



Faculty of Computer Science and Information Technology

3D Geological Borehole Model Using Python

Tahmid Muhtashim Nafy

**Bachelor of Computer Science with Honours
(Software Engineer)**

3D GEOLOGICAL BOREHOLE MODEL USING PYTHON

TAHMID MUHTASHIM NAFY

This project is submitted in partial fulfillment of the requirements for the degree of
Bachelor of Computer Science and Information Technology

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK

2021

3D GEOLOGICAL BOREHOLE MODEL USING PYTHON

TAHMID MUHTASHIM NAFY

Projek ini merupakan salah satu keperluan untuk Ijazah
Sarjana Muda Sains Komputer dengan Kepujian

Fakulti Sains Komputer dan Teknologi Maklumat
UNIVERSITI MALAYSIA SARAWAK

2021

UNIVERSITI MALAYSIA SARAWAK

THESIS STATUS ENDORSEMENT FORM

TITLE 3D GEOLOGICAL BOREHOLE MODEL USING PYTHON

ACADEMIC SESSION: 20/21 semester 2

(CAPITAL LETTERS)

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order to develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [or for the purpose of interlibrary loan between HLI]
5. ** Please tick (✓)

☐

CONFIDENTIAL (Contains classified information bounded by the OFFICIAL SECRETS ACT 1972)

☐

RESTRICTED (Contains restricted information as dictated by the body or organization where the research was conducted)

☒

UNRESTRICTED

Tahmid

(AUTHOR'S SIGNATURE)

Permanent Address

Bhalamary, Milon Bazar, Madargonj,

Jamalpur, Bangladesh.

16/06/2021

Date:

Validated by

(SUPERVISOR'S SIGNATURE)

Date:

Note * Thesis refers to PhD, Master, and Bachelor Degree

** For Confidential or Restricted materials, please attach relevant documents from relevant organizations / authorities

DECLARATION

I hereby declare that this final year project entitled “3D Geological Borehole Model Using Python” was carried out by me for the degree of Bachelor of Computer Science with Honors under the supervision of Dr. Bong Chih How, Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak.

The content of this project is not plagiarized from other students’ work or any published journals, articles, and websites. The sources which I have made use of are acknowledged at the respective place in the text.

tahmid

.....
(Tahmid Muhtashim Nafy)

13/01/2021

ACKNOWLEDGEMENT

In this Final Year Project period, many people helped me in the process of completing this project. I would like to express my gratitude to my final year project (FYP) supervisor Dr. Bong Chih How, for his guidance and organize a workshop to let me understand more about how the FYP report should be done. This project is a tough task for me if there is no support from my supervisor because it requires a follow-up with the client from another company for this project. Finally, I would like to thank my friends, my sister, and my family, especially my dad and mom for their assistance and moral encouragement in the process of completing the final year project. Their comments and feedbacks have helped me to improve my final year project to become more resourceful. Throughout the process, they have given me motivation and support to move on.

ABSTRACT

The portrayal of subsurface constructions is a fundamental part of a wide assortment of geoscientific examinations and applications, on raw material examinations, to Geo sequestration, just as numerous parts of geoscientific exploration and applications in topographical studies. A wide scope of strategies exists to create geographical models. Nonetheless, the amazing techniques are behind a paywall in costly business packages. In the project, represent a full open-source geomodeling method based on an understood potential-field addition approach. The interpolation calculation is practically identical to executions in business bundles and equipped for developing complex full 3D geological models, including a 2D cross-section model with multi-layer subsurface. This algorithm is carried out in the programming language Python, utilizing an exceptionally productive hidden library for proficient Theano that empowers an immediate execution on GPUs. The usefulness can be isolated into the center angles needed to create 3-D geographical models and extra resources for cutting-edge logical examinations.

This proposed system is only available on a web application where it is much easier to view on a personal computer the data was given in CSV format. With the help of the borehole log, GemPy can be able to visualize the 2D model of cross-section, faults, 3D geological model as well. In this paper, the documentation of the entire development of the 3D geological borehole model has been described including introduction, literature review, methodology, testing & implementation, and conclusion & future work. And at the end of the entire project, the final prototype will be handed over to the client.

ABSTRAK

Penggambaran konstruksi bawah permukaan adalah bahagian mendasar dari berbagai macam pemeriksaan dan aplikasi geosains, pada pemeriksaan bahan mentah, hingga penyerapan Geo, sama seperti banyak bahagian eksplorasi dan aplikasi geosains dalam kajian topografi. Skop strategi yang luas ada untuk membuat model geografi. Walaupun begitu, teknik yang luar biasa ada di belakang paywall dalam pakej perniagaan yang mahal. Dalam projek tersebut, mewakili kaedah geomodeling sumber terbuka penuh berdasarkan pendekatan penambahan bidang-potensi yang difahami. Pengiraan interpolasi hampir sama dengan pelaksanaan dalam kumpulan perniagaan dan dilengkapi untuk mengembangkan model geologi 3D penuh yang kompleks, termasuk model keratan rentas 2D dengan permukaan bawah berbilang lapisan. Algoritma ini dilakukan dalam bahasa pemrograman Python, menggunakan perpustakaan tersembunyi yang sangat produktif untuk Theano yang mahir yang memberi kuasa pelaksanaan segera pada GPU. Kegunaan dapat diasingkan ke sudut tengah yang diperlukan untuk membuat model geografi 3-D dan sumber tambahan untuk pemeriksaan logik canggih.

Sistem yang dicadangkan ini hanya tersedia di aplikasi web di mana lebih mudah untuk melihat pada komputer peribadi data diberikan dalam format csv. Dengan bantuan log borehole, GemPy dapat menggambarkan model keratan rentas 2D, kesalahan, model geologi 3D juga. Dalam makalah ini, dokumentasi keseluruhan pengembangan model lubang geologi 3D telah dijelaskan termasuk pengenalan, tinjauan literatur, metodologi, pengujian & pelaksanaan, dan kesimpulan & karya masa depan. Dan pada akhir keseluruhan projek, prototaip akhir akan diserahkan kepada pelanggan.

TABLE OF CONTENTS

| | |
|---|------|
| DECLARATION | v |
| ACKNOWLEDGEMENT | vi |
| ABSTRACT | vii |
| ABSTRAK | viii |
| LIST OF TABLES | xiii |
| LIST OF FIGURES | xiv |
| LIST OF ABBREVIATION | xv |
| Chapter 1: Introduction | 1 |
| 1.1 Introduction | 1 |
| 1.2 Problem Statement | 2 |
| 1.3 Scope | 3 |
| 1.4 Objective | 3 |
| 1.5 Methodology | 3 |
| 1.5.1 Requirements Planning | 4 |
| 1.5.2 User Design & Construction | 4 |
| 1.6 Significance of Project & Cutover | 4 |
| 1.7 Project Schedule | 5 |
| 1.8 Outcome | 6 |
| 1.9 Project Outline | 7 |
| 1.10 Conclusion | 7 |
| Chapter 2: Literature Review | 8 |
| 2.1 Introduction | 8 |
| 2.2 Background Study | 8 |
| 2.3 Review of the existing system | 9 |
| 2.3.1 GemPy | 9 |
| 2.3.1.1 Features | 10 |
| 2.3.1.2 Interpolation | 10 |
| 2.3.1.3 Visualization | 11 |
| 2.3.2 Leapfrog Works | 11 |
| 2.3.2.1 Features | 12 |
| 2.3.3 QGIS | 13 |

| | |
|--|----|
| 2.4 Description of borehole data | 14 |
| 2.5 Description of Interpolation Method..... | 14 |
| 2.6 Study on interpolation methods..... | 15 |
| 2.6.1 Kriging | 15 |
| 2.6.2 Inverse Distance Weight (IDW) interpolation..... | 17 |
| 2.6.3 Spline Function..... | 18 |
| 2.7 Fundamental Structural Modeling Rules | 19 |
| 2.7.1 Surface Topology..... | 20 |
| 2.7.2 Connection Between Structural Interfaces..... | 20 |
| 2.7.3 Geometric Constraints | 21 |
| 2.8 Selecting Open-source tools or GIS software | 22 |
| 2.9 GIS Principle & Functions | 22 |
| 2.9.1 Principle | 23 |
| 2.9.2 Functions..... | 23 |
| 2.10 Software Criteria | 24 |
| 2.10.1 Functionality | 25 |
| 2.10.2 Reliability..... | 25 |
| 2.10.3 Usability | 25 |
| 2.10.4 Cost | 25 |
| 2.10.5 Vendor..... | 26 |
| 2.11 Assign sub-criteria score | 26 |
| 2.12 Average scores of main software criteria | 27 |
| 2.13 Summary | 28 |
| Chapter 3: Methodology..... | 29 |
| 3.1 Introduction | 29 |
| 3.2 Summary of Methodology | 29 |
| 3.3 Phase 1: Requirements planning | 32 |
| 3.3.1 Functional Requirements..... | 32 |
| 3.3.1.1 Web Application..... | 32 |
| 3.3.2 Non- Functional Requirements & Performance | 32 |
| 3.3.3 Hardware and software requirements | 33 |
| 3.3.3.1 Computer Device | 33 |

| | |
|---|----|
| 3.3.3.2 Software Requirements | 33 |
| 3.4 Phase 2: User Design..... | 34 |
| 3.4.1 Prototype | 35 |
| 3.4.2 Test..... | 35 |
| 3.4.3 Refine | 35 |
| 3.4.4 Flowchart Diagram | 35 |
| 3.5 Phase 3: Construction | 37 |
| 3.6 Phase 4: Cutover | 38 |
| 3.7 Summary | 38 |
| CHAPTER 4: IMPLEMENTATION AND TESTING | 39 |
| 4.1 Introduction | 39 |
| 4.2 Software Installation..... | 39 |
| 4.2.1 Jupyter Notebook | 39 |
| 4.2.2 Creately | 40 |
| 4.2.3 GitHub | 40 |
| 4.3 Installing GemPy with Python | 41 |
| 4.3.1 Creating Environment | 41 |
| 4.3.2 GemPy Installation | 41 |
| 4.3.3 Theano Installation | 42 |
| 4.3.4 Installing Jupyter Notebook | 43 |
| 4.3 First Cycle of Prototyping..... | 43 |
| 4.4 Refine the prototype..... | 45 |
| 4.5 Programming Code implementation | 48 |
| 4.5.1 Library used..... | 48 |
| 4.5.1.2 Importing Theano library | 48 |
| 4.5.1.3 Importing matplotlib library | 49 |
| 4.5.1.4 Importing pyvista library..... | 49 |
| 4.5.1.5 Surfaces..... | 49 |
| 4.5.1.6 Adding more surface points..... | 50 |
| 4.5.1.7 Orientation Series | 50 |
| 4.5.1.8 Grid..... | 50 |
| 4.5.1.9 Compute the Model..... | 50 |
| 4.5.1.10 Interpolation | 51 |

| | |
|---|----|
| 4.5.1.11 Lithology..... | 51 |
| 4.5.1.12 Adding more surfaces | 51 |
| 4.5.1.13 Adding more surface points..... | 52 |
| 4.5.1.14 Grid and Compute..... | 52 |
| 4.7.1.15 Lithology..... | 52 |
| 4.5.1.16 Topology..... | 53 |
| 4.6 Testing..... | 53 |
| 4.7 User acceptance testing response | 55 |
| 4.8 Conclusion..... | 56 |
| CHAPTER 5: CONCLUSION AND FUTURE WORK | 58 |
| 5.1 Introduction | 58 |
| 5.2 Project Outcome | 58 |
| 5.3 Project Limitation..... | 59 |
| 5.4 Project Future Work..... | 60 |
| References..... | 62 |
| Appendix A- Gantt chart of project schedule | 63 |
| Appendix B- User Acceptance Testing Form..... | 64 |
| Appendix C- Programming Code..... | 65 |

LIST OF TABLES

| | |
|---|----|
| Table 1. 1 Project Timetable | 5 |
| Table 1. 2 Approximate Time for Project Development | 6 |
| Table 2.1 GIS Principle | 23 |
| Table 2. 2 GIS Functions | 23 |
| Table 2. 3 Scores per sub-criteria (Meiracker, 2019) | 26 |
| Table 2. 4 Average Scores of main software criteria (Meiracker, 2019) | 28 |
| Table 3. 1 Summary of the rapid prototyping model | 30 |
| Table 3. 2 The hardware requirement | 33 |
| Table 3. 3 The software requirement | 33 |
| Table 4. 1 Test case for 2D cross- section model | 53 |
| Table 4. 2 Test case for Lithology | 54 |
| Table 4. 3 Test case for 3D Geological Borehole Model with different layers. | 54 |
| Table 5. 1 Project Outcomes | 58 |

LIST OF FIGURES

| | |
|---|----|
| Fig 1. 1 Rapid application development model | 3 |
| Fig 2. 1 (a) Model processing in GemPy & (b) 3D visualization in GemPy | 9 |
| Fig 2. 2 3D output in Leapfrog WorksFig (a) & is 3D output in leapfrog Dashboard (b) | 12 |
| Fig 2. 3 3D monitoring well in QGIS (a) & 3D output in QGIS (b) | 13 |
| Fig 2. 4 Surface layer formed by kriging methods | 16 |
| Fig 2. 5 Using Inverse Distance Weight (IDW) specifying the surface | 17 |
| Fig 2. 6 Using spline interpolation for specifying surface. | 19 |
| Fig 3. 1 Rapid Application Development | 29 |
| Fig 3. 2 Flowchart of 3D geological borehole model..... | 36 |
| Fig 3. 3 Flowchart of Lithology | 36 |
| Fig 3. 4 Flowchart of 2D cross-section | 37 |
| Fig 4. 1 Creating and naming the environment | 41 |
| Fig 4. 2 Installing GemPy on anaconda | 42 |
| Fig 4. 3 Installing Theano compiler | 42 |
| Fig 4. 4 Installing jupyter notebook | 43 |
| Fig 4. 5 Example of different steps before achieving the 3D model; Figure 4.1 (a) simple calling the matplotlib where the given extent and the chosen direction; Figure 4.1 (b) loading the pyvista for 3D plots; Figure 4.1 2D cross-section come out in matplotlib after plotting three surface point and one orientation point; fig(d) The 3D model pop up in pyvista with two surface layer after plotting three surface point and one orientation point..... | 45 |
| Fig 4. 6 2D cross-section model | 46 |
| Fig 4. 7 Lithological scalar field structure..... | 46 |
| Fig 4. 8 3D geological model with topology | 47 |
| Fig 4. 9 Importing libraries and Creating model | 48 |
| Fig 4. 10 Importing Theano library..... | 48 |
| Fig 4. 11 Adding cross-section model which is perpendicular to Y-axis..... | 49 |
| Fig 4. 12 Calling the pyvista..... | 49 |
| Fig 4. 13 GemPy is a surface-based interpolator and after calling the default surface then add the surface points..... | 49 |
| Fig 4. 14 Adding two more surface points | 50 |
| Fig 4. 15 Adding orientation points..... | 50 |
| Fig 4. 16 Calling the grid library..... | 50 |
| Fig 4. 17 Computing the model..... | 50 |
| Fig 4. 18 Interpolate the 2D and 3D surface | 51 |
| Fig 4. 19 Calling the lithology library..... | 51 |
| Fig 4. 20 Adding new surfaces including the basement..... | 51 |
| Fig 4. 21 Adding two more surface points | 52 |
| Fig 4. 22 Calling the grid library and compute the model again | 52 |
| Fig 4. 23 Calling the lithology library..... | 52 |
| Fig 4. 24 Adding topography to execute the final model..... | 53 |
| Fig 4. 25 Questions (Optional Question)..... | 56 |

LIST OF ABBREVIATION

| Abbreviation | Word/Phase |
|--------------|---|
| 3D | Three dimensional |
| 2D | Two dimensional |
| GIS | Geographical Information System |
| RAD | Rapid Application Development |
| SDLC | Systems Development Life Cycle |
| QGIS | Quantum Geographical Information System |
| CD | Compact Disc |
| GPS | Global Positioning System |
| VTK | Visual Toolkit |

Chapter 1: Introduction

1.1 Introduction

The exponential growth of information technology in the digital era of last decades has resulted in a transpose into digital technology from analog and it has been guided by rises in storage and computation power. Geoscience is among one the fields of sciences abruptly changed due to the digital revolution. Geoscience is known as the study of Earth. Geoscience incorporates far beyond rocks and volcanoes; it examines the cycles that structure and shape of Earth's surface and how water and biological systems are interconnected. In geoscience, the use of machine learning techniques has been used immensely. Geoscientist's study and work with minerals, soils, energy assets, fossils, seas and freshwater, the air, climate, ecological science and science, regular risks, and then some. It is very difficult to understand the subsurface structures for modeling of geological process. A three-dimensional model can help the geoscientist to understand the subsurface. Generating a 3D geological model from field information and imaging procedure is a universal task in geographical, hydrological, and asset assessment considers but remains limited. Three-dimensional information structures and imagine complex geospatial information types notwithstanding the restricted information types that current free software handle. Such a structure would have to go about as a paste across existing geo software, smoothing out the information management, representation, and advancement of reproducible examination work processes (Sullivan, 2020). To construct the 3D or 2D geological model needs borehole data. Without boreholes data, it is impossible to directly access the subsurface and almost all the understanding comes up with different indirect processes. The borehole is the term that environmental consultants and engineers use generally to describe all various types of holes bored or drilled in the earth for geological or environmental site inspection. Boreholes supply

information regarding the composition of the earth. It includes holes advanced to collect the water sample, soil sample bedrock, or rock cores to determine their physical properties. 3D structural modeling is not an end, but it means of developing data interpretation through visualization it can also provide numerical simulations like faults, soil conditions, fold structures, and unconformities. Experienced geologists know to do demonstrate 3D models into 2D so there will be less inaccuracy. With the help of the interpolation technique, we can estimate the subsurface at the specific area of the borehole. Interpolation is a sort of estimation that constructs the new data in the range of the discrete set. Interpolation is used to simplify the complex function by illustrate the data and interpolating them by using easier functions. There is a different type of methods they are piecewise constant interpolation (nearest-neighbor interpolation), polynomial interpolation, linear interpolation, and spline interpolation. And another versatile interpolation method is the Gaussian process. Gaussian process is also a robust non-linear interpolation technique that is used in the geostatistics community for fitting a curve through noisy data and process regression is called as well as kriging. For better work, there are few interpolation techniques with Geographical Information System (GIS) like Spline, Gaussian (kriging), PointInterp.

1.2 Problem Statement

There is no available web application to compute the borehole interpolation for determining the soil structure of the different locations. For this reason, the interpolation method will be different for every location because the geographical structure may differ with every location. For analyzing the data of borehole users need to be clearer about the criteria and the interpolation methods else it will take more time. And without proper information about the interpolation methods and borehole data, it is hard to demonstrate the 3D geological model.

1.3 Scope

With the help of the interpolation method, we can be able to calculate the subsurface between the boreholes, and later with the help of visualization tools like PyVista or The Visualization Toolkit, we can be able to project the 3D geological model of the boreholes on the system.

1.4 Objective

For this project there are two main objectives:

1. To project the borehole data into a 3D geological representation model.
2. To represent research on open-source geological tools or software.

1.5 Methodology

Methodologies like RAD (Rapid Application Development), SCRUM, Waterfall methods are used for creating a system that can be modularized within few months. Rapid application development emphasized working software over strict modules and requirements documentation. Rapid application development is a form of Agile software development methodology that prioritizes rapid prototype releases and iterations. The benefits of RAD are fast project turnaround, minimizing the planning stage, and more focus on prototype development.

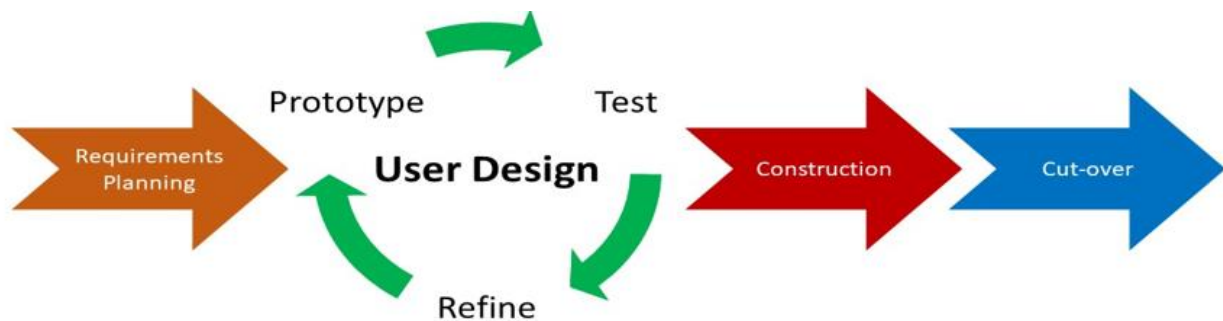


Fig 1. 1 Rapid application development model

Developing the system developers can follow the steps of the RAD cycle. RAD cycle has four phases that have been stated below:

1.5.1 Requirements Planning

In SDLC (Systems Development Life Cycle) it helps to combine the elements of systems analysis phases and system planning. It ends when the team includes users, IT managers, or IT stuffs agree and discuss the project scope, business needs, system requirements, and constraints. After that, the team agreed on the main issues and acquire management authorization to proceed.

1.5.2 User Design & Construction

In the phase of user design, the system analysts interact with users and develop the prototype which represents all the inputs, outputs, and all system processes. The user design phase allows users to modify, understand, and approved the working model that meets their needs. In the construction phase, more focus is on application and program development tasks like SDLC. In this stage users still can participate and can still suggest changes. The tasks are application development, programming, unit integration, coding, and system testing.

1.6 Significance of Project & Cutover

The borehole log is a comprehensive geological representation that is used for the description of the geological structure, soil characteristics, borehole structure, groundwater sampling, and testing. The borehole log is used for analyzing the geological condition and forming the geological map. The geographical structure is different according to the area that is why the soil structure is also different. And to get the soil information in a specific area must be calculated. To calculate the borehole log there are various interpolation techniques. Interpolation methods need to be more relevant to averted technical error. Borehole interpolation is needed to construct a clear vision about the subsurface or the layers of the soil between the location of the borehole which is more efficient and saves the time of boring. Resembling the final task in data conversion, testing, including the SDLC implementation phase to the new system and user

training. Furthermore, the entire new system is built and delivered. In this phase, the user is trained and guide for further use of the system.

1.7 Project Schedule

The project schedule-timeline given by the Faculty of Computer Science & Information Technology is shown in the table.

Table 1. 1 Project Timetable

| Date | Agenda | Action |
|-------------------------|---|---------------------------------|
| 23/10/2020 | Submission of the approved brief proposal by supervisor | Student |
| 30/10/2020 | Comment from reviewers | Student |
| 07/11/2020 | Submission of Final proposal report after the changes | Student |
| 21/11/2020 | Submission of chapter 1 | Student |
| 12/12/2020 | Submission of chapter 2 | Student |
| 02/01/2021 | Submission of chapter 3 | Student |
| 13/01/2021 | Submission of FYP1 paper and final report of the assessment | Student & Supervisor |
| 20/01/2021 – 22/01/2021 | Final Year Project 1 symposium | Student, Examiners & Supervisor |
| 25/01/2021 | Final date for the examiner to give the comment & feedback | Supervisor/ Examiner |
| 01/02/2021 | Final date for submission of assessment mark through FYP Management System | Supervisor/ Examiner |
| 05/02/2021 | Submission of FYP1 final report and paper | Student, Examiners & Supervisor |
| 07/04/2021 | Submission of chapter 4 | Student |
| 01/05/2021 | Submission of chapter 5 | Student |
| 12/05/2021 | Submission of chapter 6 | Student |
| 18/06/2021 | Submission of FYP2 | Student |
| 28/06/2021 | Final date for the examiner to give the comment & feedback | Supervisor/ Examiner |
| 19/06/2021 – 17/-7/2021 | Amendment and Modification Period for FYP | Student, Examiners & Supervisor |
| 18/07/2021 | Final Date for Submission of assessment mark through FYP Management System (Online) | Supervisor/ Examiner |
| 25/07/2021 | Submission of Final Report (Hard Cover & | |

| | | |
|--|---|--|
| | Softcopy), source code, report in word format, user guide, installation kits. | |
|--|---|--|

For further project development the estimated time is given below:

Table 1. 2 Approximate Time for Project Development

| Task | Duration |
|---|----------|
| 1. Requirement Planning | 13 |
| 1.1 Gather Elements | 8 |
| 1.2 Analysis system requirements | 7 |
| 2. User Design | 28 |
| 3. Construction | 90 |
| 3.1 Construction the application | 29 |
| 3.2 Develop the system | 30 |
| 3.3 Develop the connection between application and system | 31 |
| 4. Documentation | 7 |
| 5. Cutover | 42 |
| 5.1 Installing the system | 30 |
| 5.2 Software Testing | 12 |

1.8 Outcome

Borehole interpolation is a method of finding the subsurface profile between various locations to display it in a graph. For geographical structure, a suitable interpolation method needs to be chosen which can show the composition of the soil. Open-source software like GemPy will be used in this project to construct the three-dimensional model. GemPy has a Python-based open-source geomodelling library. It can construct a complex 3D model including various features. The expected outcome will be making a web-based application that can interpolate the boreholes and display the 3D geological model. In the end, with the help of borehole data, GemPy will construct a 2D cross-section model including a 3D geological borehole model.

1.9 Project Outline

This proposed project is divided into five major chapters. In chapter one provides information about the project where an introduction, overview, problem statement, scope, objectives, the significance of the project, and outcome. In chapter two is the literature review where the research of the chosen software that is now existed in the field will be featured, the correlation between existed systems and the proposed systems will be assessed. Moreover, the discussion about the proposed system's proclivity over the current ones will be done in chapter two. In chapter three the description of the methodology to develop the chosen project. This chapter will explain the detailing process of the system design and development needed to come up with GemPy. In chapter four, focus on testing and implementation. In this chapter, a brief description of the prototyping cycle and further software testing. The overall result will be evaluated and explained. In chapter five will cover the project outcome, project limitations, and a short suggestion on future work. The summary of the project will be narrated in this chapter.

1.10 Conclusion

The geological model is a representation of a certain portion of Earth soil, it is an intricate process to develop a model with the help of borehole data. To understand the model client needs to have basic knowledge regarding geologic modeling. After completing the project, clients will be able to execute the earth crust model and also have a clear view of the soil condition of the borehole data.

Chapter 2: Literature Review

2.1 Introduction

In the literature review, there will be a brief description regarding background study, existing software, interpolation methods, the borehole data, and open-source software which will demonstrate the 3D geological subsurface model. And there will be a comparison between three open-source tools or (GIS) Geographic information system software packages where each software will be justifying according to the valid criteria and GIS software criteria.

2.2 Background Study

We are living in an era where almost everything is technology-based. Technology gradually developing with time and with the help of technology human beings making their life easier. In the mining industry, the use of geological modeling software gradually increasing due to facing the downgraded of resources. Few software companies claim that their software is more towards reducing the geological risk. But geologists do not have a clue how to develop a model the mineralization designs are because the solitary instruction they have about geological modeling comes from the software developer, who shows geologists how to squeeze which buttons and what techniques ought to be utilized for specific demonstrating issues. Geological modeling is mainly used to control the natural resources, determine natural hazards, with main applications to soil tasting, gas fields, groundwater, mine ores deposits. Using geological models engineers able to identify the behavior of the rocks under the soil. In geological models, a 2D model is represented by a polygon that can be bounded by unconformities or by faults. And in the 3D model unit is enclosed in gridded surfaces or triangulated. These three-dimensional model grids are comparable to two-dimensional model grids which are used to communicate properties of single surfaces. Furthermore, the main purpose of the proposed system is to provide an easy

controlling, user-friendly, and less complicated process system to execute the geological model for the user.

2.3 Review of the existing system

There is numerous open-source GIS (Geographic Information System) software available with different functionalities and operations. GIS software envelops a wide scope of utilizations which include the utilization of a mix of advanced guides and georeferenced information. GIS software can be arranged into various categories. One single software is not able to satisfy all the needs. For further studies, the open-source tools and software are GemPy, leapfrog works, and QGIS.

2.3.1 GemPy

GemPy is a Python-based, local area-driven, open-source geomodelling library. It is equipped for developing complex 3D geographical models including different highlights, for example, fault structures, unconformities, and fault networks, considering underlying powerful implicit approaches. From the beginning, GemPy was intended to be handily implanted in probabilistic systems for leading vulnerability analysis concerning subsurface structures.

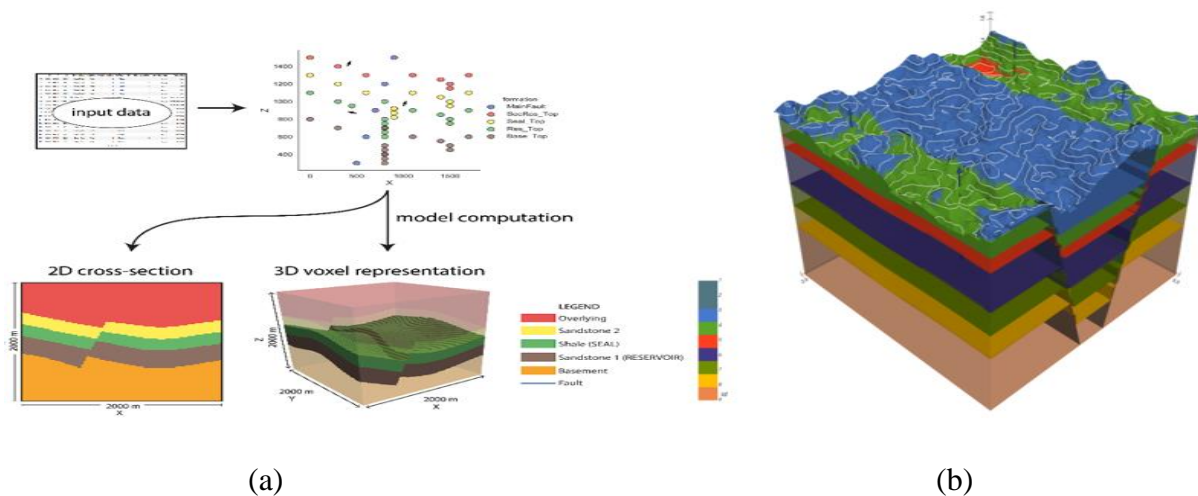


Fig 2. 1 (a) Model processing in GemPy & (b) 3D visualization in GemPy

2.3.1.1 Features

GemPy is equipped for modeling complex 3D geographical situations, including:

- I. Multiple conformal layers.
- II. Some sequences of layers with unconformities or conformal continuation.
- III. Faults.
- IV. Full fault networks.
- V. Folds.

Joining these components in GemPy considers the age of reasonable 3D geological models, on a standard with most commercial geomodelling software.

2.3.1.2 Interpolation

The age of complex primary settings depends on the powerful interpolation algorithm underlying GemPy, a universal cokriging technique. This method is utilized to interpolate a 3D scalar field, with the end goal that geologically huge interfaces are isosurfaces in this field. The algorithm considers an immediate integration of two of the most applicable geographical input data types.

- I. **Surface contact points:** Three-dimensional coordinates of points denoting the limits between various highlights (for example layer interfaces, unconformities, fault plane) (Varga, Schaaf, & FlorianWellmann, 2019).
- II. **Orientation measurements:** Orientation of the poles perpendicular to the dipping of surfaces at any point in the 3D space (Varga, Schaaf, & FlorianWellmann, 2019).

GemPy additionally takes into consideration the meaning of topological components, for example, joining numerous stratigraphic successions and complex fault organizations to be considered in the modeling process.

2.3.1.3 Visualization

Models produced with GemPy can be envisioned severally:

- I. Direct visualization of 2D model segments using matplotlib, including hill shading and different choices for natural representation of results (Varga, Schaaf, & FlorianWellmann, 2019).
- II. Intuitive 3D visualization and model input manipulation using the Visualization Toolkit (VTK);
- III. GemPy effectively builds up a connection to the phenomenal by vista project for far superior visualization and model interaction in 3D.

Notwithstanding visualization, the produced models can be traded in an assortment of ways:

- I. VTK documents for additional perception and preparation in another programming, for example, ParaView (Varga, Schaaf, & FlorianWellmann, 2019).
- II. Triangulated surface meshes.
- III. Export of images.

GemPy right now chipping away at tighter integration with a few cross-section libraries and Gmesh. Furthermore, GemPy has set up connections to a few other open-source libraries, including pygiml for geophysical inversion and modeling.

2.3.2 Leapfrog Works

Leapfrog is geological modeling gave by Seequent. Leapfrog offers a few modules, including Leapfrog Geo and Leapfrog Works. Leapfrog Works is explicitly created for the demonstrating of the subsurface and environmental project and gives bits of knowledge into subsurface information inside a 3D environment. The geological modeling software permits quick operation

of geographical organizations straightforwardly acquired from drill hole data. Leapfrog Works offers a few paid licensing choices for various purposes.

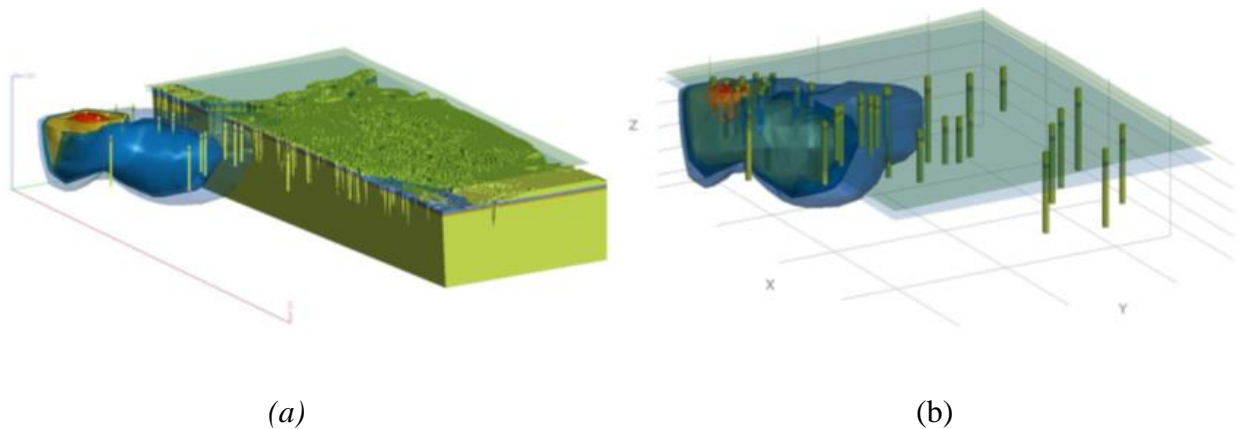


Fig 2. 2 3D output in Leapfrog WorksFig (a) & is 3D output in leapfrog Dashboard (b)

2.3.2.1 Features

Leapfrog works help the data with:

- I. Dynamic 3D modeling. Create static models for reference and comparison. Automatically can be able to add more new borehole data while the parameters hold.
- II. Ground-type classifications. Rapidly can build 3D geological and numerous data from various data sources including GIS, borehole, map, 2D grid, cross-section data, mesh.
- III. Collaboration and sharing. Easily can be able to share the images, cross-sections, interactive 3D scenes.
- IV. Interoperability.
- V. Engineering design rapidly visualizes and understands the borehole, bridge, tunnel, and building foundation designs in a geological context.
- VI. Environmental insight.
- VII. Generate cross-sections.

VIII. Hydrogeology extension.

2.3.3 QGIS

QGIS is an open-source GIS where geographic data can be seen, altered, and analyzed. QGIS is a Free and Open-Source Software (FOSS) and depends consequently on the help of volunteers and benefactors. QGIS offers a developing number of functionalities, in the application itself and just as by the establishment of modules. For a year, QGIS incorporates a 3D plugin offering instruments for 3D visualization.

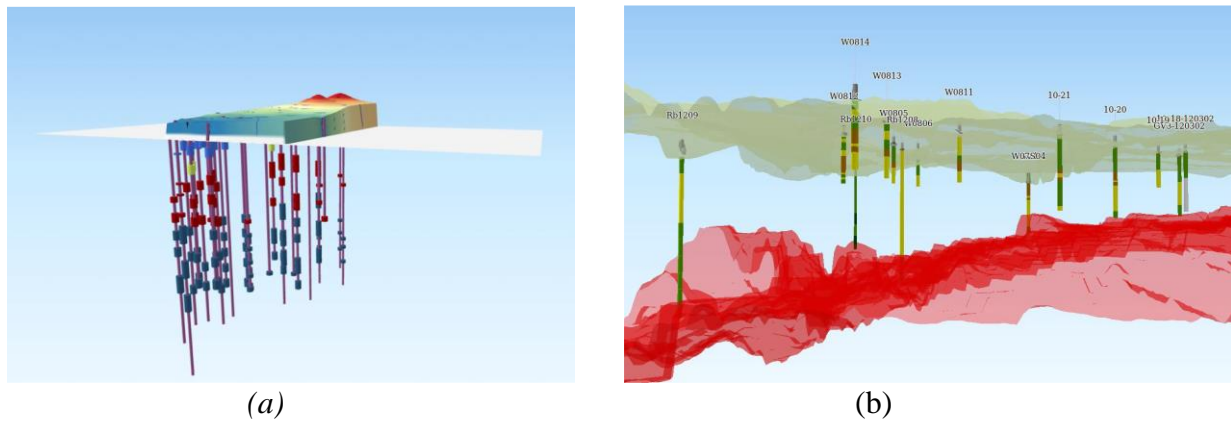


Fig 2. 3 3D monitoring well in QGIS (a) & 3D output in QGIS (b)

2.3.3.1 Features

QGIS offers numerous basic GIS functionalities given by center highlights and modules. A short outline of six general classifications of highlights and modules is introduced underneath, trailed by initial experiences into the incorporated Python support.

- I. Can view and overlay vector and raster information in various organizations and projections without transformation to an inside or regular configuration.
- II. Can create maps and intuitively investigate spatial information with a benevolent GUI.
- III. Can edit, create, export, and manage vector and raster layers in a few configurations.

- IV. Can perform spatial information examination on spatial data sets and other OGR-upheld designs. QGIS at present offers vector analysis, testing, geoprocessing, geometry, and data set management tools. Likewise, can be able to use the incorporated GRASS tools, which incorporate the total GRASS functionality of more than 400 modules.
- V. QGIS can be utilized as WMTS, WMS, WFS-T, and WFS customer, and as a WFS, WMS server. Moreover, users can publish the data or information on the Internet utilizing a webserver with GeoServer or MapServer installed.
- VI. QGIS can be adjusted the extraordinary requirements with the extensible plugin architecture and libraries that can be utilized to create plugins.

2.4 Description of borehole data

The borehole is the term that environmental consultants and engineers use generally to describe all various types of holes bored or drilled in the earth for geological or environmental site inspection. Boreholes supply information regarding the composition of the earth. Borehole data originate from the borehole log which is used to demonstrate the soil subsurface or the soil sediment. And this data can be used in the 3D geological model to demonstrate the soil layer in between the boreholes.

2.5 Description of Interpolation Method

Interpolation is the way toward utilizing points with known values or surface points to assess values at other obscure points. It is used to prognosticate unknown points any geographic point data like noise levels, elevations, and so on. The interpolation method is used in GIS (Geographic Information Systems) which is used for identifying the location in a vector or tensor field, multivariate scalar. Typical examples are climatic phenomena, elevations, population densities, fluxes of matter, soil properties (MITAS & H MITASOVA, 1999). Because of limited

resources and high costs, data collection is usually done in a limited number of specific locations. There are few suppositions of interpolation which is work for estimating the attribute data of location according to the data boundary. These suppositions then use to develop the interpolation method. These methods have been designed to support transformations between different discrete and continuous representations of spatial and spatiotemporal fields, typically to transform irregular point or line data to raster representation, or to resample between different raster resolutions (MITAS & H MITASOVA, 1999). Since there exist a boundless number of capacities that satisfy this requirement, extra conditions must be imposed, characterizing the different interpolation techniques. Geostatic interpolation and deterministic interpolation are the two group of interpolation method which is categorized under spatial interpolation. In the dataset, data are noisy and discrete which makes the task more difficult to decide which interpolation method need to use. The objective of interpolation is to make a surface that is planned to best speak to empirical reality subsequently the technique chose should be surveyed for exactness. So, choosing the methods of interpolation that need to evaluate and test the accuracy to understand the method.

2.6 Study on interpolation methods

For further studies, the interpolation method that is chosen is Kriging, Inverse Distance Weighting (IDW), and Spline Function.

2.6.1 Kriging

Kriging is maybe the most unmistakable of interpolation methods. Kriging is an amazing sort of spatial interpolation that utilizes complex numerical equations to gauge esteems at unknown points dependent on the values at known points. A kriged approximate is a weighted linear combination of the realized selected value around the point be assessed. Kriging technique that

produces an expected surface from a dispersed arrangement of points with z-values. Kriging accepts that the distance between selected values mirrors a spatial relationship that can be utilized to clarify the variation on the surface layer.

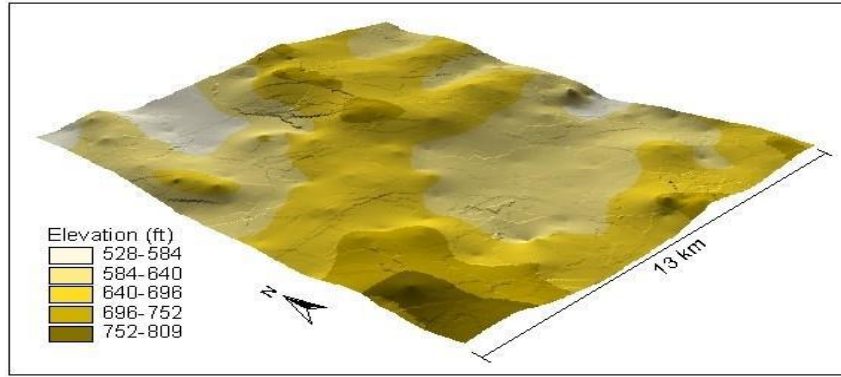


Fig 2. 4 Surface layer formed by kriging methods

Kriging method is familiar to Inverse Distance Weight in that it weights the encompassing estimated values to infer a prediction for an unmeasured location. The formula for kriging methods as a weighted sum of the data:

$$\hat{Z}(s_0) = \sum_{i=1}^N \lambda_i Z(s_i)$$

Where $Z(s_i)$ = to estimate the value at i^{th} location, S_0 = to predict the location, λ_i = weighted value to predict the value at i^{th} location, N = number of predicted values. In the Kriging method, weights are on the distance between prediction location and measured points and spatial arrangement of measured points. Kriging methods define as two separated forms one is universal kriging and another one is ordinary kriging. The main fundamental of the kriging method is to identify the weight at λ_i . With different predictions and calculations, both universal kriging and ordinary kriging can measure the interpolation values with weight sum. The difference is the Universal kriging refer that the mean value is not constant but as a polynomial function and on the other side Ordinary kriging refer that all data value is constant.

2.6.2 Inverse Distance Weight (IDW) interpolation

This method is the most readily and simplest. Inverse distance weighted (IDW) interpolation is a careful technique that upholds that the assessed estimation of a point is impacted more by close-by known points than those are farther away. Sample points are weighted during interpolation to such an extent that the impact of one point relative toward another decays with distance from the obscure point. IDW function should be utilized when the arrangement of points is sufficient enough to catch the extent of local surface variation required for analysis.

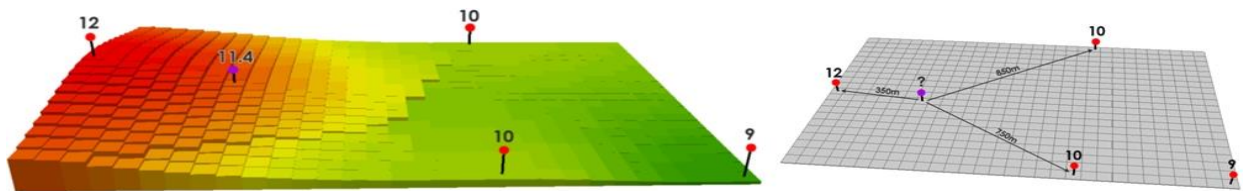


Fig 2. 5 Using Inverse Distance Weight (IDW) specifying the surface

Inverse Distance Weight procedure computes a value for every grid node by looking at encompassing data points that exist in a user-defined search radius. A few or the entirety of the data points can be utilized in the interpolation method. The node esteem is determined by averaging the weighted sum of all the points. Data points that lie dynamically further from the node impact the processed incentive far not exactly those lying nearer to the node. The distance-based methods precisely utilize these distances between the estimation points and known points to weigh their impact in the figuring of the assessed esteem. Incidentally, they require a linear spatial relationship among phenomena. Given below is the Inverse distance weight formula (Tan & Xu, 2014).

| | |
|---|--|
| $\hat{V} = \frac{\sum_{i=1}^n \frac{1}{d_j} v_t}{\sum_{i=1}^n \frac{1}{d_t}}$ <p>The basic formula of Inverse Distance Weighting IDW</p> | <p>\hat{V} = is valued to be estimated</p> <p>v_t = is known to value</p> <p>d_j, \dots, d_t = is the distance point of n from the data point to the estimated point</p> |
| $\hat{V}_1 = \frac{\sum_{i=1}^n \frac{1}{d_i^p} v_t}{\sum_{i=1}^n \frac{1}{d_n^p}}$ <p>IDW formula with added distance weighting exponent</p> | <p>\hat{V} = is valued to be estimated</p> <p>v_t = is known to value</p> <p>d_i^p, \dots, d_n^p = is the distance point of n from the data point to the estimated point</p> |

From the above formula, we come to know that in the second exponent the more consistently all neighbors are fused into the calculation, and we can get a smoother surface the higher exponent the more complemented and unstable is the surface.

2.6.3 Spline Function

The spline method approximates values utilizing a mathematical function that limits the total surface bend, bringing about a smooth surface that goes precisely through the sample points. While there is more entry point indicated, the more prominent the impact of inaccessible points also, the smoother the surface. Points of interest of splining capacities are they can create adequately exact surfaces from a couple of examined points and they hold little highlights.

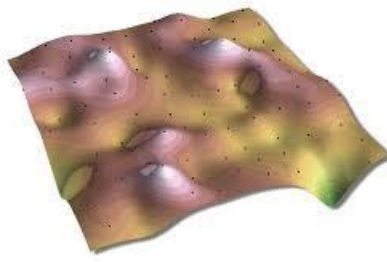


Fig 2. 6 Using spline interpolation for specifying surface.

An inconvenience is that they may have unexpected least and most extreme qualities in comparison to the data set and the capacities are delicate to exceptions due to the consideration of the original data values at the selected points. There are more like quadratic spline, linear spline, cubic spline, and higher degree. The following formula is used for surface interpolation (Tan & Xu, 2014):

$$S(x, y) = T(x, y) + \sum_{j=1}^N \lambda_j R(r_j)$$

Where N = Number of points, λ_j = is coefficients which are found by the system of linear equation, $j = 1, 2, 3, 4, \dots, N$, r_j = is the distant point (x, y) to j^{th} point. Depending on the selected option, $R(r)$ and $T(x, y)$ can be defined differently.

2.7 Fundamental Structural Modeling Rules

3D primary model is made of geological interfaces, for example, horizons and faults respecting accessible perception data. These surfaces should fit the data inside a satisfactory range, contingent upon data exactness, and most of the part-work more precisely. Every 3D surface represents a geological irregularity because of the progressions of depositional conditions, structural occasions, or erosion like intrusion or faulting. A predictable structural model contains surfaces fitting perception data yet in addition right connections between the geological interfaces (Caumon, Collon-Drouaillet, Veslud, & Sausse, 2009). For this reason, some fundamental demonstrating rules must be seen in the modeling output. Much of the time, these

imperatives are authorized by recognizing the macro-topology, or lattice, which is utilized to show the border of an object, and the micro-topology, or cross-section, which manages the mesh of the object (Caumon, Collon-Drouaillet, Veslud, & Sausse, 2009). In this segment, we center around rules identified with the macro-topology. On a basic level, these guidelines are similar when drawing a 2D cross-section. In any case, the three-dimension makes it hard to identify the area where these principles have encroached. Along these lines, we currently expressly stress some topological prerequisites which consistently hold in structural modeling. A portion of these standards may naturally be authorized by software usage, yet all are examined here for oversimplification.

2.7.1 Surface Topology

A 3D surface model that is legitimate from a mathematical point of view does not address a substantial natural object. Undoubtedly, a geological surface is a limit between two volumes of rocks with various attributes (seismic impedance, lithology, and so on). Hence, the surface direction state that a land surface is consistently orientable, and it has two all-around characterized sides. Moreover, a surface will not self-converge, it would propose that the volumes isolated by this surface overlap one another.

2.7.2 Connection Between Structural Interfaces

A large portion of the 3D structural modeling attempt adds up to sorting out how horizons and faults are associated with one another. In three-dimensional modeling, it is conceivable and advantageous to utilize a casual variation of this surface non-intersection rule, expressing that any two surfaces ought not to cross one another, aside from it if one has been cut by the other (Caumon, Collon-Drouaillet, Veslud, & Sausse, 2009). This implies, for example, that a fault surface necessity is not cut along horizon tear lines, which makes the model refreshing a lot

simpler when new data open. Additional consistency conditions depend on the idea of logical borders on a surface, which portrays the macro-geography (Caumon, Collon-Drouaillet, Veslud, & Sausse, 2009). A surface border is characterized by a bunch of associated border edges. For building the geo-modeling, the border of a surface can be part into a few pieces called logical borders, contingent upon their beginning or role (Caumon, Collon-Drouaillet, Veslud, & Sausse, 2009). For example, a fault tear line on a horizon comprises two logical borders for the footwall and hanging wall. From this definition, the free border decides states that only fault surfaces may have a logical border not associated with other structural model interfaces. Surely, stratigraphic surfaces fundamentally end onto unconformities, faults, or on the other hand model boundaries.

2.7.3 Geometric Constraints

Notwithstanding data consistency, the authenticity of the structural model geometry, however harder to portray unbiasedly, ought to consistently be assessed. This might be done visually in total or cut 3D scenes, and by extricating cross-segments. A significant advance in quality control is to utilize faults juxtaposition graphs to check that fault displacement does not shift unexpectedly. Strike varieties might be reviewed by showing horizon cutoff lines on the fault footwall and hanging wall (Caumon, Collon-Drouaillet, Veslud, & Sausse, 2009). Moreover, vertical variations ought to be taken a gander at by checking layer thickness on both sides of a fault and its similarity with fault kinematics. Furthermore, the nearby surface direction might be checked visually to identify modeling artifacts, for example, saddle geometrics between cross-segments. Quantitative approaches may likewise be utilized, for example, surface curvature analysis. Structural models ought to be viable with a wide range of observation data that result from the physical process in the earth, for example as seismograms or reservoir production data (Caumon, Collon-Drouaillet, Veslud, & Sausse, 2009).

2.8 Selecting Open-source tools or GIS software

Generally, it is a quite complex and tiring process to select the right tools or software because to elect a preferred software package that meets the required conditions and requirements. Because of the expanded capability of the software, the costs of cutting-edge GIS software bundles have expanded. Then again, because of the developing requests and creative activities more available. GIS software is created, which has even prompted free software, for example, QGIS or tools like GEMPY. All this new software is worked to meet diverse client prerequisites and to run on various equipment stages. For further studies, we choose the open-source tools and software which are GemPy, leapfrog works, and QGIS. But before working on software we need to meet the software criteria as well as need to know about the principle and functions of GIS so that we specify our needs.

2.9 GIS Principle & Functions

The significance of researching and recognizing the dimension, reason, and objective of GIS software should be more precise and from this point of view which will help to choose an appropriate software which is brought up in the past area. To do this it is significant that the activity of GIS software bundles and explicitly which is for this examination of the 3D modeling operation and limit identified with subsurface information is perceived, and known. This part thusly examines GIS principles identifying with estimation scales and levels, vector and raster information, soil properties, point mists, and secret elements. The subsections prove the significance of explicit soil-related 3D functionalities to sensibly speak to the subsurface. Below the table, we will discuss the principle and functions of GIS.

2.9.1 Principle

Table 2.1 GIS Principle

| Principle | Objectives |
|--------------------------------|---|
| Data Capture | Data sources are mainly collected from scanning and manual digitization of aerial photographs, existing digital datasets, and paper maps. |
| Database Management and Update | Data integrity, data security, data storage, data maintenance abilities, and data retrieval. |
| Geographic Analysis | The collected information is interpreted and analyzed quantitatively and qualitatively. |
| Preparing Result | The most exciting part is there are different ways and a variety of information that can be presented in GIS technology. |

2.9.2 Functions

Table 2. 2 GIS Functions

| Functions | Objectives |
|--------------|--|
| Data capture | There are many different methods of gathering the input of data into a GIS like scanning, aerial photography, GPS (Global Positioning System), digitizing. These are the few ways a GIS user can be able to obtain the data. |

| | |
|--------------------|---|
| Data manipulation | Users can able to edit the digital geographical data. This allows for many attributes to be edited, deleted, added to the specification of the project. |
| Data storage | Some data is stored in a drawer as a map, while digital data can be stored on a hard drive or in CD as a hardcopy. |
| Query and analysis | GIS was used extensively in decision-making for the commission districts. Like, using the population data to the area for each district to establish an equal representation of the population. |
| Visualization | The ability to display the input data, maps, and information. |

2.10 Software Criteria

The criteria on which the software bundles are assessed and significant in the dynamic cycle. The necessities and wants that the software or tools should meet are set by the criteria. Studies are done portraying techniques and instruments for software choice. Five main criteria are defined for selecting suitable software or tools and they are functionality, reliability, usability, cost, and vendor (Eldrandaly & Naguib, 2011). Below we discuss the five criteria for more information.

2.10.1 Functionality

Criteria identified with functionality assess the accessibility of all required and wanted capacities of the software. The assessment of the ability of the GIS software bundle is an urgent and advance in choosing reasonable GIS software. Since, if the software is truly inadequate in the accessible functionalities the product is unfruitful, notwithstanding what different models may score. Foreordained activities or operations should be executed at a specific level with accessible functionalities and tools.

2.10.2 Reliability

Software reliability assesses the nature of the software's exhibition. The software framework should have the option to play out the necessary capacities under-expressed conditions, including at a specific speed or for a specific information size. Development, adaptation to internal failure, and recoverability relate to unwavering quality.

2.10.3 Usability

The viability, effectiveness, and fulfillment with which the item clients can work and accomplish their objectives are delegated as a convenience. The ease-of-use models mirror the nature of utilization qualities.

2.10.4 Cost

The principle rules of cost contain all uses related to the GIS software package. This incorporates the software item, training, license, software membership, maintenance, and support services costs. These expenses can be isolated into two gatherings: operating expenditures and capital uses. Capital uses are non-repeating costs including the software product, license, and training. Operating expenditures are repeating costs associated with the GIS project, for example, service costs and maintenance.

2.10.5 Vendor

The vendor contains rules to gauge the administration and backing which is provided with the software package. This incorporates sub-measures that relate to the tools and support that the seller offers. Instructional exercises, manuals, consultancy, and help services with the splitting of the vendor criteria.

2.11 Assign sub-criteria score

The score in the table for each sub-criterion is provided per alternative. Here, the score which is 1 is the least desirable, and the score 5 is the most desirable.

Table 2. 3 Scores per sub-criteria (Meiracker, 2019)

| Criteria | Sub-criteria | GemPy | Leapfrog | QGIS | Max |
|-------------------------|---------------------|-------|----------|------|-----|
| Work | | | | | |
| 1. Functionality | Included | 3 | 4 | 1 | 5 |
| | functionality | 3 | 4 | 3 | 5 |
| | Data formats | 4 | 5 | 3 | 5 |
| | Data visualization | 4 | 5 | 4 | 5 |
| | Adaptability | 4 | 5 | 4 | 5 |
| | Data sharing | | | | |
| 2. Reliability | Time behavior | 4 | 3 | 2 | 5 |
| | Robustness | 3 | 4 | 1 | 5 |
| | Backup and recovery | 4 | 4 | 3 | 5 |
| | | | | | |
| 3. Usability | User interface | 4 | 4 | 3 | 5 |
| | User types | 4 | 4 | 2 | 5 |

| | | | | | |
|------------------|------------------|----|----|----|-----|
| | Ease of use | 3 | 3 | 2 | 5 |
| | Domain variety | 4 | 4 | 5 | 5 |
| | Error reporting | 2 | 2 | 2 | 5 |
| 4. Cost | Software license | 5 | 3 | 5 | 5 |
| | Installation | 5 | 4 | 5 | 5 |
| | Training | 4 | 4 | 5 | 5 |
| | Maintenance | 4 | 4 | 5 | 5 |
| | Support service | 4 | 5 | 5 | 5 |
| 5. Vendor | User manual | 5 | 5 | 5 | 5 |
| | Tutorial | 5 | 5 | 5 | 5 |
| | Demo version | 5 | 4 | 5 | 5 |
| | Training | 4 | 4 | 4 | 5 |
| | Forum | 3 | 5 | 2 | 5 |
| | Σ | 90 | 94 | 81 | 115 |

2.12 Average scores of main software criteria

The average scores of the main software criteria are shown for each alternative. The max value of functionality, usability, cost, vendors are 25 and reliability has 15.

Table 2. 4 Average Scores of main software criteria (Meiracker, 2019)

| Criteria | GemPy | Leapfrog work | QGIS | MAX |
|---------------|-------|---------------|------|-----|
| Functionality | 18 | 23 | 15 | 25 |
| Reliability | 11 | 11 | 6 | 15 |
| Usability | 17 | 17 | 14 | 25 |
| Cost | 22 | 20 | 25 | 25 |
| Vendor | 22 | 23 | 21 | 25 |
| Σ | 90 | 94 | 81 | 115 |

2.13 Summary

This research is aimed at 3D functionalities among software or tools to support 3D geological models. Three software or tools package is evaluated based on predefined criteria to support 3D data. According to the above data, we come to know that Leapfrog is the leading among the three of them, but it is commercial software so it needs to buy, and it will cost around \$12,500 which is very expensive. QGIS is crash-sensitive especially during extended operations such as interpolation. Due to the crash the software being shut down immediately. And GemPy is user-friendly python-based can-do complex 3D geological models and most importantly GemPy is open source and easily accessible and does not any cost. But due to the limitations, GemPy could not be able to generate the 3D borehole model it can only execute the surface model.

Chapter 3: Methodology

3.1 Introduction

Based on the methodology which is represented in chapter 1 is Rapid Application Development (RAD) and which consists of four phases and in this chapter will discuss more on the phases of the methodology which is design and analysis. Chapter 3 will summarize each of the phases including specifying the requirements. The system design will be illustrated using the flowchart. Then the mockup design is proposed 3D geological scenarios as the RAD cycle undergo will be shown. And at the end of this chapter, there will be a summarization of the content.

3.2 Summary of Methodology

Rapid application development is a form of Agile software development methodology that prioritizes rapid prototype releases and iterations. The benefits of RAD are fast project turnaround, minimizing the planning stage, and more focus on prototype development. In this section, there will be a summary of each RAD phase on its techniques and software used.

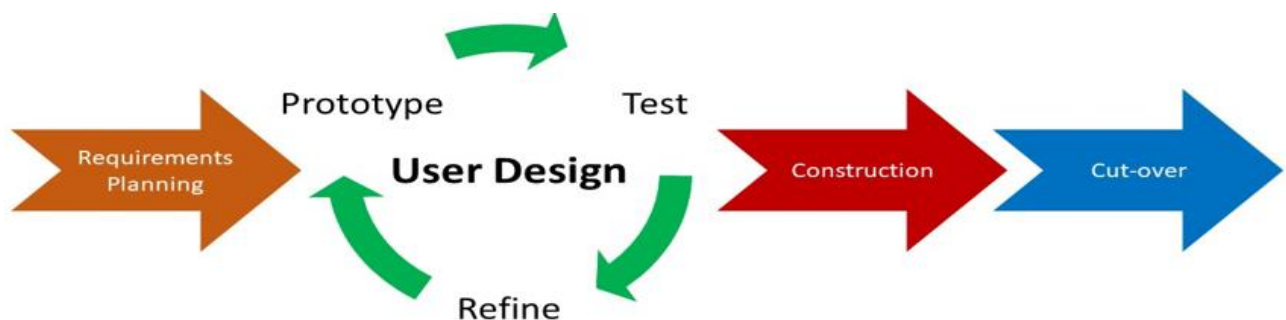


Fig 3. 1 Rapid Application Development

Table 3. 1 Summary of the rapid prototyping model

| Phases | Task | Techniques | Software |
|------------------------------|---|--|----------------------------|
| Requirements Planning | <ul style="list-style-type: none"> Obtain the project requirements Analyze the hardware and software needed Simplify the non-functional and functional Design mock-up interface for 3D geological surface | <ul style="list-style-type: none"> Research on open-source geological software. Draw flowchart for system design. Design mock-up interfaces for the web application with the variant process. | Online based tools Creatly |
| User Design | <ul style="list-style-type: none"> Implementing prototype. Presenting the prototype. | <ul style="list-style-type: none"> Coding Meeting with the client to represent the prototype. Brainstorming the feedback from the client. Reconstruct the | Jupyter Notebook |

| | | | |
|---------------------|---|---|------------------|
| | | <p>prototype by changing the methods and codes to have better performance and meeting the client's needs.</p> | |
| Construction | <ul style="list-style-type: none"> • Make several testings on the prototype whenever needed before finalizing the prototype. • Redefine and solve the problem if the prototype faces any bug. | <ul style="list-style-type: none"> • Provide the prototype to the tester for testing. • Final rechecked and met all the client's needs before finalizing the prototype. | Jupyter Notebook |
| Cutover | <ul style="list-style-type: none"> • Prepare a full final document. • Handover the final project to | <ul style="list-style-type: none"> • Prepare a user manual to train the user how to use the 3D | Microsoft word |

| | | | |
|--|------------|-------------------------------|--|
| | the client | geological borehole model. | |
|--|------------|-------------------------------|--|

3.3 Phase 1: Requirements planning

Requirement planning is the fundamental method to distinguish client assumptions on how the item works dependent on their requirements. Hence, an assortment of information is needed to see more about what sort of data will be utilized. In this part, the functional and non-functional necessities will be talked about so as the software prerequisites.

3.3.1 Functional Requirements

3D geological borehole model is proposed by using a web application where the functional requirements will be described in web application requirements.

3.3.1.1 Web Application

The function that is available on the web application is stated below:

- The web application can display a 3D geological model.
- The web application can show several sequences of a layer.
- The web application can display the direct visualization of 2D sections.
- The web application will analyze the borehole data to generate the cross-section.

3.3.2 Non- Functional Requirements & Performance

Non-function requirements will discuss the performance of the 3D geological borehole model. Generate the borehole data and show the sub-layers which will determine the calculation speed so that the 2D cross-section will represent and as well as display the subsurface.

3.3.3 Hardware and software requirements

The hardware requirements which are used in the proposed 3D geological model is the computer device which will be stated below.

3.3.3.1 Computer Device

The computer device which is used to operate the web application is stated below in table 1.

Table 3. 2 The hardware requirement

| Hardware Component | Requirement (Recommended) |
|--------------------------------|-------------------------------|
| Processor (CPU) | Inter ® Core™ i7-10750H |
| Memory (RAM) | 8192 MB |
| Graphic Card | GeForce GTX 1650 Max-Q Design |
| Operating System (OS) Platform | 64 bit |

3.3.3.2 Software Requirements

The software requirements that are proposed for the 3D geological borehole model are stated below in table 3.3.

Table 3. 3 The software requirement

| Software | Description |
|-------------|--|
| Web Browser | A software application for accessing the |

| | |
|------------------|---|
| | information on the World wide web. |
| Jupyter Notebook | The Jupyter Notebook is an open-source web application that allows you to create and share documents. It is used to write the source code for this project. |
| Create | It is an online-based tool to draw a flow chat for this project. |
| GitHub | GitHub is a code hosting platform for version control and collaboration |

3.4 Phase 2: User Design

When the project work is investigated, it's an ideal opportunity to bounce directly into development and working out the client plan through different prototype repetition. During this stage, users work inseparably with the developers to guarantee their necessities are being met at each progression in the planning cycle. It's practically similar to adaptable software development where the user can test every step of their prototype, to guarantee it lives up to their desires. Each bug is worked out in an iterative cycle. The software developer plans a model the client tests it, and afterward, they meet up to impart on what worked and what didn't. This technique offers software developers the chance to change the model as they go until they arrive at a palatable plan. Both the product designers and the customers gain from the experience to ensure there is no potential for something to escape everyone's notice.

3.4.1 Prototype

In this phase where system and application design will be work on. In the prototyping phase, where programming language, different types of tools to design will be involved to develop the 3D geological borehole model.

3.4.2 Test

The test is the next process after the prototype in the user design cycle. Once the prototype is ready to use them the system must need to be tested by the developer before being handed over to the client. If everything is functioning well then the client will also run the system for finalizing the system.

3.4.3 Refine

Refine is the last phase of the user design cycle where the client or developer will refine the project if there are any flaws find out after testing. In this process, the system will again back to the first cycle where the developer refines the prototype and will be tested again until the refined prototype meets the requirements. This is the reason refine cycle is one of the crucial and key phases to build a project.

3.4.4 Flowchart Diagram

The system design of the 3D geological borehole model system is presented using a flow chart or a different type of logical structure. Here, the flowchart is used to demonstrate the sequence of the steps of decision acquire to perform the process. Below is the flowchart process:

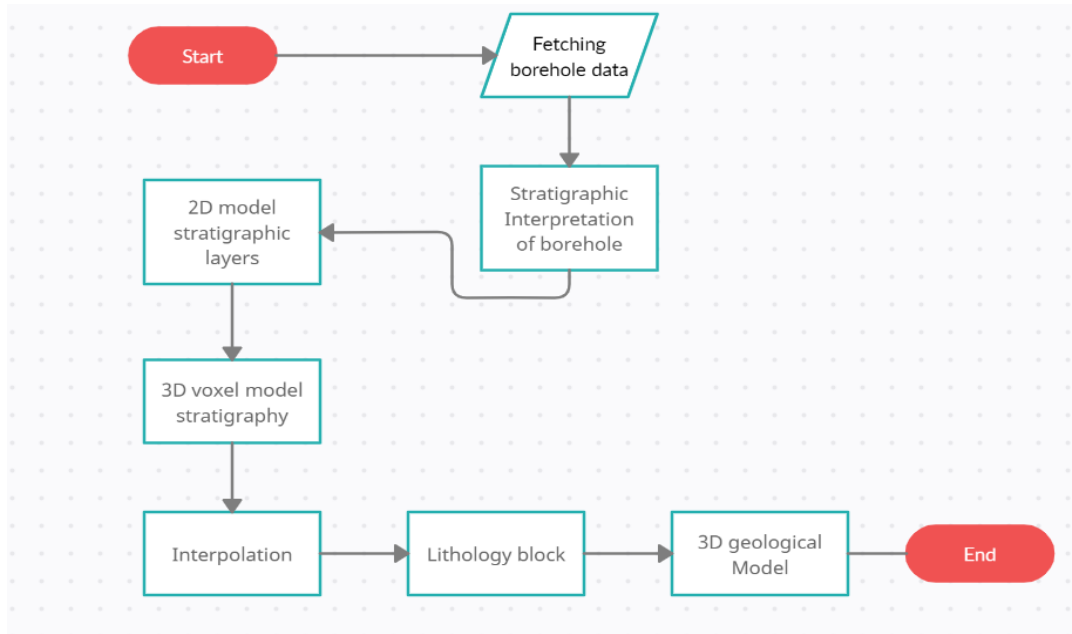


Fig 3. 2 Flowchart of 3D geological borehole model

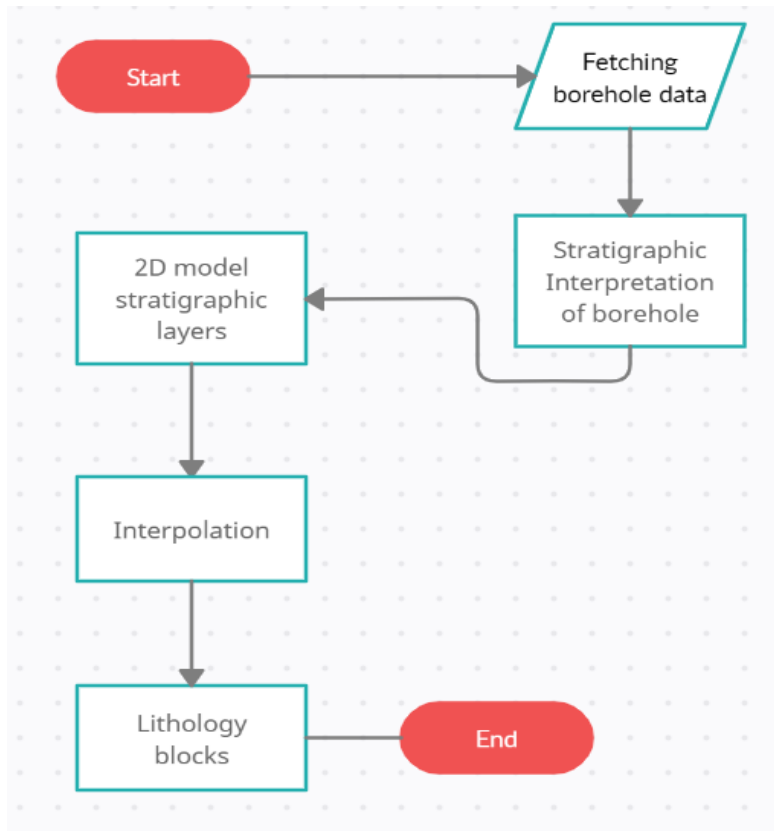


Fig 3. 3 Flowchart of Lithology

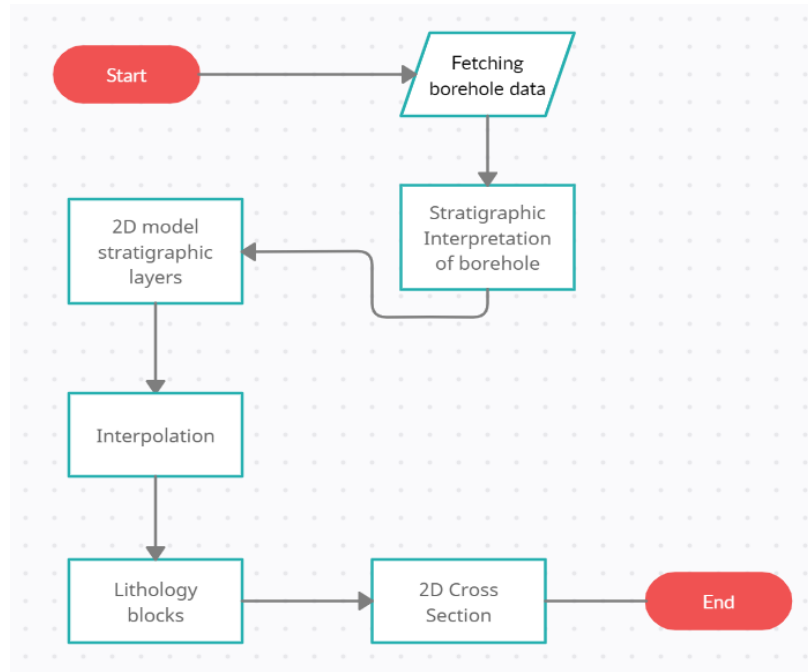


Fig 3. 4 Flowchart of 2D cross-section

3.5 Phase 3: Construction

In the construction phase, the prototype and beta system from the designing stage and converts into the functioning model. Since most of the issues and changes were tended to during the careful iterative designing stage, developers can develop the last working model more rapidly than they could by following a traditional project management approach. The software improvement team of coders, programmers, developers, and testers and cooperate during this stage to ensure everything is working easily and that the outcome fulfills the client assumptions and requirements.

In this phase, python language will be used to develop the geological modeling system. In the beginning, anaconda (distribution of python programming) needs to install first so that the applied python libraries can be written, and all the code will be generated on jupyter notebook. After retrieving the borehole data and calling all the necessary libraries then all the relevant models like 2D cross-section or 3D geological model can be seen in the monitor.

This third stage is significant in light of the fact that the customer actually will give contribution all through the interaction. They can propose changes, adjustments, or even bring new thoughts that can tackle issues as they emerge.

3.6 Phase 4: Cutover

In this execution phase where the completed system goes to dispatch. It incorporates data transformation, a changeover to the new framework and testing, and even user training. All last changes are made while the coders and users keep on searching for bugs in the software. In this phase, the developer will test all the functionality of every module. It includes the finalization of the features, interface, functions, and everything else related to the software project. Interfaces between the different autonomous modules require legitimate testing. This is done in the cutover stage. This is done in the cutover phase. For this project, after completing all possible functionality testing, the developer then handed over the project to the client.

3.7 Summary

In conclusion, the methodology will bring a clear view of how to complete and carry the process with the help of rapid application development. This project is low-cost and also needs to finish it on a very short period, for this reason, compared to other methodologies RAD is quicker and flexible. This method is easy and fast processing which will help to finish the project early.

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Introduction

This chapter describes the process of implementing the 3D borehole model. In chapter 3 it was stated that how the system was designed and describe the methodology which is Rapid Application Development (RAD). In the User Design phase, there is a cycle that illustrates the Prototype, Test, and Refine. This phase continues until the prototype is complete and matches all the objectives which are demonstrated in the project. Before proceeding with the implementation part, need to install and set up the software tools and environments for a further touch-up. These configurations are explained more precisely in this chapter. The software testing is performed in both functional testing and non-functional testing. This software testing is conducted to ensure the fulfillment of the objectives and performing the prototype without any error.

4.2 Software Installation

Before proceeding to develop the prototype, we need to set up and install various software. For our main software GemPy, it has specific hardware requirements, as we are doing this project using Python, so we need to go through some trail as every dependency has a different type of version and for this reason, there be compatibility problems between the dependency's version with python. Below is the software information that is used to develop the system and needed to install before proceeding with the project.

4.2.1 Jupyter Notebook

Jupyter notebook is a free, open-source, intelligent web apparatus known as a computational notebook, which researchers can use to consolidate programming code, computational yield, informative content, and sight and sound assets in a solitary document. Computational notebooks have been around for quite a long time, however, Jupyter specifically has detonated in notoriety

over the recent years. This fast take-up has been helped by an excited user-developer and upgraded engineering that permits the journal to communicate in many programming languages. The Jupyter notebook has two segments. User input programming code or text in rectangular cells in a front-end website page. The program at that point passes that code to a back end 'Kernel', which runs the code and returns the outcomes.

4.2.2 Creately

A graphing tool worked considering coordinated effort, straightforwardness, and simplicity. Creately simplified diagramming so anyone can convert their thoughts into visuals without any difficult steps. From advanced infographics to flowcharts, it supports different types of diagrams. Create focused on making the diagram experience profitable as could be expected. Anyone can easily sign up for free and begin drawing immediately utilizing their expertly planned templates. Anyone can appreciate the benefits like constant coordinated effort and the anyplace availability of online-based software.

4.2.3 GitHub

GitHub is a code facilitating stage for adaptation control and joint effort. It lets you and others cooperate on projects from any place. Projects on GitHub can be gotten to and overseen utilizing the standard Git order line interface; all standard Git orders work with it. GitHub likewise permits users to peruse public storehouses on the site. Different work area customers and Git modules are likewise accessible. Anybody can peruse and download public stores however just enlisted users can contribute substantially to vaults. With an enlisted user's account, clients can have conversations, oversee storehouses, submit commitments to others' archives, and survey changes to code. GitHub started offering limitless private archives at no expense in January 2019 (restricted to three donors for every undertaking). Beforehand, just open storehouses were free.

4.3 Installing GemPy with Python

After installing the anaconda, now we are going to install the GemPy to construct the geological model.

4.3.1 Creating Environment

```
(base) PS C:\Users\legui> conda create --clone transform2020 --name t20-gempy
WARNING: A directory already exists at the target location 'C:\Users\legui\miniconda3\envs\t20-gempy'
but it is not a conda environment.
Continue creating environment (y/[n])? y
```

Fig 4. 1 Creating and naming the environment

4.3.2 GemPy Installation

```
(gempy-tutorial2) C:\Users\legui> pip install gempy
Collecting gempy
  Using cached gempy-2.2.9-py3-none-any.whl (446 kB)
Collecting scikit-image>=0.17
  Using cached scikit_image-0.18.1-cp38-cp38-win_amd64.whl (12.2 MB)
Requirement already satisfied: Theano>=1.0.4 in w:\miniconda\envs\gempy-tutorial2\lib\site-packages (from gempy) (1.0.4)
Collecting pytest
  Using cached pytest-6.2.2-py3-none-any.whl (280 kB)
Collecting seaborn>=0.9
  Using cached seaborn-0.11.1-py3-none-any.whl (285 kB)
Requirement already satisfied: numpy in w:\miniconda\envs\gempy-tutorial2\lib\site-packages (from gempy) (1.19.2)
Collecting networkx
  Using cached networkx-2.5-py3-none-any.whl (1.6 MB)
Collecting pyvistaqt
  Using cached pyvistaqt-0.3.0-py3-none-any.whl (160 kB)
Collecting pyvista>=0.25
  Using cached pyvista-0.29.0-py3-none-any.whl (1.2 MB)
Collecting iPython
  Using cached ipython-7.21.0-py3-none-any.whl (784 kB)
Collecting pandas>=1.0.5
  Using cached pandas-1.2.3-cp38-cp38-win_amd64.whl (9.3 MB)
Collecting matplotlib
  Using cached matplotlib-3.3.4-cp38-cp38-win_amd64.whl (8.5 MB)
Collecting python-dateutil>=2.7.3
  Using cached python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
Collecting pytz>=2017.3
  Using cached pytz-2021.1-py2.py3-none-any.whl (510 kB)
Requirement already satisfied: six>=1.5 in w:\miniconda\envs\gempy-tutorial2\lib\site-packages (from python-dateutil>=2.7.3->pandas>=1.0.5->gempy) (1.15.0)
Collecting meshio<5.0,>=4.0.3
  Using cached meshio-4.3.11-py3-none-any.whl (153 kB)
Collecting scooby>=0.5.1
```



```

Using cached imageio-2.9.0-py3-none-any.whl (3.3 MB)
Requirement already satisfied: scipy>=1.0.1 in w:\miniconda\envs\gempy-tutorial2\lib\site-packages (from scikit-image>=0.17->gem
py) (1.6.1)
Collecting tifffile>=2019.7.26
Using cached tifffile-2021.3.17-py3-none-any.whl (163 kB)
Collecting PyWavelets>=1.1.1
Using cached PyWavelets-1.1.1-cp38-cp38-win_amd64.whl (4.3 MB)
Collecting cycler>=0.10
Using cached cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3
Using cached pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
Collecting kiwisolver>=1.0.1
Using cached kiwisolver-1.3.1-cp38-cp38-win_amd64.whl (51 kB)
Collecting decorator>=4.3.0
Using cached decorator-4.4.2-py2.py3-none-any.whl (9.2 kB)
Collecting traitlets>=4.2
Using cached traitlets-5.0.5-py3-none-any.whl (100 kB)
Collecting backcall
Using cached backcall-0.2.0-py2.py3-none-any.whl (11 kB)
Collecting colorama
Using cached colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Collecting jedi>=0.16
Using cached jedi-0.18.0-py2.py3-none-any.whl (1.4 MB)
Collecting prompt-toolkit!=3.0.0,!<3.0.1,<3.1.0,>=2.0.0
Using cached prompt_toolkit-3.0.18-py3-none-any.whl (367 kB)
Requirement already satisfied: setuptools>=18.5 in w:\miniconda\envs\gempy-tutorial2\lib\site-packages (from iPython->gempy) (52.0.0.post20210125)
Collecting pickleshare
Using cached pickleshare-0.7.5-py2.py3-none-any.whl (6.9 kB)
Collecting pygments
Using cached Pygments-2.8.1-py3-none-any.whl (983 kB)
Collecting parso<0.9.0,>=0.8.0
Using cached parso-0.8.1-py2.py3-none-any.whl (93 kB)
Collecting wcwidth
Using cached wcwidth-0.2.5-py2.py3-none-any.whl (30 kB)
Collecting ipython-genutils
Using cached ipython_genutils-0.2.0-py2.py3-none-any.whl (26 kB)
Collecting tomli
Using cached tomli-0.10.2-py2.py3-none-any.whl (16 kB)
Collecting iniconfig
Using cached iniconfig-1.1.1-py2.py3-none-any.whl (5.0 kB)
Using cached PyQt5_sip-12.8.1-cp38-cp38-win_amd64.whl (63 kB)
Installing collected packages: pillow, wcwidth, vtk, transforms3d, scooby, pytz, python-dateutil, PyQt5-sip, PyQt5-Qt5, pyparsin
g, parso, meshio, kiwisolver, ipython-genutils, imageio, decorator, cycler, appdirs, traitlets, tomli, tifffile, QtPy, PyWavelets
, pyvista, PyQt5, pygments, py, prompt-toolkit, pluggy, pickleshare, pandas, packaging, networkx, matplotlib, jedi, iniconfig, c
olorama, backcall, attrs, atomicwrites, seaborn, scikit-image, pyvistaqt, pytest, iPython, gempy
Successfully installed PyQt5-5.15.4 PyQt5-Qt5-5.15.2 PyQt5-sip-12.8.1 PyWavelets-1.1.1 QtPy-1.9.0 appdirs-1.4.4 atomicwrites-1.4
.0 attrs-20.3.0 backcall-0.2.0 colorama-0.4.4 cycler-0.10.0 decorator-4.4.2 gempy-2.2.9 iPython-7.21.0 imageio-2.9.0 iniconfig-1
.1 ipython-genutils-0.2.0 jedi-0.18.0 kiwisolver-1.3.1 matplotlib-3.3.4 meshio-4.3.11 networkx-2.5 packaging-20.9 pandas-1.2.3
parso-0.8.1 pickleshare-0.7.5 pillow-8.1.2 pluggy-0.13.1 prompt-toolkit-3.0.18 py-1.10.0 pygments-2.8.1 pyparsing-2.4.7 pytest-
5.2.2 python-dateutil-2.8.1 pytz-2021.1 pyvista-0.29.0 pyvistaqt-0.3.0 scikit-image-0.18.1 scooby-0.5.6 seaborn-0.11.1 tifffile-
2021.3.17 tomli-0.10.2 traitlets-5.0.5 transforms3d-0.3.1 vtk-9.0.1 wcwidth-0.2.5

```

Fig 4. 2 Installing GemPy on anaconda

4.3.3 Theano Installation

```

(t20-gempy) PS C:\Users\legui> conda install theano
Collecting package metadata (current_repodata.json): done
Solving environment:
The following NEW packages will be INSTALLED:
libgomparray pkgs/main/win-64::libgomparray-0.7.6-hfa6e2cd_0
libpython pkgs/main/win-64::libpython-2.1.py37_0
m2w64-binutils pkgs/msys2/win-64::m2w64-binutils-2.25.1-5
m2w64-bzip2 pkgs/msys2/win-64::m2w64-bzip2-1.0.6-6
m2w64-crt-git pkgs/msys2/win-64::m2w64-crt-git-5.0.0.4636.2595036-2
m2w64-gcc pkgs/msys2/win-64::m2w64-gcc-5.3.0-6
m2w64-gcc-ada pkgs/msys2/win-64::m2w64-gcc-ada-5.3.0-6
m2w64-gcc-fortran pkgs/msys2/win-64::m2w64-gcc-fortran-5.3.0-6
m2w64-gcc-objc pkgs/msys2/win-64::m2w64-gcc-objc-5.3.0-6
m2w64-headers-git pkgs/msys2/win-64::m2w64-headers-git-5.0.0.4636.c0ad18a-2
m2w64-isl pkgs/msys2/win-64::m2w64-isl-0.16.1-2
m2w64-libiconv pkgs/msys2/win-64::m2w64-libiconv-1.14-6
m2w64-libmangle-g pkgs/msys2/win-64::m2w64-libmangle-git-5.0.0.4509.2e5a9a2-2
m2w64-make pkgs/msys2/win-64::m2w64-make-4.1.20151.a0a0a0d-2
m2w64-mpc pkgs/msys2/win-64::m2w64-mpc-1.0.3-3
m2w64-mpfr pkgs/msys2/win-64::m2w64-mpfr-3.1.4-4
m2w64-pkg-config pkgs/msys2/win-64::m2w64-pkg-config-0.29.1-2
m2w64-toolchain pkgs/msys2/win-64::m2w64-toolchain-5.1.0-7
m2w64-tools-git pkgs/msys2/win-64::m2w64-tools-git-5.0.0.4552.90b8472-2
m2w64-windows-def pkgs/msys2/win-64::m2w64-windows-default-manifest-6.4-3
m2w64-winpthreads pkgs/msys2/win-64::m2w64-winpthreads-git-5.0.0.4636.697f752-2
m2w64-zlib pkgs/msys2/win-64::m2w64-zlib-1.2.0-10
make pkgs/main/narch:make-1.1.2-py_0
pygpu pkgs/main/win-64::pygpu-0.7.6-py37h452e1ab_0
theano pkgs/main/win-64::theano-1.0.4-py37_0
Proceed [y/n]?

```

Fig 4. 3 Installing Theano compiler

4.3.4 Installing Jupyter Notebook

```
(t20-gempy) PS C:\Users\legui> jupyter notebook
[I 10:14:22.326 NotebookApp] JupyterLab extension loaded from C:\Users\legui\miniconda3\envs\t20-gempy\lib\site-packages\jupyterlab
[I 10:14:22.327 NotebookApp] JupyterLab application directory is C:\Users\legui\miniconda3\envs\t20-gempy\share\jupyter\lab
[I 10:14:22.350 NotebookApp] Serving notebooks from local directory: C:\Users\legui
[I 10:14:22.350 NotebookApp] The Jupyter Notebook is running at:
[I 10:14:22.351 NotebookApp] http://localhost:8888/?token=5fdad16c8db71a00aa8480e15c435ecd44a7fc1eca307e36
[I 10:14:22.351 NotebookApp] or http://127.0.0.1:8888/?token=5fdad16c8db71a00aa8480e15c435ecd44a7fc1eca307e36
[I 10:14:22.351 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 10:14:22.423 NotebookApp]

To access the notebook, open this file in a browser:
    file:///C:/Users/legui/AppData/Roaming/jupyter/runtime/nbserver-12428-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=5fdad16c8db71a00aa8480e15c435ecd44a7fc1eca307e36
    or http://127.0.0.1:8888/?token=5fdad16c8db71a00aa8480e15c435ecd44a7fc1eca307e36
```

Fig 4. 4 Installing jupyter notebook

4.3 First Cycle of Prototyping

A wide scope of techniques exists to produce geological models. Nonetheless, the incredible techniques are behind a paywall in costly business bundles. Here presenting a full open-source geo-modeling strategy, because of a potential-field interpolation method. The interpolation calculation is tantamount to executions in a package and fit for developing a complex full 3D geological model. This interpolation algorithm is carried out in a programming language like Python and utilizing a profoundly proficient basic library for effective code era Theano that empowers an immediate execution on GPUs. GemPy is new but it can create a complex three-dimension geological model including multiple conformal layers, several levels of a layer including conformal and unconformities, folds, and so on combining these elements which help to generate a 3D geological model. As GemPy is python-based software, after installing the Anaconda version 4.9.2 once installed we need to install the GemPy by calling its dependencies. As GemPy required numerous libraries it automatically installs other dependencies but to install we need to install theano manually and in the conda version theano comes up with a non-critical

bug but once Theano compatible compiler is installed it solve the problem. Once all the dependencies are installed then we install the Jupyter notebook, conda automatically install the latest version of Jupyter notebook and connect with the jupyter server. Once open the jupyter notebook we need to import the data to construct the model in GemPy and which is stored in a python object. Then we generate the data from Github where we store the raw data, once we retrieve the data then we can start constructing the model by declaring the sequential order of geological formations. After declaring the data, we need to do the interpolation approach, the complex structural model is mainly based on interpolation methods in GemPy. To generate integration of this algorithm GemPy needs two input data types one is surface contact points, and another is orientation measurements. Surface contact points are three-dimension coordinates of points to demonstrate the boundaries between the different types of features like layer interfaces, unconformities, faults, and so on. Orientation measurements are the direction of the poles perpendicular to the plunging of surfaces anytime in the three-dimension space. To generate the model with GemPy the visualization needs to go through several ways. Generating the visualization of 2D model GemPy using matplotlib. Matplotlib is a numerical mathematics extension that is a plotting python library, it provides API (Application Programming Interface) to embedding plots in the application to generate the figure or plots some direction line in a plotted area. Matplotlib helps to visualize all the input data and measurements in 2D projection according to the chosen direction. With the help of Pyvista GemPy can also visualize the same data in 3D. GemPy also uses Visualization Toolkit (VTK) to demonstrate the 3D model visualization but due to confliction of python dependencies that are using pyvista to execute the 3D model. The given below Figure 4.1 show the process and the stages from the cross-section model to the 3D model.

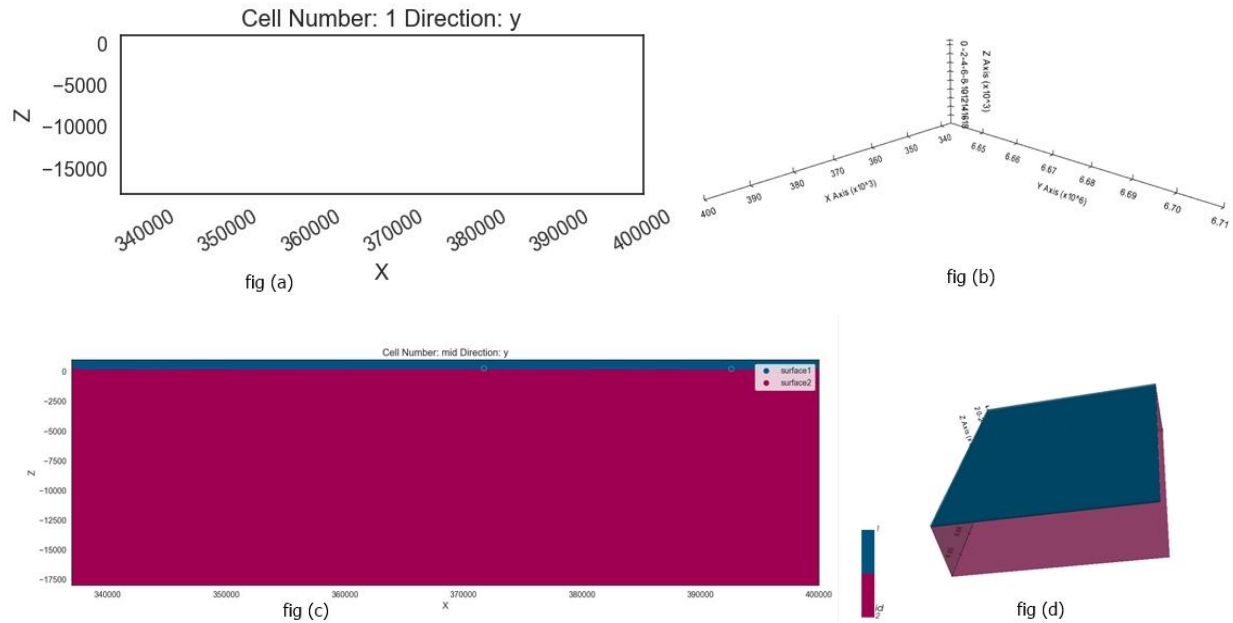


Fig 4. 5 Example of different steps before achieving the 3D model; Figure 4.1 (a) simple calling the matplotlib where the given extent and the chosen direction; Figure 4.1 (b) loading the pyvista for 3D plots; Figure 4.1 2D cross-section come out in matplotlib after plotting three surface point and one orientation point; fig(d) The 3D model pop up in pyvista with two surface layer after plotting three surface point and one orientation point.

4.4 Refine the prototype

After considering the feedback, start changing the function because in the beginning the function was to import the data from Github but now we add the surface points and orientations measurement according to the need. In GemPy, the surfaces are always empty by default. GemPy is mainly a surface-based interpolator. This implies that all the information we add must be referred to a surface. The surfaces consistently mark the bottom or lower part of a unit and the bottom layer consider as the basement which is not to be interpolated so in GemPy it always interpolates the surface. Once we import GemPy and creating a set of input data that is used to construct the model in GemPy which is stored in python then we call matplotlib and pyvista to visualize the input data. Adding the surface points data and orientations, the 2D plots give the X and Z coordinates which will be visualized on matplotlib. To construct the model in GemPy requires a minimum of 2 surface points and one orientation per series. Once we have enough

data then proceed to interpolate the 3D surfaces. This time we add extra two surface points to generate the second layer in the 3D geological model. Figure 4.2 illustrates the 2D cross-section model.

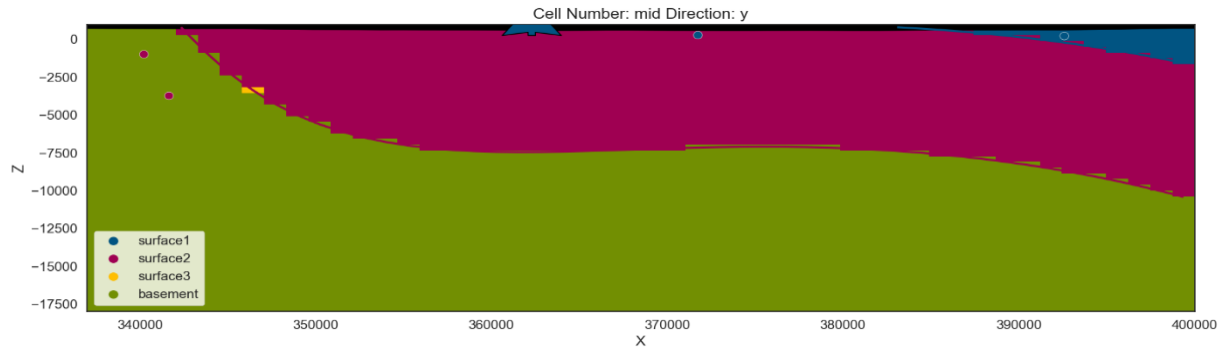


Fig 4. 6 2D cross-section model

Figure 4.6 is the direct visualization of the 2D cross-section model using matplotlib. In the 2D cross-section, there is a prospect to display all the possible locations of the interfaces. In the 2D cross-section layer, it shows the difference in soil structure, for example, the first layer is silt (fine sand or clay) and at the bottom, the layer is empty which represent that the portion of that soil has not been discovered yet or do not have the information. Figure 4.3 demonstrates the lithological scalar field structure.

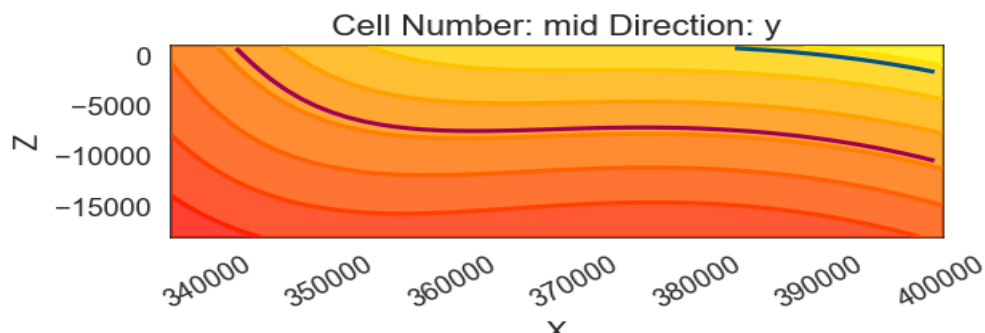


Fig 4. 7 Lithological scalar field structure

Figure 4.7 is the lithological scalar field structure that is generated through GemPy. In the beginning, the resolution was set at 50 cells in every direction. In GemPy cells moved in mid direction y but the plots illustrate straight parallel to the X-axis and Y-axis. Figure 4.4: 3D geological model with topology.

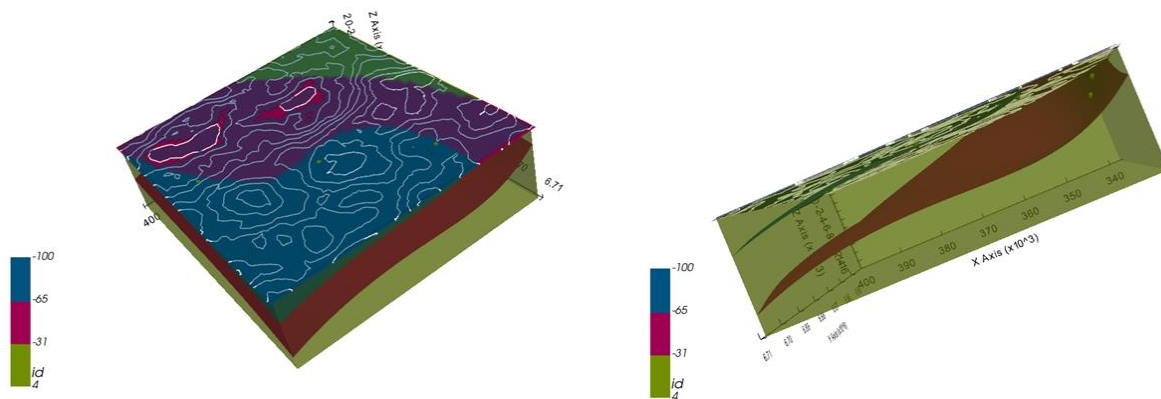


Fig 4. 8 3D geological model with topology

The above figure is the direct visualization of a 3D geological model including topology. In this model, there are slits, two surfaces layer, and a basement. In GemPy, the 3D model structural can be viewed in the open-source Visualization Toolkit or Pyvista but over here to overview the 3D model pyvista is being used. It gives the user an intelligent 3D perspective on the geographical model. Users can be able to choose to plot the borehole data, the land surfaces, or both. This usefulness can improve the comprehension of the model as well as assist the client with acquiring the ideal result by working straightforwardly in a 3D geological model while getting immediate visual criticism on the displaying results. After this refinement, there is no more refining the model but need to add data manually and have to interpolate the model to view the 2D cross-section and 3D geological model.

4.5 Programming Code implementation

All the code was written in python language as GemPy is totally python based. The data used to construct the model in GemPy all is stored in python objects.

4.5.1 Library used

```
In [1]: import gempy as gp
import sys, os
os.environ["THEANO_FLAGS"] = "mode=FAST_RUN,device=cpu"

import gempy as gp

# Importing aux libraries
from ipywidgets import interact
import numpy as np
import matplotlib.pyplot as plt

# Embedding matplotlib figures in the notebooks
%matplotlib qt5

No module named 'osgeo'

In [2]: geo_model = gp.create_model('Model2')
        # of the regular grid
geo_model = gp.init_data(geo_model,
                        extent=[337000,400000,664000,671000,-18000,1000], # for the regular grid
                        resolution=[50, 50, 50]) # for the regular grid

Active grids: ['regular']
```

Fig 4. 9 Importing libraries and Creating model

4.5.1.2 Importing Theano library

```
In [3]: # Calling the Library Theano
gp.set_interpolator(
    geo_model,
    output=['geology'],
    theano_optimizer='fast_compile')

Setting kriging parameters to their default values.
Compiling theano function...

Level of Optimization: fast_compile
Device: cpu
Precision: float64
Number of faults: 0
Compilation Done!
Kriging values:
range          96072.888996
$C_o$          219761904.761905
drift equations [3]

Out[3]: <gempy.core.interpolator.InterpolatorModel at 0x1d9cdeeae80>
```

Fig 4. 10 Importing Theano library

4.5.1.3 Importing matplotlib library

```
In [4]: #calling the matplotlib
p2d = gp.plot_2d(geo_model, section_names=None,
                direction=None, cell_number=None)

In [5]: # 2D cross section for the model which is perpendicular to y axis
ax = p2d.add_section(cell_number=1, direction='y')
```

Fig 4. 11 Adding cross-section model which is perpendicular to Y-axis

4.5.1.4 Importing pyvista library

```
In [6]: # Calling the Pyvista
p3d = gp.plot_3d(geo_model, plotter_type='background', notebook=False)
```

Fig 4. 12 Calling the pyvista

4.5.1.5 Surfaces

```
In [7]: geo_model.surfaces

Out[7]:
  surface  series  order_surfaces  color  id
0  surface1  Default series         1  #015482  1
1  surface2  Default series         2  #9f0052  2

In [8]: #default surfaces
geo_model.set_default_surfaces()

Out[8]:
  surface  series  order_surfaces  color  id
0  surface1  Default series         1  #015482  1
1  surface2  Default series         2  #9f0052  2

In [9]: # Add a surface point. Here idx is None so next available index will be used
geo_model.add_surface_points(X=352798.25, Y= 6698505.5, Z= 254.579992, surface='surface1', idx=0)

#2D plot
p2d.plot_data(ax, cell_number=11)

#3D plot
p3d.plot_surface_points()

Out[9]: (vtkmodules.vtkRenderingOpenGL2.vtkOpenGLActor)000001D9D14CFF40
```

Fig 4. 13 GemPy is a surface-based interpolator and after calling the default surface then add the surface points.

4.5.1.6 Adding more surface points

```
In [10]: #Adding more surface points
geo_model.add_surface_points(X=371748.625, Y=6683887, Z=282.857477999999, surface='surface1', idx=1)
geo_model.add_surface_points(X=392594.063, Y=6668185, Z=228.405954, surface='surface1', idx=2)

# 2D & 3D Plotting
p2d.plot_data(ax, cell_number=11)
p3d.plot_surface_points()

Out[10]: (vtkmodules.vtkRenderingOpenGL2.vtkOpenGLActor)000001D9D15942E0

In [26]: geo_model.surface_points

Out[26]:
```

| | X | Y | Z | smooth | surface |
|---|------------|-----------|------------|----------|----------|
| 0 | 352798.250 | 6698505.5 | 254.579992 | 0.000001 | surface1 |
| 1 | 371748.625 | 6683887.0 | 282.857478 | 0.000001 | surface1 |
| 2 | 392594.063 | 6668185.0 | 228.405954 | 0.000001 | surface1 |

Fig 4. 14 Adding two more surface points

4.5.1.7 Orientation Series

```
In [11]: #Adding orientation point
geo_model.add_orientations(X=362273.4375, Y= 6691196.25, Z= 265.802829, surface='surface1',
                           pole_vector= (0,0,1))

#2D & 3D plotting
p2d.plot_data(ax, cell_number=5)
p3d.plot_data()
```

Fig 4. 15 Adding orientation points

4.5.1.8 Grid

```
In [13]: geo_model.grid

Out[13]: Grid Object. Values:
array([[ 3.3763e+05,  6.6407e+06, -1.7810e+04],
       [ 3.3763e+05,  6.6407e+06, -1.7430e+04],
       [ 3.3763e+05,  6.6407e+06, -1.7050e+04],
       ...,
       [ 3.9937e+05,  6.7093e+06,  5.0000e+01],
       [ 3.9937e+05,  6.7093e+06,  4.3000e+02],
       [ 3.9937e+05,  6.7093e+06,  8.1000e+02]])
```

Fig 4. 16 Calling the grid library

4.5.1.9 Compute the Model

```
In [14]: gp.compute_model(geo_model)

. . .

Out[14]: Lithology ids
         [2. 2. 2. ... 2. 1. 1.]
```

Fig 4. 17 Computing the model

4.5.1.10 Interpolation

```
In [15]: #Interpolate the 2D surface
p2d.plot_contacts(ax, cell_number=5)

#Interpolate the 3D surface
p3d.plot_surfaces()
p3d.plot_structured_grid()

Out[15]: [StructuredGrid (0x1d9d04a5820)
  N Cells:      117649
  N Points:     125000
  X Bounds:     3.376e+05, 3.994e+05
  Y Bounds:     6.641e+06, 6.709e+06
  Z Bounds:     -1.781e+04, 8.100e+02
  Dimensions:   50, 50, 50
  N Arrays:     2,
  <matplotlib.colors.ListedColormap at 0x1d9d367bd00>]
```

Fig 4. 18 Interpolate the 2D and 3D surface

4.5.1.11 Lithology

```
In [17]: # Litology
gp.plot_2d(geo_model, show_data=True,figsize=(12,5))
plt.show()

In [18]: # lithology in scalar-field solution

gp.plot_2d(geo_model, show_data=False, show_scalar=True, show_lith=False)
plt.show()
```

Fig 4. 19 Calling the lithology library

4.5.1.12 Adding more surfaces

```
In [20]: # adding new surfaces including the basement

geo_model.add_surfaces(["surface3", "basement"])
```

```
Out[20]:
```

| | surface | series | order_surfaces | color | id |
|---|----------|----------------|----------------|---------|----|
| 0 | surface1 | Default series | 1 | #015482 | 1 |
| 1 | surface2 | Default series | 2 | #9f0052 | 2 |
| 2 | surface3 | Default series | 3 | #ffe000 | 3 |
| 3 | basement | Default series | 4 | #728f02 | 4 |

Fig 4. 20 Adding new surfaces including the basement

4.5.1.13 Adding more surface points

```
In [21]: # adding two more surface points
geo_model.add_surface_points(X=340231.045846, Y=6687301.780702, Z=-983.011697, surface='surface2', idx=3)
geo_model.add_surface_points(X=341657.530298, Y=6687469.941049, Z=-3728.998697, surface='surface2', idx=4)

# Plotting
p2d.plot_data(ax, cell_number=11)
p3d.plot_surface_points()
```

Out[21]: (vtkmodules.vtkRenderingOpenGL2.vtkOpenGLActor)000001D9D04A5340

Fig 4. 21 Adding two more surface points

4.5.1.14 Grid and Compute

```
In [22]: geo_model.grid
```

Out[22]: Grid Object. Values:

```
array([[ 3.3763e+05,  6.6407e+06, -1.7810e+04],
       [ 3.3763e+05,  6.6407e+06, -1.7430e+04],
       [ 3.3763e+05,  6.6407e+06, -1.7050e+04],
       ...,
       [ 3.9937e+05,  6.7093e+06,  5.0000e+01],
       [ 3.9937e+05,  6.7093e+06,  4.3000e+02],
       [ 3.9937e+05,  6.7093e+06,  8.1000e+02]])
```

```
In [23]: gp.compute_model(geo_model)
```

Out[23]:

```
Lithology ids
[4. 4. 4. ... 1. 1. 1.]
```

Fig 4. 22 Calling the grid library and compute the model again

4.7.1.15 Lithology

```
In [24]: # Litology
gp.plot_2d(geo_model, show_data=True, figsize=(12,5))
plt.show()
```

```
In [25]: # Litthology in scalar-field solution
gp.plot_2d(geo_model, show_data=False, show_scalar=True, show_lith=False)
plt.show()
```

Fig 4. 23 Calling the lithology library

4.5.1.16 Topology

```
In [26]: #topology model if needed

geo_model.set_topography(d_z=(350, 750))

Active grids: ['regular' 'topography']

Out[26]: Grid Object. Values:
array([[ 3.37630000e+05,  6.64070000e+06, -1.78100000e+04],
       [ 3.37630000e+05,  6.64070000e+06, -1.74300000e+04],
       [ 3.37630000e+05,  6.64070000e+06, -1.70500000e+04],
       ...,
       [ 4.00000000e+05,  6.70714286e+06,  6.36588938e+02],
       [ 4.00000000e+05,  6.70857143e+06,  6.40636816e+02],
       [ 4.00000000e+05,  6.71000000e+06,  6.43542714e+02]])

In [*]: #topology model execute

gp.compute_model(geo_model)
gp.plot_2d(geo_model, show_topography=True)
plt.show()

# sphinx_gallery_thumbnail_number = 9
gpv = gp.plot_3d(geo_model, plotter_type='basic', show_topography=True, show_surfaces=True,
                show_lith=True,
                image=False)
```

Fig 4. 24 Adding topography to execute the final model

4.6 Testing

Unit testing guarantees that all code satisfies quality guidelines before it is bringing into action. This guarantees a solid designing environment where quality is principal. Throughout the item improvement life cycle, unit testing sets aside time and cash, and assists engineers with composing better programming, more efficiently. Testing is significant since it finds bugs before the conveyance to the customer, which ensures the nature of the software. It makes the software more solid and simple to utilize. Completely tried programming guarantees dependable and elite software activity. Below table is the unit testing of the 3D geological borehole model: -

Table 4. 1 Test case for 2D cross- section model

| | | | | | | |
|----------------|------------------|------------------------------------|--|---------------|---------------------|------------------|
| Project | | 3D Geological Borehole Model Using | | Date | | 29/05/2021 |
| Name | | Python | | | | |
| Test | | Cross section | | Pre-requisite | Internet connection | |
| No | Test Case | Procedure | | Result | | Fail/Pass |

| | | | | |
|---|-------------------------|--|------------------------------------|------|
| 1 | 2D cross- section model | After plotting the surface point and orientations 2D cross-section model will show on matplotlib | Display the 2D cross-section model | Pass |
|---|-------------------------|--|------------------------------------|------|

Table 4. 2 Test case for Lithology

| | | | | | |
|---------------------|---|---|-----------------------|---------------------|--|
| Project Name | 3D Geological Borehole Model Using Python | | Date | 29/05/2021 | |
| Test | Lithology | | Pre-requisite | Internet connection | |
| No | Test Case | Procedure | Result | Fail/Pass | |
| 1 | Lithology model | The lithology model can be easily visualize in the 2D section directly. | Display the Lithology | Pass | |

Table 4. 3 Test case for 3D Geological Borehole Model with different layers.

| | | | | |
|----------------|------------------------------------|---------------|---------------------|------------|
| Project | 3D Geological Borehole Model Using | | Date | 29/05/2021 |
| Name | Python | | | |
| Test | 3D Geological Borehole Model | Pre-requisite | Internet connection | |

| No | Test Case | Procedure | Result | Fail/Pass |
|----|------------------------------|--|--|-----------|
| 1 | 3D Geological Borehole Model | After the 2D cross-section, GemPy implies the surfaces to visualize the 3D Model | Display the 3D geological borehole model with different layers and can be able to hover the model. | Pass |

4.7 User acceptance testing response

The system is given to 14 people for testing its functionality and the respondent are from the Faculty of Computer Science and Information Technology (FCSIT) and Faculty of Engineering (FENG) students. After using the system the test is carried out by the respondent. There are 3 questions regarding the system use and one optional question based on an understanding of the system. The full Questionnaire will be attached to Appendix B and below are the outcomes of the Questionnaires.

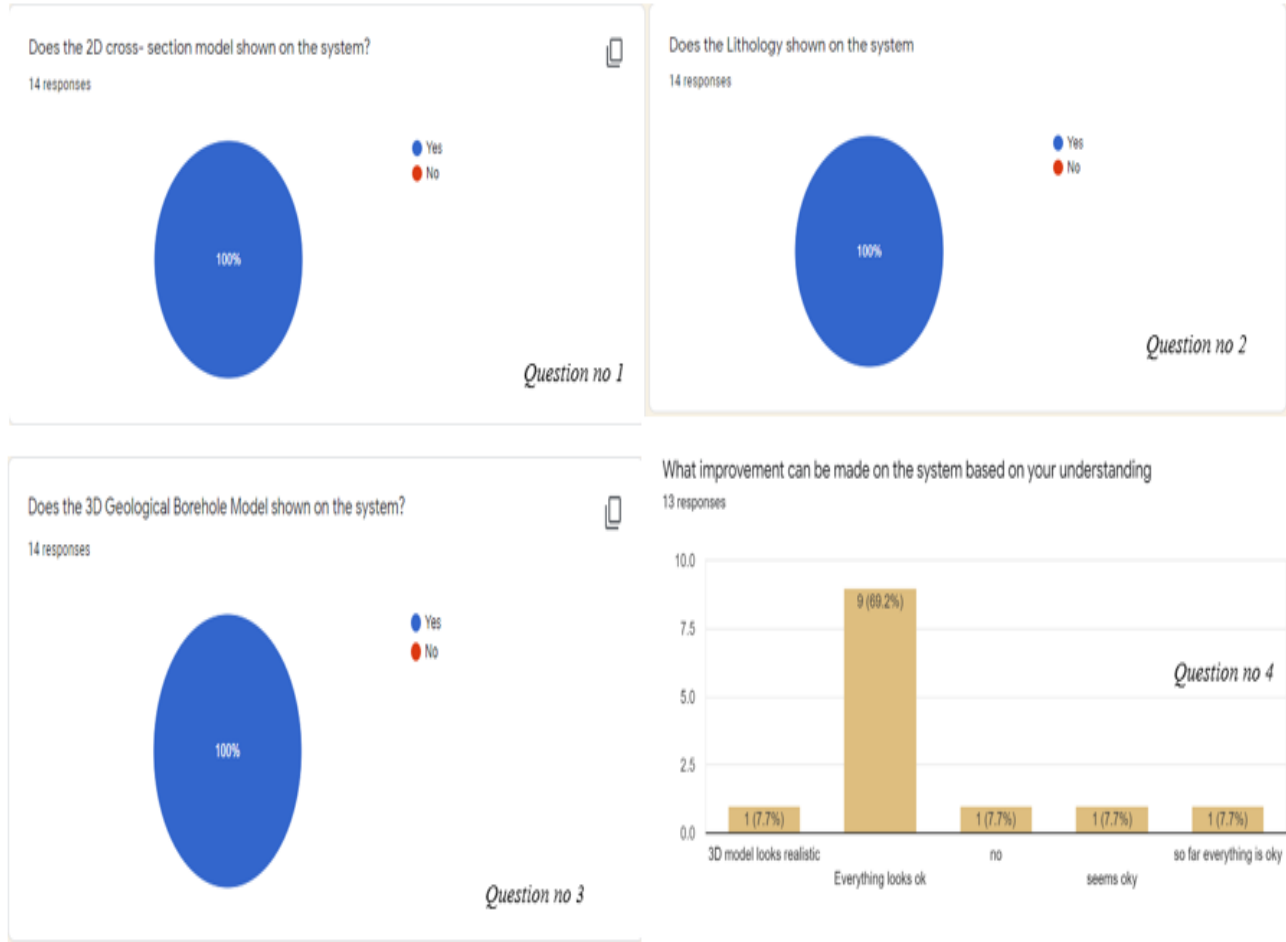


Fig 4. 25 Questions (Optional Question)

From the above result, it shows that all the functionality is working properly and the feedback from the users is also in favor. In addition, some of the suggestions are they are satisfied with the outcomes but for further use, the system needs to be tested on a different kind of platform to prevent user understanding complexity and to avert any bugs.

4.8 Conclusion

In conclusion, the implementation and testing of the 3D geological borehole model have been described and include the explanation of the software. The process of prototyping from the first prototyping cycle until the refined prototyping has been explained adequately. Moreover, the

figure shown above is for a better and clear vision regarding the prototype. The testing strategy is being used to test the functionality of the whole system, wherein every test case reveals all the functionality is working properly. And for the non-functionality testing, a user runs the system to determine the functionality of the system.

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Introduction

This chapter summarizes the entire project such as project outcome, future work, and project limitations. 3D geological borehole model is intricate but with the help of GemPy, the entire process to create the model becomes more efficient. But still, there are some limitations due to borehole data unavailability. Creating a model in GemPy is a convenience but the installation process and containing the data structure and necessary functionality are difficult at the beginning. The main objectives of this 3D geological borehole model are to show the 2d cross-section, lithology, and 3D realistic model. With the help of a 2D cross-section model, the user can be able to identify the soil structure. With the help of borehole data, lithology will demonstrate the differences in soil layers. And with the help of a 3D model users can be able to detect the probabilistic soil structure, soil layers, surface points, soil condition, and basement.

5.2 Project Outcome

The main objectives were to project the borehole data into a 3D geological representation model and to represent research on open-source geological tools or software. After completing the project the outcomes are stated in table 5.1.

Table 5. 1 Project Outcomes

| Objectives | Achievements |
|---|---|
| To project the borehole data into a 3D geological model | From the software GemPy, the model development and design have been done successfully. Using borehole data GemPy was able to construct a 2D cross-section model, lithology, and expected 3D geological model. |

| | |
|---|--|
| | In GemPy, we can manually add more surface points in real-time. |
| To represent research an open-source geological tools or software | In chapter 2, we represent three open-source tools which are Leapfrog, GemPy, and QGIS. Every software has its limitations and working capability. Leapfrog is more efficient than others but for further procedure need to buy the license and it costs a lot. QGIS is crash-sensitive especially during extended operations such as interpolation. Due to the crash the software being shut down immediately. And GemPy is user-friendly python-based can do complex 3D geological models and most importantly GemPy is open source and easily accessible and does not any cost. But due to the limitations, GemPy could not able to generate the 3D borehole model it can only execute the surface model. |

5.3 Project Limitation

The software GemPy that is used is Python-based and open-source geological modeling library. GemPy is capable to execute intricate models including various features because it is designed to

be efficiently embedded in expected frameworks for managing subsurface structures. Still, after completing the project there are some limitations arise. The project limitations are stated below:

- Due to the latest update of GemPy, there is a conflict between Anaconda and grampy's compatibility. For this reason, mini conda need to use for further progress.
- GemPy uses VTK (Virtual Toolkits) to demonstrate the 3D geological model but the outdated version of VTK becomes conflictive regarding Theano installation. Because there are some compatibility problems with python and VTK.
- As GemPy is new and still under development so to construct the geological model it needs minimum specific data and they are: - surface points and one orientation layer. Without this minimum data, GemPy will not execute the model.
- In GemPy, the borehole data is used to demonstrate the geological subsurface only it will not display the model of a borehole.

5.4 Project Future Work

In this report, addressing the complete 3D geological borehole model as well as its limitations we come up with some recommendations for future work but still, there are some different modeling approaches in this geological model field. Because GemPy is not limited only to visualizing the models, but it also gives the back-end developer an enormous available option of visualization to accomplish their needs. After completing this project there are some ideas regarding developing this system further the recommendation is stated below:

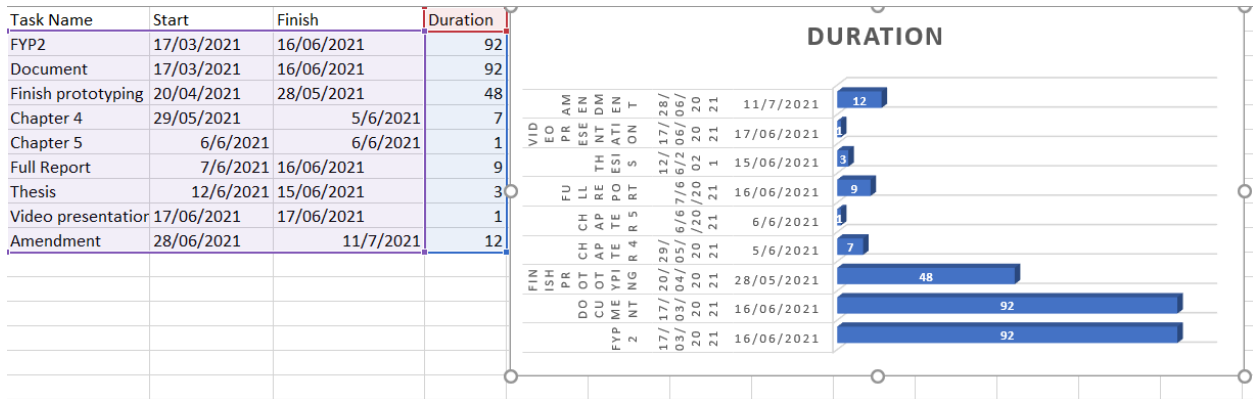
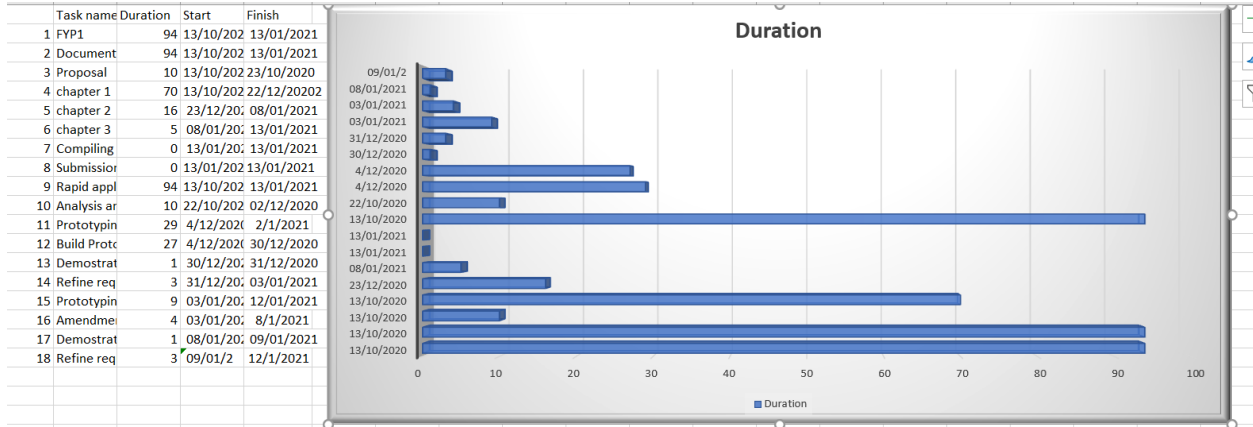
- Allow users to input the data both in the system and on Github to retrieve the data.
- GemPy not only shows the 3D geological model it also has different types of features like showing the full faults network and Folds on a single layer or all the layers.

- Showing more probabilistic 3d model with the help of topology.
- Soon GemPy is planning for the visualization of the models using virtual reality so with the help of this algorithm we can demonstrate the model in virtual reality.


References

- Caumon, G., Collon-Drouaillet, P., Veslud, C. L., & Sausse, S. V. (2009). Surface-Based 3D Modeling of Geological Structures.
- Eldrandaly, K., & Naguib, S. (2011). A Knowledge-Based System for GIS Software Selection .
- Meiracker, I. v. (2019). Exploring 3D functionalities: A research into. *Geographical Information Management and Application*.
- MITAS, L., & H MITASOVA. (1999). Spatial interpolation. *Geographic Information Systems: Principles, Techniques, Management and Applications*, 481-492.
- Njeban, H. S. (2018). Journal of Geographic Information System. *Journal of Geographic Information System*, 19.
- Sullivan, C. B. (2020). PYVISTA: MANAGING & VISUALIZING GEOSPATIAL DATA USING AN OPEN-SOURCE FRAMEWORK.
- Tan, Q., & Xu, X. (2014). Comparative Analysis of Spatial Interpolation Methods:. *Sensors & Transducers*, 155-163.
- Varga, M. d., Schaaf, A., & FlorianWellmann. (2019). GemPy 1.0: open-source stochastic geological modeling and inversion.
- Features of Leapfrog works Retrieve from https://www.seequent.com/products-solutions/leapfrog-works/?gclid=CjwKCAiAgc-ABhA7EiwAjev-j0VmngOXsVZ4aUSW06F5RLcSG_RQbqbP0g6Os7RSJI9_TILrm_8vRORoC9fQQAuD_BwE#features
- Features of QGIS Retrieve from https://docs.qgis.org/2.8/en/docs/user_manual/preamble/features.html

Appendix A- Gantt chart of project schedule



Appendix B- User Acceptance Testing Form



Final Year Project (User Acceptance Testing)

Thank you for participating as my respondent. This survey is conducted regarding the 3D Geological Borehole Model by using Python system is functioning

Matric Number

Your answer _____

Does the 2D cross- section model shown on the system?

☐ Yes

☐ No

Does the Lithology shown on the system

☐ Yes

☐ No

Does the 3D Geological Borehole Model shown on the system?

☐ Yes

☐ No

What improvement can be made on the system based on your understanding

Your answer _____

Submit

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#)

Google Forms

Appendix C- Programming Code

```
In [1]: import gempy as gp
import sys, os
os.environ["THEANO_FLAGS"] = "mode=FAST_RUN,device=cpu"

import gempy as gp

# Importing aux libraries
from ipywidgets import interact
import numpy as np
import matplotlib.pyplot as plt

# Embedding matplotlib figures in the notebooks
%matplotlib qt5

No module named 'osgeo'
```

```
In [2]: geo_model = gp.create_model('Model12')
# of the regular grid
geo_model = gp.init_data(geo_model,
                        extent=[337000,400000,664000,671000,-18000,1000], # for the regular grid
                        resolution=[50, 50, 50]) # for the regular grid

Active grids: ['regular']
```

```
In [3]: # Calling the Library Theano
gp.set_interpolator(
    geo_model,
    output=['geology'],
    theano_optimizer='fast_compile')

Setting kriging parameters to their default values.
Compiling theano function...

Level of Optimization: fast_compile
Device: cpu
Precision: float64
Number of faults: 0
Compilation Done!
Kriging values:
range          96072.888996
$C_o$          219761904.761905
drift equations [3]
```

```
Out[3]: <gempy.core.interpolator.InterpolatorModel at 0x1d9cdeae80>
```

```
In [4]: #calling the matplotlib
p2d = gp.plot_2d(geo_model, section_names=None,
                direction=None, cell_number=None)
```

```
In [5]: # 2D cross section for the model which is perpendicular to y axis
ax = p2d.add_section(cell_number=1, direction='y')
```

```
In [6]: # Calling the Pyvista
p3d = gp.plot_3d(geo_model, plotter_type='background', notebook=False)
```

```
In [7]: geo_model.surfaces
```

```
Out[7]:
```

| | surface | series | order_surfaces | color | id |
|---|----------|----------------|----------------|---------|----|
| 0 | surface1 | Default series | 1 | #015482 | 1 |
| 1 | surface2 | Default series | 2 | #9f0052 | 2 |

```
In [8]: #default surfaces
geo_model.set_default_surfaces()
```

```
Out[8]:
```

| | surface | series | order_surfaces | color | id |
|---|----------|----------------|----------------|---------|----|
| 0 | surface1 | Default series | 1 | #015482 | 1 |
| 1 | surface2 | Default series | 2 | #9f0052 | 2 |

```
In [9]: # Add a surface point.Here idx is None so next available index will be used
geo_model.add_surface_points(X=352798.25, Y= 6698505.5, Z= 254.579992,surface='surface1', idx=0)

#2D plot
p2d.plot_data(ax, cell_number=11)

#3D plot
p3d.plot_surface_points()
```

```
Out[9]: (vtkmodules.vtkRenderingOpenGL2.vtkOpenGLActor)000001D9D14CF40
```



```
In [10]: #Adding more surface points
geo_model.add_surface_points(X=371748.625, Y=6683887, Z=282.8574779999999, surface='surface1', idx=1)
geo_model.add_surface_points(X=392594.063, Y=6668185, Z=228.405954, surface='surface1', idx=2)

# 2D & 3D Plotting
p2d.plot_data(ax, cell_number=11)
p3d.plot_surface_points()
```

```
Out[10]: (vtkmodules.vtkRenderingOpenGL2.vtkOpenGLActor)000001D9D15942E0
```

```
In [26]: geo_model.surface_points
```

```
Out[26]:
```

| | X | Y | Z | smooth | surface |
|---|------------|-----------|------------|----------|----------|
| 0 | 352798.250 | 6698505.5 | 254.579992 | 0.000001 | surface1 |
| 1 | 371748.625 | 6683887.0 | 282.857478 | 0.000001 | surface1 |
| 2 | 392594.063 | 6668185.0 | 228.405954 | 0.000001 | surface1 |

```
In [11]: #Adding orientation point
geo_model.add_orientations(X=362273.4375, Y= 6691196.25, Z= 265.802829, surface='surface1',
                           pole_vector=(0,0,1))

#2D & 3D plotting
p2d.plot_data(ax, cell_number=5)
p3d.plot_data()
```

```
In [12]: geo_model.surface_points
```

```
Out[12]:
```

| | X | Y | Z | smooth | surface |
|---|------------|-----------|------------|----------|----------|
| 0 | 352798.250 | 6698505.5 | 254.579992 | 0.000001 | surface1 |
| 1 | 371748.625 | 6683887.0 | 282.857478 | 0.000001 | surface1 |
| 2 | 392594.063 | 6668185.0 | 228.405954 | 0.000001 | surface1 |

```
In [13]: geo_model.grid
```

```
Out[13]: Grid Object. Values:
array([[ 3.3763e+05,  6.6407e+06, -1.7810e+04],
       [ 3.3763e+05,  6.6407e+06, -1.7430e+04],
       [ 3.3763e+05,  6.6407e+06, -1.7050e+04],
       ...,
       [ 3.9937e+05,  6.7093e+06,  5.0000e+01],
       [ 3.9937e+05,  6.7093e+06,  4.3000e+02],
       [ 3.9937e+05,  6.7093e+06,  8.1000e+02]])
```

```
In [14]: gp.compute_model(geo_model)
```

```
Out[14]:
Lithology ids
[2. 2. 2. ... 2. 1. 1.]
```

```
In [15]: #Interpolate the 2D surface
p2d.plot_contacts(ax, cell_number=5)

#Interpolate the 3D surface
p3d.plot_surfaces()
p3d.plot_structured_grid()
```

```
Out[15]: [StructuredGrid (0x1d9d04a5820)
N Cells:      117649
N Points:     125000
X Bounds:     3.376e+05, 3.994e+05
Y Bounds:     6.641e+06, 6.709e+06
Z Bounds:     -1.781e+04, 8.100e+02
Dimensions:   50, 50, 50
N Arrays:     2,
<matplotlib.colors.ListedColormap at 0x1d9d367bd00>]
```

```
In [16]: sol = gp.compute_model(geo_model)
```

```
In [17]: # Litology
gp.plot_2d(geo_model, show_data=True, figsize=(12,5))
plt.show()
```

```
In [18]: # lithology in scalar-field solution
gp.plot_2d(geo_model, show_data=False, show_scalar=True, show_lith=False)
plt.show()
```

```
In [19]: geo_model.surfaces
```

```
Out[19]:
```

| | surface | series | order_surfaces | color | id |
|---|----------|----------------|----------------|---------|----|
| 0 | surface1 | Default series | 1 | #015482 | 1 |
| 1 | surface2 | Default series | 2 | #9f0052 | 2 |

```
In [20]: # adding new surfaces including the basement
geo_model.add_surfaces(["surface3", "basement"])
```

Out[20]:

| | surface | series | order_surfaces | color | id |
|---|----------|----------------|----------------|---------|----|
| 0 | surface1 | Default series | 1 | #015482 | 1 |
| 1 | surface2 | Default series | 2 | #9f0052 | 2 |
| 2 | surface3 | Default series | 3 | #f0e000 | 3 |
| 3 | basement | Default series | 4 | #728f02 | 4 |

```
In [21]: # adding two more surface points
geo_model.add_surface_points(X=340231.045846, Y=6687301.780702, Z=-983.011697, surface='surface2', idx=3)
geo_model.add_surface_points(X=341657.530298, Y=6687469.941049, Z=-3728.998697, surface='surface2', idx=4)

# Plotting
p2d.plot_data(ax, cell_number=11)
p3d.plot_surface_points()
```

Out[21]: (vtkmodules.vtkRenderingOpenGL2.vtkOpenGLActor)000001D9D04A5340

In [22]: geo_model.grid

Out[22]: Grid Object. Values:
array([[3.3763e+05, 6.6407e+06, -1.7810e+04],
 [3.3763e+05, 6.6407e+06, -1.7430e+04],
 [3.3763e+05, 6.6407e+06, -1.7050e+04],
 ...,
 [3.9937e+05, 6.7093e+06, 5.0000e+01],
 [3.9937e+05, 6.7093e+06, 4.3000e+02],
 [3.9937e+05, 6.7093e+06, 8.1000e+02]])

In [23]: gp.compute_model(geo_model)

Out[23]: Lithology ids
[4. 4. 4. ... 1. 1. 1.]

```
In [24]: # Litology
gp.plot_2d(geo_model, show_data=True, figsize=(12,5))
plt.show()
```

```
In [25]: # Lithology in scalar-field solution
gp.plot_2d(geo_model, show_data=False, show_scalar=True, show_lith=False)
plt.show()
```

Out[23]: Lithology ids
[4. 4. 4. ... 1. 1. 1.]

```
In [24]: # Litology
gp.plot_2d(geo_model, show_data=True, figsize=(12,5))
plt.show()
```

```
In [25]: # Lithology in scalar-field solution
gp.plot_2d(geo_model, show_data=False, show_scalar=True, show_lith=False)
plt.show()
```

```
In [26]: #topology model if needed

geo_model.set_topography(d_z=(350, 750))

Active grids: ['regular' 'topography']
```

Out[26]: Grid Object. Values:
array([[3.37630000e+05, 6.64070000e+06, -1.78100000e+04],
 [3.37630000e+05, 6.64070000e+06, -1.74300000e+04],
 [3.37630000e+05, 6.64070000e+06, -1.70500000e+04],
 ...,
 [4.00000000e+05, 6.70714286e+06, 6.36588938e+02],
 [4.00000000e+05, 6.70857143e+06, 6.40636816e+02],
 [4.00000000e+05, 6.71000000e+06, 6.43542714e+02]])

```
In [*]: #topology model execute

gp.compute_model(geo_model)
gp.plot_2d(geo_model, show_topography=True)
plt.show()

# sphinx_gallery_thumbnail_number = 9
gpv = gp.plot_3d(geo_model, plotter_type='basic', show_topography=True, show_surfaces=True,
                 show_lith=True,
                 image=False)
```