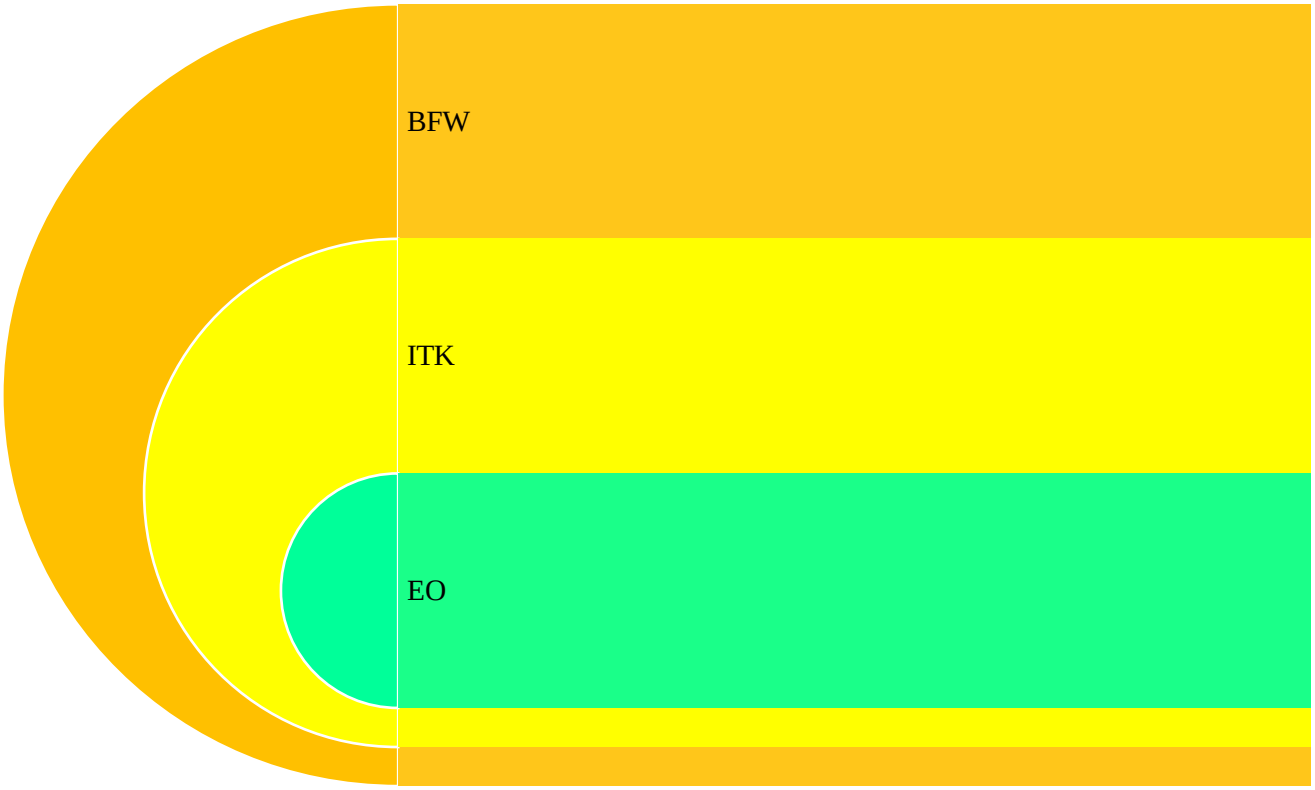

Figure 1. Curricular Priorities. Enduring outcomes are at the core of the rings in green. Important-to-know concepts are in the yellow ring, and good-to-be-familiar-with concepts are in the orange ring.

In the table below, we attempt to map each learning objective to the curricular priorities.

Table 1: Mapping Learning Objectives to Curricular Priorities

Learning Objective	Curricular Priorities
1. Students should be able to outline the ordered computational tasks that constitute the operation of a reservoir simulator software.	EO1: The logical flow of the sequence of computational tasks of a reservoir simulator software. BFW1: Flowchart of a reservoir simulator's operations
2. Students should be able to deploy Python's functionalities to automate the tasks of reservoir discretization and gridblock ordering.	EO2: The use of Python's functionalities to implement the reservoir simulation workflow. ITK1: Python implementation of reservoir discretization and gridblock ordering.
3. Students should be able to validate the discretization parameters on the visual depiction of the discretized reservoir model.	BFW2: Visualization of the discretized reservoir model.
4. Students should be able to develop Python scripts to compute preliminary simulation parameters.	EO2: The use of Python's functionalities to implement the reservoir simulation workflow. ITK2: Computation of preliminary simulation parameters, in Python.
5. Students should be able to create a custom Python function that can classify gridblocks with respect to their interactions with reservoir boundaries.	EO2: EO2: The use of Python's functionalities to implement the reservoir simulation workflow. BFW3: Gridblock categorization algorithm and implementation in Python.
6. Students should be able to set up nested repetitive loops in Python to transverse gridlocks in all relevant axes of the discretized reservoir.	EO2: The use of Python's functionalities to implement the reservoir simulation workflow. ITK3: Use of Python's looping structure to traverse gridblocks.
7. Students should be able to develop Python scripts to execute core simulation tasks such as inter-block flow, boundary conditions, well model, coefficient matrix, RHS vector and matrix solution to linear models.	EO1: The logical flow of the sequence of computational tasks of a reservoir simulator software. EO2: The use of Python's functionalities to implement the reservoir simulation workflow. ITK4: Use of Python to implement block-level modelling.
8. Students should be able to develop a Python script to execute the MBE-based volumetric computation workflow.	EO2: The use of Python's functionalities to implement the reservoir simulation workflow. ITK5: Use of Python to execute MBE-based volumetric computations.
9. Students should be able to relate changes in reservoir model parameters (inputs) to changes in reservoir performance parameters (outputs).	EO3: The sensitivity of reservoir performance predictions to values of reservoir model parameters.

Table 2: Taxonomy of the Learning Outcomes

	Remember	Understand	Apply	Analyze	Evaluate	Create
Factual knowledge						
Conceptual Knowledge		LO1: Outline the ordered computational tasks that constitute the operation of a reservoir simulator software.		LO9: Relate changes in reservoir model parameters (inputs) to changes in reservoir performance parameters (outputs).		
Procedural Knowledge			LO2: Deploy Python's functionalities to automate the tasks of reservoir discretization and gridblock ordering. LO6: Set up nested repetitive loops in Python to transverse gridlocks in all relevant axes of the discretized reservoir.			LO4: Develop Python scripts to compute preliminary simulation parameters. LO5: Create a custom Python function that can classify gridblocks with respect to their interactions with reservoir boundaries. LO7: Develop Python scripts to execute core simulation tasks such as inter-block flow, boundary conditions, well model, coefficient matrix, RHS vector and matrix solution to linear models. LO8: Develop a Python script to execute the MBE-based volumetric computation workflow.
Metacognitive knowledge				LO3: Validate the discretization parameters on the visual depiction of the discretized reservoir model.		

Assessment

Table 3: Assessment Worksheet

Learning Objective	Assessment
LO1: Students should be able to outline the ordered computational tasks that constitute the operation of a reservoir simulator software.	**Task:** students would write an in-class quiz
	Acceptable evidence of this learning objective will be: ▪ Coherent responses to questions on the essence, rationality and interrelatedness of the following tasks in reservoir simulation: ✓ Data input ✓ Reservoir discretization and gridblock ordering ✓ Simulation parameters computation ✓ Looping through gridblocks ✓ Gridblock-level modelling ✓ MBE-based computations
LO2: Students should be able to deploy Python's functionalities to automate the tasks of reservoir discretization and gridblock ordering.	**Task:** Students would undertake a take-home programming assignment to be submitted via GitHub
	Acceptable evidence of this learning objective will be: ▪ A re-adaptation (transfer) of the class-delivered discretization script to a different scenario; for examples: ✓ Point-centered discretization versus block-centered discretization ✓ Engineering ordering versus natural ordering ✓ 2-D grid versus 3-D grid
LO3: Students should be able to validate the discretization parameters on the visual depiction of the discretized reservoir model.	**Task:** students would write an in-class quiz
	Acceptable Evidence of this learning objective will be: ▪ A successful attempt at correlating the following input data and discretization parameters to graphical depictions on the discretized model. ✓ Reservoir dimensions ✓ Block dimensions ✓ Number of gridblocks in relevant axis.

Learning Objective	Assessment
	Block ordering scheme adopted
LO4: Students should be able to develop Python scripts to compute preliminary simulation parameters.	**Task:** Students would participate in an in-class hands-on exercise.
	Acceptable evidence of this learning objective will be: ▪ Attainment of error-free Python statements for the purpose of computing the following: ✓ Gridblock areas and volumes ✓ STOIP ✓ Compressibility ✓ Coefficients
LO5: Students should be able to create a custom Python function that can classify gridblocks with respect to their interactions with reservoir boundaries.	**Task:** Students would undertake a take-home programming assignment to be submitted via GitHub. Reworking an incomplete, error-prone version of the class-delivered function script.
	Acceptable evidence of this learning objective will be: ▪ A successful attempt to identify and fix bugs (errors) in the function script. Completion of the script to handle all possible gridblock categories.
LO6: Students should be able to set up nested repetitive loops in Python to transverse gridlocks in all relevant axes of the discretized reservoir.	**Task:** students would write an in-class quiz
	Acceptable evidence of this learning objective will be: ▪ Coherent responses to questions on the following contexts: ✓ Frequency of loops execution: the fastest loop; the slowest loop ✓ Inter-relationship of loops: inner loop; outer loop etc. ✓ Loop counters: initialization, incrementation and termination criteria. ✓ Axis of loops.
LO7: Students should be able to develop Python scripts to execute core simulation	**Task:** Students would write a mid-term test. Reworking an incomplete version of the class-delivered function script.

Learning Objective	Assessment
tasks such as inter-block flow, boundary conditions, well model, coefficient matrix, RHS vector and matrix solution to linear models.	**Acceptable evidence** of this learning objective will be: ▪ Completion of the script ▪ A re-adaptation (transfer) of the class-delivered script to a different scenario within the following context: ✓ Boundary conditions ✓ Well models.
LO8: Students should be able to develop a Python script to execute the MBE-based volumetric computation workflow.	**Task:** Students would undertake a group project: ▪ Class presentation ▪ Peer reviews
	Acceptable evidence of this learning objective will be: ▪ A successful re-development of the class-delivered MBE script as a stand-alone subroutine with the following capabilities: ✓ works independent of the main simulation script ✓ accepts block pressure matrix and reservoir parameters as inputs ✓ terminates at attainment of bubble point Group members coherent responses to questions from peers.
LO9: Students should be able to relate changes in reservoir model parameters (inputs) to changes in reservoir performance parameters (outputs).	**Task:** Students would undertake a take-home assignment: a short essay on the sensitivity of performance parameters to changes in reservoir model parameters.
	Acceptable evidence of this learning objective will be: ▪ Production of performance prediction for a case-study ▪ Coherent discussions on the impact of impact of various reservoir model parameters on predicted performance ▪ Use of visual tools such as spider and Tornado plots.

Pedagogy

We interpret the course objectives as the content intended to be introduced to the students. Essentially, these are responsibilities of the instructor and they are as follows:

Instructor's Course Objectives

1. Present the operation of a reservoir simulator as a flowchart of sequential computational tasks, using *schemedraw* Python library.
2. Showcase a Python implementation of reservoir discretization and gridblock ordering schemes.
3. Depict the discretized reservoir model with static and interactive graphics using *matplotlib* and *plotly* Python libraries.
4. Present simple Python statements that compute the preliminary simulation parameters.
5. Introduce and implement a gridblock categorization algorithm, as a precursor to block-level modeling.
6. Describe Python looping structures as a means of programmatically traversing through layers, rows and columns of a discretized model.
7. Present a Python implementation of block-level model concepts: inter-block flow, boundary conditions, well model, coefficient matrix, RHS vector; leading to the obtainment of gridblock pressure solution.
8. Showcase a Python implementation of MBE-based volumetric computations as the ultimate step in reservoir performance prediction.
9. Analyze the sensitivity of simulation outputs (performance parameters) to various inputs (reservoir model parameters).

Learning Environment

- This module is recommended to be treated as lab component of the traditional undergraduate module on Reservoir Modeling and Simulation.
- Hence, the learning environment should be a computer laboratory with a large display monitor and sufficient PCs to go all students.
- Also, the module can be treated as a stand-alone course.
- Base Python and Jupyter Notebook installations are required on each PC. In addition, Python libraries such as *numpy*, *matplotlib*, *pandas*, *scipy*, *plotly* and *schemedraw* are all required on the PCs.
- To enhance collaborative learning among students (team members) as well as use of technology, it is recommended that Distributed Version Control tools (GitHub) should be used in this module. In actual fact, a GitHub repository should be set up for this module where all course resources would be made available. Specifically, the PCs should have the GitHub Desktop app and each student should have a GitHub account with a forked and cloned copy of the repository.

Learning Resources

- First, the main resource available for this module is the Python script presently being developed as the main product of this project. The interactive Python script completely implements the workflow of a simple 3D single-phase oil reservoir simulator.
- Additionally, interactive web apps that would enable students to visualize the effects of model parameters on simulation results are provided as supplementary resources.
- All these resources (scripts, data file, graphical objects and web apps) are freely available in the public GitHub repository for this project: https://github.com/TTOWG/Unboxing_Reservoir_Simulation_with_Python
- In order to cascade the knowledge acquisition for students, a course module on basic Python programming with applications in petroleum engineering is recommended as a pre-requisite for this course module. For this purpose, lecture notes and demo scripts developed in our Computer Applications in Petroleum Engineering (PET 328) module are freely available at https://github.com/TTOWG/PET328_2021_Class.

Lesson Sequence

1. Reservoir Simulator Workflow - presentation of major workflow steps
2. Input Data File
   - Data preparation with the *.csv* template
   - Data importation from *.csv* into Python as DataFrame
   - DataFrame formatting
3. Reservoir Discretization and Visualization
   - Computation of gridblock dimensions
   - Generation of gridblock ordering data
   - 3D visualization of discretized model
   - Gridblock categorization
     - Stating the need for categorization
     - Establishing the basis of categorization
     - Scripting categorization functions
   - Visualization of gridblock categories
     - Static color-coded graphics using *matplotlib*
     - Dynamic interactive plots – using *plotly*
4. Simulation Parameter Computations
   - Gridblock cross-sectional area, in x, y, z directions
   - Gridblock bulk volume, in ft^3 and bbl.
   - Reservoir STOIIP

- Gridblock STOIIP
- Effective compressibility
- Inter-block flow transmissibilities, in x, y, z directions
- Setting flow model coefficients

5. Gridblock-Level Modelling
   - Simulation loops through the discretized model: variables and counters
   - Programmatically-convenient presentation of the governing equation
   - Loop flowchart
   - Indices of the current block
   - Indices of the neighboring blocks:
     - Formulating the algorithm
     - Scripting the function
   - Implementation of boundary condition
   - The Coefficient matrix
   - Well-blocks identification and modelling
   - The RHS vector
   - Block pressures matrix
6. Volumetric Computations
   - The MBE model
   - Computation flowchart
   - PVT Updating
   - Performance prediction
7. Loop termination mechanism
8. Output Aggregation and Export
9. Sensitivity Analysis

Instructional Strategies

The primary mode of instruction recommended for teaching this content is classroom lectures. At any given stage in the content, the instructor should focus on guiding the students to establish the logic of the workflow at that stage. The use of analogical reasoning is highly recommended in this regard [18]. For example, the typical grid-like seating arrangement in classrooms lends itself as an analogy the gridblocks in a discretized reservoir. Once the logic is established, a class demonstration of the coding of that stage should be carried out with the active learning mode fully activated. Additionally, the simulation-based learners reflection activity is recommended as a

supplementary strategy. Case-study data provided with the Python script could also be leveraged for learners' independent experimentation.

Conclusion

This paper has laid out the underpinnings of an ongoing project aimed at pedagogically unboxing reservoir simulation with the Python programming language. The anecdotal evidence that motivated this project has been presented. The project's framework and expected pedagogical products has also been previewed. At the core of the paper, the content, assessment and pedagogy (CAP) for classroom adaptation of the Python script has been designed using the backward design approach. Being a work in progress, we anticipate feedback suggestions on the alignment of the CAP design presented in this paper.

References

[1] J. H. Abou-Kassem, M. R. Islam, and S. M. Farouq Ali, "The engineering approach versus the mathematical approach in developing reservoir simulators," 2020, pp. 373–396. doi: 10.1016/B978-0-12-819150-7.00010-4.

[2] L. P. Dake, *Fundamentals of Reservoir Engineering*. Elsevier, 1983.

[3] P. Andrews and J. Playfoot, *Education and Training for the Oil and Gas Industry: Building A Technically Competent Workforce*. Elsevier, 2014.

[4] R. A. Berkley and D. M. Kaplan, *Strategic training and development*. Los Angeles : London: SAGE, 2020.

[5] LinkedIn Learning, "2023 Workplace Learning Report: Building the Agile Future," LinkedIn, San Diego, CA, 2023. Accessed: Feb. 28, 2023. [Online]. Available: https://learning.linkedin.com/content/dam/me/learning/en-us/pdfs/workplace-learning-report/LinkedIn-Learning_Workplace-Learning-Report-2023-EN.pdf

[6] National Universities Commission, "Engineering and Technology," *NUC CCMAS*, Dec. 27, 2022. https://nuc-ccmas.ng/engineering-and-technology/ (accessed Feb. 27, 2023).

[7] B. V. Koen, *Discussion of the method: Conducting the engineer's approach to problem solving*. New York, NY: Oxford University Press, 2003.

[8] S. B. Davidson and J. Freire, "Provenance and scientific workflows: challenges and opportunities," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, New York, NY, USA, Jun. 2008, pp. 1345–1350. doi: 10.1145/1376616.1376772.

[9] T. Ertekin and Q. Sun, "Artificial Intelligence Applications in Reservoir Engineering: A Status Check," *Energies*, vol. 12, no. 15, Art. no. 15, Jan. 2019, doi: 10.3390/en12152897.

[10] F. Kalelioğlu, "A new way of teaching programming skills to K-12 students: Code.org," *Comput. Hum. Behav.*, vol. 52, pp. 200–210, Nov. 2015, doi: 10.1016/j.chb.2015.05.047.

[11] S. J. Dickerson and R. M. Clark, “Simulation-Based Reflection in a Digital,” *Comput. Educ. J.*, vol. 12, no. 3, 2021.

[12] H. Belyadi and A. Haghighat, “Chapter 1 - Introduction to machine learning and Python,” in *Machine Learning Guide for Oil and Gas Using Python*, H. Belyadi and A. Haghighat, Eds. Gulf Professional Publishing, 2021, pp. 1–55. doi: 10.1016/B978-0-12-821929-4.00006-8.

[13] P. Bangert, “Chapter 1 - Introduction,” in *Machine Learning and Data Science in the Oil and Gas Industry*, P. Bangert, Ed. Gulf Professional Publishing, 2021, pp. 1–11. doi: 10.1016/B978-0-12-820714-7.00001-7.

[14] R. A. Streveler and K. A. Smith, “Opinion: Course Design in the Time of Coronavirus: Put on Your Designer’s CAP,” *Adv. Eng. Educ.*, vol. 8, no. 4, 2020, Accessed: Feb. 08, 2022. [Online]. Available: https://eric.ed.gov/?id=EJ1287320

[15] G. Wiggins, G. P. Wiggins, and J. McTighe, *Understanding by Design*. ASCD, 2005.

[16] L. W. Anderson and D. R. Krathwohl, *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom’s Taxonomy of Educational Objectives*. Longman, 2001.

[17] Y. M. Reddy and H. Andrade, “A review of rubric use in higher education,” *Assess. Eval. High. Educ.*, vol. 35, pp. 435–448, 2010, doi: 10.1080/02602930902862859.

[18] M. S. Vendetti, J. Bryan, B. J. Matlen, L. E. Richland and S. A. Bunge, “Analogical Reasoning in the Classroom: Insights from Cognitive Science” *Mind, Brain, and Education*, 9, 100-106. https://doi.org/10.1111/mbe.12080