MinEx CRC Limited
26 Dick Perry Avenue, Kensington, WA, 6151
PO Box 1130, Bentley, WA, 6102, Australia
admin@minexcrc.com.au

MinEx CRC
ABN: 66 625 533 913

MinEx CRC provides financial support to the value of $1K to promote Honours and Masters by Coursework projects that are aligned with the mission of MinEx CRC and to encourage young researchers toward a career in mineral exploration research. Projects are not restricted to MinEx CRC Participants and Affiliates.

Please note that the content of this thesis has not been subjected to peer-review and subsequent corrections.

Australian Government
Department of Industry, Science and Resources
Cooperative Research Centres Program

Drillhole Data Density as a Control on 3D Modelling of Banded Iron Formations from the Hamersley Basin, Western Australia

Özgür Yedek

(23147231)

Supervisors

Prof Mark Jessell
Dr Guillaume Pirot

This thesis is submitted to fulfil the requirement for a Master of Geoscience by way of Coursework & Dissertation

(6738 words)

Faculty of Science

The University of Western Australia

August, 2023

TABLE OF CONTENTS

1. ACKNOWLEDGEMENTS

I would like to extend my heartfelt appreciation to my supervisor, Prof Mark Jessell, for his unwavering dedication, invaluable insights, and continuous encouragement throughout this research journey. His mentorship has been instrumental in shaping the direction and quality of this thesis. His guidance has not only enriched my understanding of the subject matter but has also enhanced my research and analytical skills.

I am also indebted to my co-supervisor, Dr. Guillaume Pirot, for his insightful feedback, constructive criticism, and guidance that have significantly contributed to the refinement of this research. His expertise and willingness to engage in meaningful discussions have been instrumental in shaping the depth and scope of this work.

I am grateful to BHP Group Ltd for providing the essential data required for this research. Their support in granting access to the necessary resources has been pivotal in conducting a comprehensive analysis and drawing meaningful conclusions.

I wish to extend a special acknowledgment to my friend Kübra Akbulut, with whom I had the opportunity to work on the same research project. Her incisive contributions to the interpretation of the research findings and her unswerving cooperation have added layers of depth and insight, enriching the entirety of this study. Her camaraderie throughout my research study has been a wellspring of motivation and encouragement.

I express my sincere gratitude to my sponsors, the Ministry of National Education and the General Directorate of Mineral Research and Exploration of the Republic of Türkiye, for providing me with the opportunity and financial support to pursue my master's degree at the University of Western Australia (UWA).

I would like to extend my heartfelt gratitude to my family for their unwavering support, patience, and encouragement throughout this academic journey. Their belief in my abilities and constant motivation has been my driving force.

Lastly, I appreciate all my friends and colleagues who provided valuable insights, discussions, and encouragement throughout this research endeavour.

2. ABSTRACT

This research thesis investigates the impact of drilling density on the fidelity of 3D geological models. With the focal point on the Hamersley basin in Western Australia, stratigraphic log data stemming from 115 drillholes has been scrutinized. The dataset primarily encapsulates banded iron formations, dating back between 2630 million years and 2450 million years, forming the majority of the units.

The core approach of the study involves the incremental removal of drillholes to observe the evolving influence of drillhole density on model quality. Within the context of the sampling process, a two-fold grouping methodology has been implemented. In the first group, a systematic 5x5 grid subdivides the study area. From each cell, a set number of drillholes ensures comprehensive coverage. Conversely, the second group, global sampling, employs a distinct approach by selecting samples randomly, irrespective of their spatial coordinates. The number of drillings in the sample groups was determined by reducing the original data by 10% each. In this way, ten groups were formed for each sample group.

The findings underscore the direct correlation between sample size and model quality. In instances where sub-area sampling is utilized, the model quality correlates positively and consistently with the increased proportion, from 40% to 100% of sampled drillholes. On the other hand, models derived from global sampling exhibit a decline in quality for subsets comprising fewer than 70% of the total samples.

These insights illuminate the significance of sampling and size in shaping the quality of 3D geological models. The superiority of sub-area sampling in ensuring a stable and escalating model quality is evident. This research serves as a valuable guide for researchers in the realm of geological modelling, advocating for meticulous sample selection and emphasizing the paramount role of sample size. Future endeavours may benefit from larger sample sizes to achieve even more precise and comprehensive outcomes.

3. INTRODUCTION

Sampling strategies for structural data and their impacts on the accuracy and fidelity of 3D models have been explored (Putz *et al.*, 2006). However, the impact of drillhole sampling on the model quality has not yet been tested using a similar approach. A better understanding of just how many drillholes are required to establish a robust 3D model in a given setting would have obvious economic benefits. In this study, a densely drilled part of the Hamersley basin, W.A., was modelled using Loop3D software and the effects of drilling density on the 3D models were tested. Advancements in 3D geological modelling software have facilitated the creation of workflows previously unavailable in commercial systems, specifically in stochastic modelling and model variability (Grose, Ailleres, Laurent, & Jessell, 2021; Pirot *et al.*, 2022). Loop3D, the 3D modelling software utilised in this study, is an open-source platform for probabilistic geological and geophysical modelling in 3D. The development of the software is supported by the Australian Territory/State and Federal Geological Surveys, the Australian Research Council, MinEx CRC and industry-leading mining companies (Loop3D, 2020).

This research aims to investigate the influence of drillhole data density on the reliability and precision of 3D models generated using Loop3D. Within the study area, a densely drilled part of the Hamersley Basin is modelled through Loop3D software (Figure 1). The drillholes were progressively sub-sampled to test the impact of the density of drillholes on the 3D modelling. Thus, the similarities and differences between the models generated were evaluated, and the dependence on data density was evaluated. The drillhole log data used within the study is provided by BHP Group Ltd. Since the drillhole data is confidential, their exact location cannot be shown on the map.

Figure 1. Approximate location of the study area.

3.1. Research Aims & Objectives

The accurate representation of subsurface geology, particularly the relationships between geological formations and structures, is essential for effective and practical resource evaluation and exploration targeting processes. Ideally, this representation will be constrained by a comprehensive suite of drillhole log data, including the quantity of drillhole, density, spatial distribution, and lithological and structural features. By integrating and interpreting the drilling data, the aim is to gain a deeper understanding of geological structures and their spatial connectivity, providing valuable insight into geological processes and controls over BIF accumulation within the study area.

The second objective is to evaluate the Loop3D geological modelling software using stochastic and model comparison methods. As the software is still under development, it is crucial to assess its performance, accuracy, and reliability. Stochastic modelling techniques assessed the variability and uncertainty associated with Loop3D-generated models. Model comparison methods will be applied to compare the Loop3D models with other established geological models to evaluate their agreement, biases, and limitations.

3.2. Significance and Outcomes

This research aims to contribute to the field of geoscience by providing insights into the importance of drilling density in establishing geological relationships. A secondary goal is to evaluate the Loop3D software for the 3D modelling of BIF deposits. BIF deposits are economically valuable, and understanding their distribution and characteristics is crucial for resource evaluation and exploration targeting. The significance of this project lies in its potential to improve the accuracy and reliability of 3D geological models, enhance resource evaluation strategies, and optimise exploration efforts. The findings will enhance our understanding of the optimal drillhole data density required for accurate modelling and validate the capabilities of Loop3D in representing complex geological structures. The objective of this study is to enhance resource assessment methodologies, optimise exploration targeting endeavours, and deepen the comprehension of the geological development of the Hamersley Basin.

By examining the effects of drillhole density on 3D models of BIF, the project aims to provide valuable insights into the optimal density of drillhole data required for an accurate representation of subsurface geology. This knowledge is essential for resource evaluation as accurate modelling of BIF deposits is crucial for estimating their volume and quality. Improved resource evaluation techniques will enable better-informed decisions regarding the economic viability and utilisation of BIF resources in the Hamersley Basin.

Additionally, the project will contribute to advancing 3D modelling methodologies in geology. As a new 3D geological modelling software, Loop3D presents an opportunity to evaluate its performance and reliability in modelling BIF deposits. By utilising Loop3D and varying the drilling density, the project will validate and provide valuable feedback for further improvements and enhancements to the software. This validation process is crucial in ensuring the effectiveness and applicability of Loop3D for modelling complex geological structures.

4. LITERATURE REVIEW

4.1. Previous Studies on Hamersley Basin

The Hamersley Basin is located in the northwest of Western Australia (Figure 2). It is renowned for its rich iron mineral resources, particularly Banded Iron Formation (BIF) deposits. One of the three parts of Western Australia's Pilbara Craton, the basin covers an area of about 80,000 km^2. The Hamersley basin has been known for over a century and remains attractive for scientific research.

Geologically, the basin was first investigated by Maitland and Jackson (1903). During the following decade, this research was followed by Maitland (1904, 1905, 1906, 1908, 1909). Talbot (1920) named the rocks in the basin the 'Nullagine Series' by revealing the unconformity between the underlying Pilbara Block. In the following years, several notes and descriptions of certain rock formations were published by Forman (1937), Finucane (1939), and Miles (1942). Subsequently, the 1:250,000 scale geological map, which served as the basis for many subsequent studies, was prepared by Maitland and Talbot on behalf of the Western Australian Geological Survey. MacLeod (1966) synthesised and published the existing data as a 1:500,000-scale regional geology map, including almost the entire basin. The term Hamersley Basin was first formally defined by Trendall (1968) as the sedimentary basin containing the Precambrian Hamersley Group. His continuous contributions from 1963 to 2004 have taken an important place in the geological understanding of the basin.

Blockley (1975) demonstrated the presence of copper-lead-zinc within the Fortescue Group and the principal iron ore in the basin. The first geochemical study of certain minerals was performed by Trendall (1976) and Ewers and Morris (1980). Banded Iron Formations, the principal ore formation of the Hamersley basin, were studied by Gole and Klein (1981) and Smith *et al.* (1982). Advancing mineral exploration methods have helped to provide knowledge about the metamorphism processes and conditions of the region (Horwitz, 1976; Blockley, 1979; Trendall, 1979). The first systematic study on the metamorphic processes of the area was by Smith *et al.* (1982). Morris and Horwitz (1983) studied the basin's sediments in terms of their chemistry or origin. They suggested that the units' composition comprises biochemical (carbonates, chert and iron formations) and chemical/pyroclastic (shales).

The recent studies are generally on the ages of the units in the basin (Thorne & Trendall, 2001; Trendall *et al.*, 2004). The depositional chronology of the Hamersley

Group is according to the SHRIMP (Sensitive High-Resolution Ion Microprobe) zircon dating revealed approximately 330Ma of continuous fill model (Trendall *et al.*, 2004).

4.2. Stratigraphy

The Hamersley Basin is one of the three components of the Mt Bruce Supergroup (Trendall *et al.*, 1998) (Figure 2). The basin is a volcano-sedimentary basin deposited between 2.75 Ga and 2.45 Ga in the Precambrian era (Trendall *et al.*, 2004). The strata accumulated during this period are defined as Mt Bruce Supergroup (Trendall *et al.*, 2004). The components of the Mt Bruce group are the Fortescue Group, which is the basement of the supergroup, Banded Iron Formations (BIF) containing the Hamersley Group and the uppermost Turee Creek Group (Figure 2).

The older granite-greenstone terrane of the Pilbara craton is unconformably overlain by the Fortescue Group (Trendall *et al.*, 2004; Williams, 2023). The thickness of the group is up to ~7 km and consists of mafic lava and interbedded felsic and mafic volcanoclastic materials (Figure 3). The Hamersley Group, which lies on the Fortescue Group, has a thickness of approximately 2.5 km and includes a Banded Iron Formation (BIF) that can reach a thickness of 300 metres (Trendall & Blockey, 1970; Ewers & Morris, 1980; Trendall *et al.*, 2004; Shibuya *et al.*, 2010; Figure 3). Although it is stated that the Fortescue Group overlaps the Hamersley Group conformably by most researchers, there are also studies stating that the Hamersley Group overlies the Fortescue Group unconformably (MacLeod *et al.*, 1962; Trendall *et al.*, 2004; Shibuya *et al.*, 2010).

At the lowest level of the Hamersley group is the Marra Mamba formation, which overlies the Fortescue Group conformably (Lascelles, 2000; Trendall *et al.*, 2004; Figure 3). It consists of three members, from top to bottom, Mt Newman, MacLeod and Nammuldi, alternating chert and BIF layers, forming a distinctive “taconite” texture (Trendall, 1990). The unit is aged 2629 Ma and has a 248m thickness (Trendall *et al.*, 2004). Marra Mamba Formation is overlaid by the 520 m thick Wittenoom Formation (Figure 2). According to the dating analysis conducted by Trendall *et al.* (2004), the formation is 2596 Ma – 2506 Ma of age. The West Angelas member, at the base of the Wittenoom Formation, consists of a BIF-bearing shale alternation. It is followed by the dolomitic Paraburdoo member. The uppermost part of the Wittenoom

Formation, the Bee Gorge member, is dominated by shales with minor ferruginous chert and volcaniclastics (Simonson *et al.*, 1993). Marra Mamba and Wittenoom formations are interpreted as a passive margin-deep marine shelf regarding the depositional environment (Howard & Martin, 2018). Mount Sylvia Formation has a 30 m thickness and consists of shales intercalated with thin BIF and chert horizons. The lithology of the following Mount McRae Formation is pyritic and carbonaceous shales with thin interbedded BIF layers units toward the top. The formation has a 50 m thickness and represents an irregularly distributed carbonate turbidite environment (Perring *et al.*, 2020). Brockman Iron Formation is the thickest in the Hamersley Group, with 610 m. Trendall *et al.* (2004) determined the age range of the formation as 2451-2494 Ma. The formation is made of four members: the lowermost Dales Gorge BIF member, gradually overlying the Whaleback Shale member, the Joffre member BIF and Yandicoogina Shale member. It is stated that starting with the Brockman Formation, the environment is changed from a passive margin to an active margin-back arc (Howard & Martin, 2018). Of the two formations with a thickness of 225 m, the Weeli Wolli Formation is characterised by doleritic sills that have intruded into a sequence of alternating BIF and shale. The latter, Boolgeeda Iron Formation, comprises BIF with intermittent shale-rich units (Perring *et al.*, 2020).

The Turee Creek Group, predominantly composed of epiclastic sediments, stratigraphically overlies the Hamersley Group, exhibiting considerable variations in its thickness (Figure 3). The Turee Creek Group is inferred to represent a foreland basin. It is thought to have formed as a result of the initial tectonic and erosional processes between the Pilbara and the Yilgarn Cratons (Kepert, 2018).

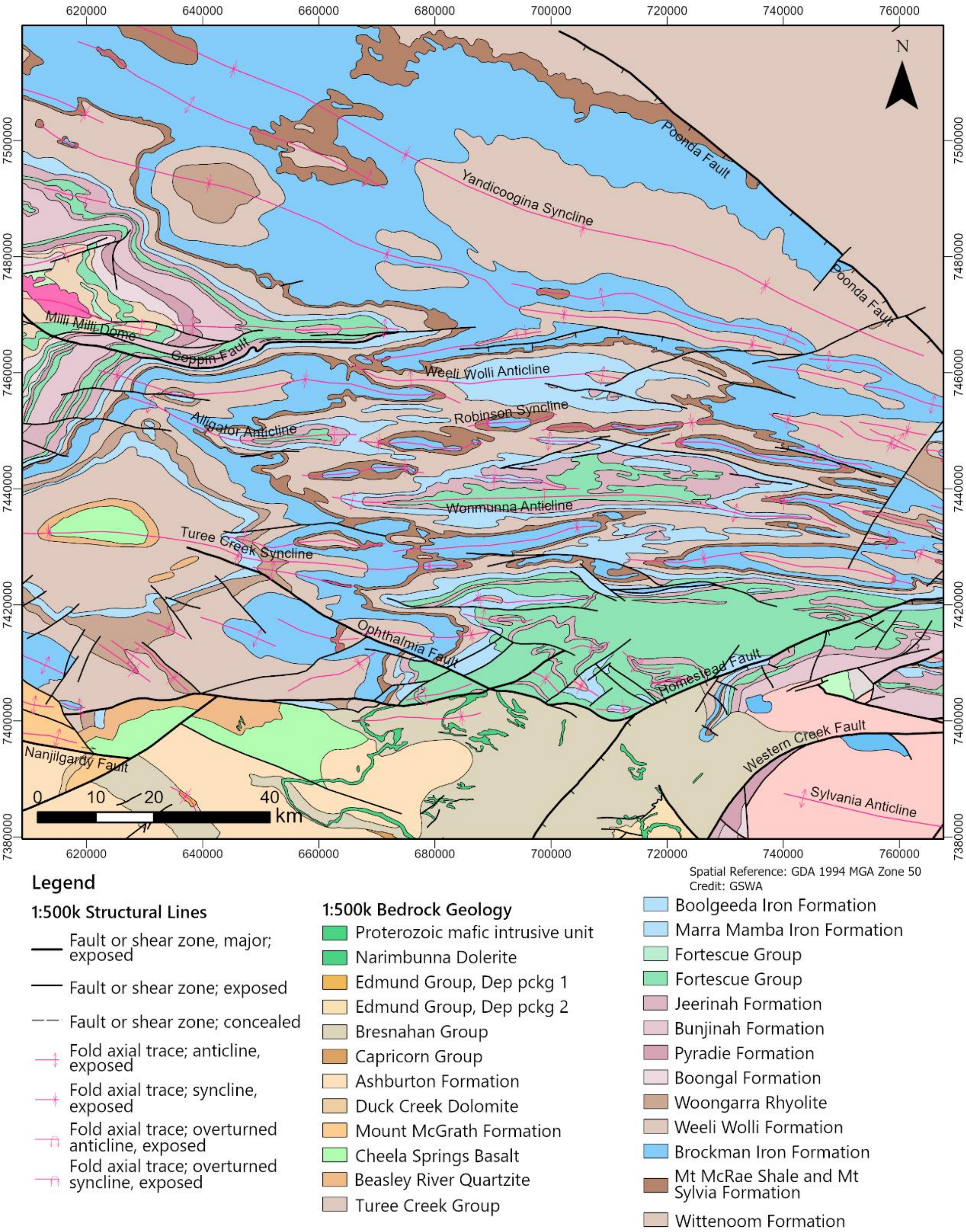

Figure 2. Geological map of the Hamersley and Fortescue Basins (GSWA, 2018).

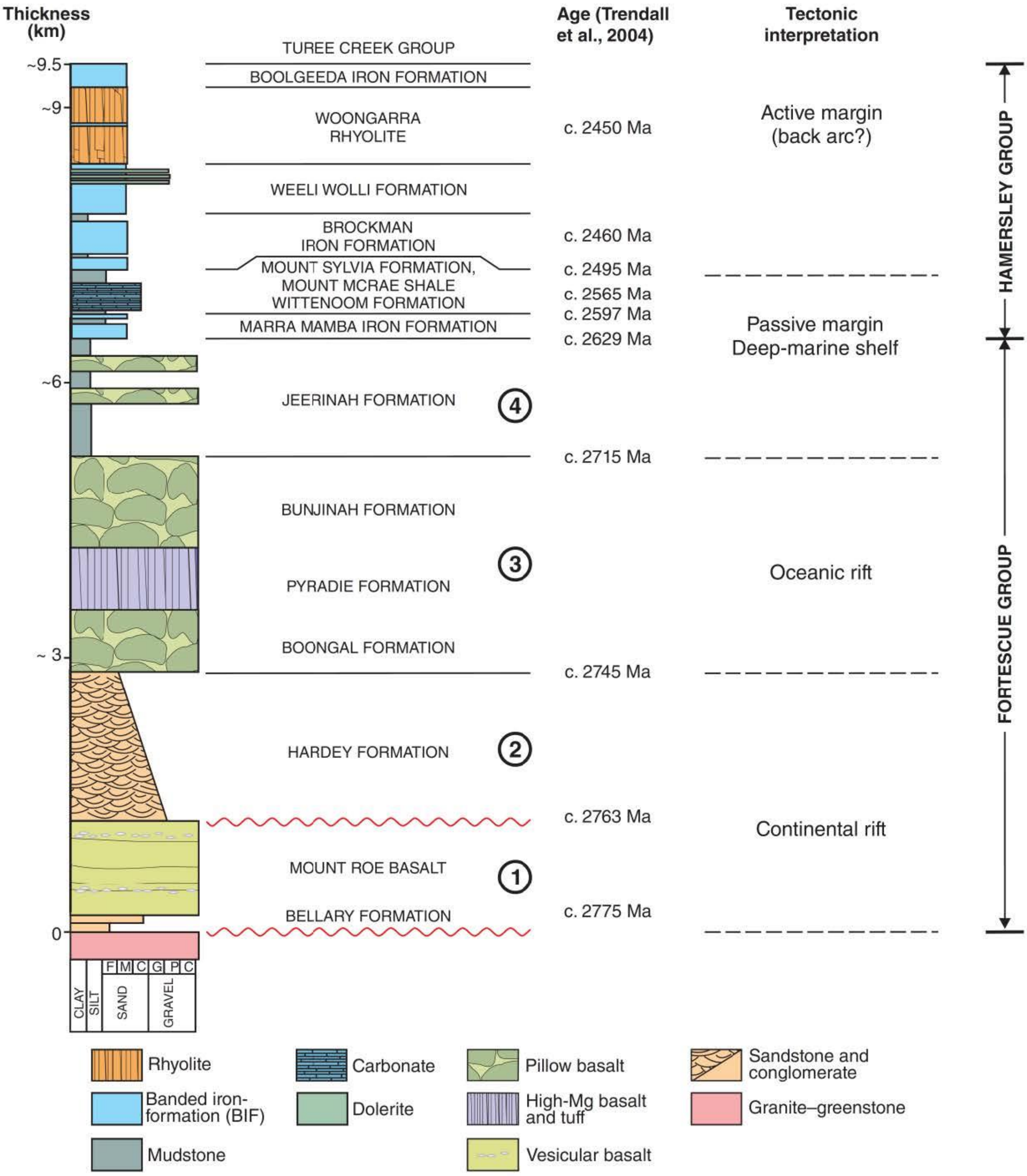

Figure 3. The stratigraphic chart of the Fortescue and Hamersley Groups in the Southern Pilbara including the study area. Tectonic interpretations from Howard and Martin (2018); ages after Trendall et al. (2004). Tectonic packages of the Fortescue Group are numbered 1-4 (Howard & Martin, 2018).

The Hamersley Basin is principally distinguished by the accumulation of chemical sediments in a deep-water environment. Turbiditic sediments, along with assorted intrusive and extrusive rocks, are found in relatively diminished quantities within the region. The sedimentary assemblage is made up of a variety of lithologies that are roughly organised according to their abundance. These lithologies include banded iron formation (BIF), hemipelagic shale, dolomite made from peri-platformal ooze, chert, pyroclastic shale and tuff, turbiditic carbonate, and turbiditic volcanics. The further north boundary of the basin represents a shallow marine to terrestrial Fortescue Group exposed to the Pilbara craton (Kepert, 2018).

4.3. Structural Geology

The Hamersley Basin is known for its complex structural geology, which reflects the intricate tectonic history of the basin. The basin exhibits overprinting folds, thrust faults, shear zones, jointing, fracturing as well as unconformities. These rocks have undergone different structural events and metamorphism.

The structural framework of the Hamersley Basin consists of various significant components. First, the basin is underlain by an ancient basement composed of granite and gneissic rocks, indicating a stable cratonic region. This basement contains a sequence of sedimentary rocks, including Hamersley Basin’s BIFs, shales, sandstones, and chert.

The basin has experienced multiple deformation events, including folding, faulting, and metamorphism. The majority of rocks within the Hamersley Basin exhibit indications of deformation in soft sediments (Ewers & Morris, 1980). The Hamersley Orogeny, which lasted from the Late Archean to the Early Proterozoic, was the most significant tectonic event. The basin was compressed and deformed during this orogeny, resulting in the construction of regional-scale folds and thrust faults. These features can be seen within the basin as anticlines, synclines, and thrust faults.

The folding in the Hamersley Basin is typically straight and gentle, generating broad, elongated anticlines and synclines. These folds are generally asymmetric and plunge gently towards the basin's centre. Thrust faults linked with the Hamersley Orogeny caused the stacking of rock units, resulting in the emplacement of younger rocks over older ones. Deformation within the province manifests as gentle in the northern sector, but progressively intensifies as one moves southward, accompanied by pronounced folding and faulting (Smith *et al.*, 1982; Lascelles, 2002). Two forms of folding impact the Mt Bruce Supergroup: the first consists of relatively gentle and open folds that extend across the majority of the surface area, while the second involves more intense deformation, typically occurring in the circumferential confines of a constrained and structurally limiting ellipse. Within the northern half of the ellipse, the initial type primarily exhibits a north-south orientation along the synclinorium ridges of the greenstone belts within the Pilbara Block. In the southern portion of the ellipse, the predominant open folds generally trend in a south-southeast direction, as clearly demonstrated by the Hamersley Range (Trendall, 1983). In the southern section, the open folds exhibit axial variations marked by the presence of en échelon offsetting of anticlines and synclines, and the dips are generally less than 40º. The folds progressively constrict as one moves southward, and in close proximity to the limiting ellipse, there is a rapid escalation in their frequency (Arndt *et al.*, 1991).

The rocks within the basin have also been influenced by metamorphic processes. The BIFs have undergone low-grade metamorphism, transforming iron oxides and silica into minerals, including hematite, magnetite, and quartz. This metamorphism is believed to have occurred during regional deformation events (Smith *et al.*, 1982; Lascelles, 2002; Shibuya *et al.*, 2010).

4.4. The 3D Geological Modelling Software: Loop

The 3D models created in this study were generated by Loop software, which is being developed as part of the MinEx CRC Research Project.

The software aims to address the challenge of geological data sparsity and the need for more accurate geologic modelling in complex terrains with limited data. The main goal of Loop is to develop geologically reasonable starting framework models in regions with sparse data and complicated geology. It accomplishes this by automating data processing and using implicit geological modelling engines that satisfy topological and kinematic constraints.

Loop open-source software, written in Python language, offers a wide range of possibilities for 3D geological modelling and structural analysis. The software comprises Python packages, principally map2loop (Jessell *et al.*, 2021) and LoopStructural (Grose, Ailleres, Laurent, Caumon, *et al.*, 2021; Grose, Ailleres, Laurent, & Jessell, 2021). One of the key advantages of Loop is its accessibility, as all of its sources and examples can be found on GitHub (http://www.github.com/Loop3D), allowing for collaboration, customization, and continuous improvement within the geoscience community.

4.4.1. Map2loop

Map2loop is a Python software package designed to automate the process of dissecting geological maps into a collection of enhanced outputs. These outputs can be employed directly in 3D geological modeling systems or as analytical resources for 2D investigations. The software assesses the spatial and temporal relationships within geological maps, incorporating data from sources beyond the maps themselves, such as stratigraphy databases. The resulting outputs are categorized into three types: positional, gradient, and topological, and they are derived from combinations of input data pertaining to position, gradient, and topology (Jessell *et al.*, 2021).

The advantage of map2loop is that it provides a reproducible and automated solution for preparing input data for 3D geology modelling. Thanks to the package, all data required for modelling can be produced in a couple of minutes. Therefore, it will save time for geologists who are exploring or researching and make their work much more manageable. By automating the deconstruction of geological maps, map2loop improves the accuracy and efficiency of 3D geology modelling, which has potential applications in mineral exploration, groundwater management, and natural hazard assessment (Jessell *et al.*, 2021).

4.4.2. LoopStructural

LoopStructural is one of the modules of Loop and is responsible for 3D modelling operations. It is also an open-source Python library for implicit 3D geological modelling. LoopStructural is a 3D probabilistic geological and geophysical modelling platform that uses implicit surfaces to depict various stratigraphic groups, faults, folds, and unconformities. The module is powered by the core scientific Python libraries pandas, numpy, and scipy. It comes with a generic API that allows it to operate with 3D modelling applications. This provided API allows different interpolation algorithms to be mixed and matched within a geological model, which means that different geological objects can be modelled using different algorithms, allowing for more flexibility in the modelling process (Grose, Ailleres, Laurent, & Jessell, 2021).

LoopStructural adopts a time-aware modelling approach that permits complicated overprinting relationships to be integrated into the implicit geological models based on the relative timing between different geological phenomena (Grose, Ailleres, Laurent, Caumon, *et al.*, 2021). The software design of LoopStructural is object-oriented, enabling easy development and enhancement of 3D modelling algorithms (Grose, Ailleres, Laurent, & Jessell, 2021).

5. MATERIAL AND METHOD

All coding operations of this study were compiled using the Jupyter Notebook interpreter/IDE within the Conda environment (https://www.anaconda.com) using Python (v3.9). Popular Python packages, pandas, numpy, and matplotlib, were used for pre-processing data, and the LoopStructural package was used for 3D model generation.

On the other hand, GIS operations were performed through QGIS. The geological map, the faults and the bedding measurements in the study area were provided by BHP Group Ltd in ESRI shapefile format. In this study, all geographical data and coordinates are referenced using the GDA 1994 MGA zone 50 coordinate system.

5.1. Map Data

The materials utilized in this research consist of map data presented in the shapefile format, encompassing crucial geological information such as geological formations, faults, and bedding measurements (Figure 4). These data were processed using QGIS, which facilitated efficient mapping operations. The geological formations data provided insights into the spatial distribution and extent of different geological units in the study area. The fault data enabled the representation of structural complexities, while the bedding measurements assisted in accurately orienting the geological formations. Leveraging the capabilities of QGIS, these shapefile data served as primary inputs for the geological 3D modelling process, enabling the creation of a comprehensive and accurate representation of the subsurface geology.

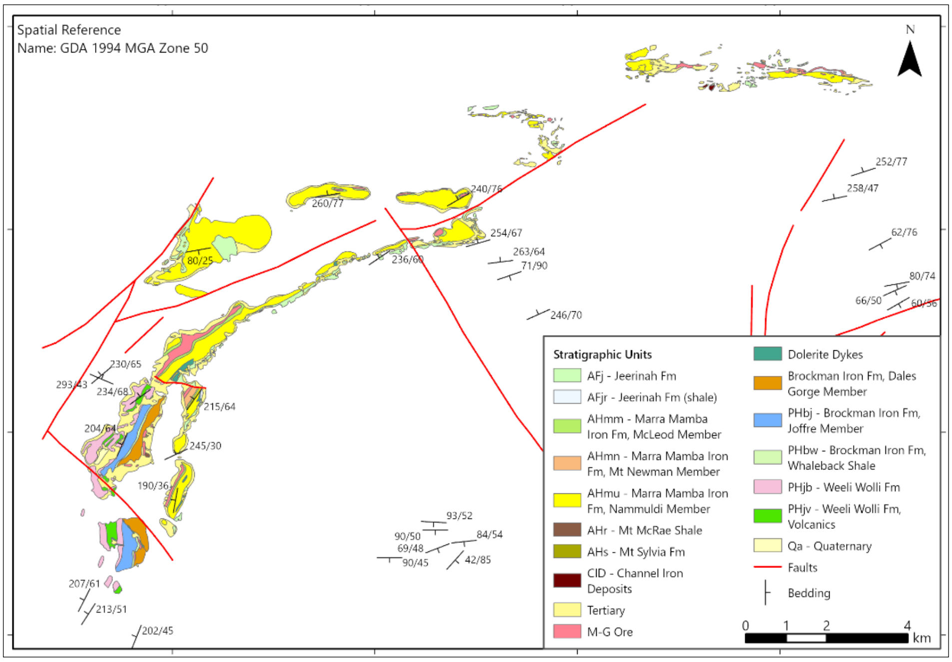

Figure 4. BHP's geological map of the study area was used for modelling purposes.

The stratigraphic data were visualised utilizing the OpenLog QGIS plugin, whose experimental version was released during the study. OpenLog is a free plugin that proved invaluable for analysing detailed stratigraphic logs. Using the provided drillhole data, OpenLog allowed for validation of the collar polarity codes, ensuring an accurate representation of the drillhole orientations in the stratigraphic columns.

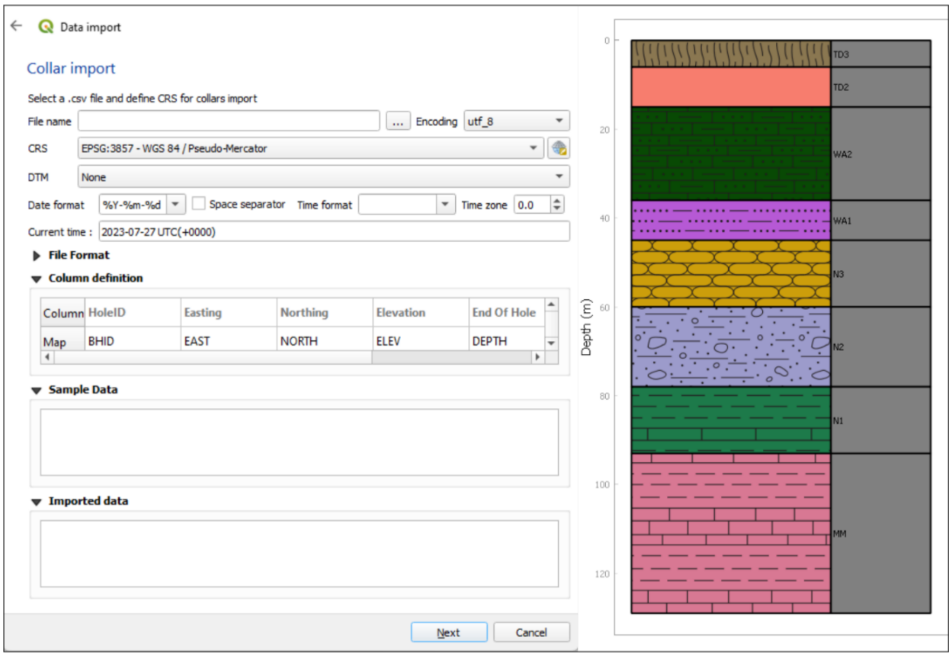

Figure 5. Data importation user interface of the OpenLog plugin (left) and a sample log (right) generated by OpenLog.

5.2. Primary Data

The primary material of this study is the drillhole data consisting of 115 collars provided by BHP Group Ltd. The drillhole data comprises three tables in CSV format: collars, surveys, and stratigraphy tables.

The collars table contains BHID as collar names, EAST, NORTH, ELEV, and DEPTH columns and additional metadata information, such as the coordinate system. EAST and NORTH columns specify the geographical location, and the ELEV column stands for the altitude of the drillhole area, whereas the DEPTH column represents the maximum drillhole depth (Figure 6). The collar table has 115 rows corresponding to the 115 drillholes.

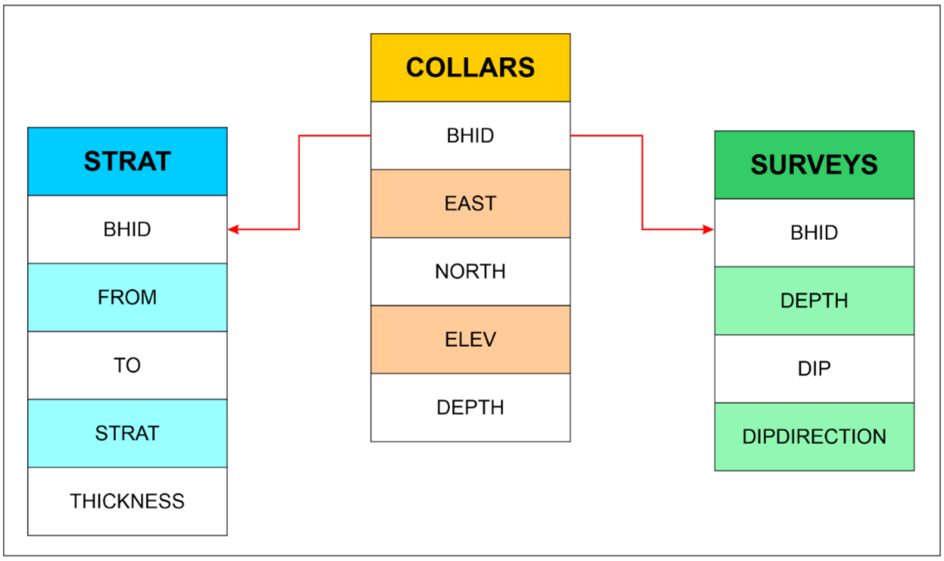

Figure 6. Relationship chart of the stratigraphy and surveys tables with the collars table.

The Surveys table contains the collar name and dip and dip direction measurements of each drillhole, including the measurement depths. The link between the Collars and Surveys tables is based on the unique collar name (BHID) column (Figure 6). The orientation measurement of the drillholes was performed at intervals of 5 meters starting from the surface for each drillhole, and the file contains 1418 measurements. Most holes were vertical.

The Stratigraphy table comprises collar name (BHID), FROM, TO and STRAT columns. This table provides mainly stratigraphic log data. The STRAT column provides the stratigraphic unit drillhole code data (Table 1), while the FROM and TO columns give the range of the relevant stratigraphic unit (Figure 6). The table contains

398 rows for 115 collars. The Surveys and Stratigraphy tables are associated with the Collars table via the BHID common column (Figure 6).

The stratigraphic codes used in the geological maps prepared by BHP and GSWA differ from the drillhole stratigraphic codes. The conversion table of the map and probe codes is given in Table 1.

Table 1. Conversion chart of map codes and drilling codes.

Stratigraphic Order	Drillhole Code	Map Codes	Formation Name and Descriptions
0	F	H2	M-G ore
0	K	K	Dykes/Sills
0	QA	Qa	Alluvium
0	TD1	ROD	Tertiary Detritals
0	TD2	CID	Channel Iron Deposit
0	TD3	Cz	Colluvium
0	PDD	Pd	Dolerite dyke
1	WW	PHj	Weeli Wolli BIF
2	Y	Phby	Brockman Iron Formation, Yandicoogina Shale Member
3	J6	PHbj	Brockman Iron Formation, Joffre Member
4	J3J5	Phbj	Brockman Iron formation, Joffre Member - differentiated
5	J2	PHbj	Brockman Iron Formation, Joffre Member
6	J1	PHbj	Brockman Iron Formation, Joffre Member
7	W	PHbw	Brockman Iron Formation, Whaleback Shale
8	D4	PHbd	Brockman Iron Formation, Dales Gorge Member
9	D3	PHbd	Brockman Iron Formation, Dales Gorge Member
10	D2	PHbd	Brockman Iron Formation, Dales Gorge Member
11	D1	Ahrc	Colonial chert Member
12	R	SHr	Mount McRae Shale
13	S	Ahs	Mount Silvia Formation
14	O	Ahd/WD	Wittenoom Formation - Undifferentiated
15	OB	Ahdp/WP	Wittenoom Formation, Paraburdoo Member
16	WA2	A2	Wittenoom Formation, West Angela Member
17	WA1	A1	Wittenoom Formation, West Angela Member
18	N3	N3	Marra Mamba Iron Formation, Mount Newman Member
19	N2	N2	Marra Mamba Iron Formation, Mount Newman Member
20	N1	N1	Marra Mamba Iron Formation, Mount Newman Member
21	MM	Ahmm	Marra Mamba Iron Formation, MacLeod Member

22	MU	Ahmu	Marra Mamba Iron Formation, Nammuldi Member
23	NX	Afjr	Jeerinah Formation - differentiated
24	XX	Afjr	Jeerinah Formation

Another primary input data is the stratigraphic order table. This table contains the age order of stratigraphic units from top to bottom for use in modelling. The table consists of the order number of the unit, unit code, and average thickness of the stratigraphic units. Since Tertiary and Quaternary units in the study area were not included in the modelling, 0 was assigned for their order number.

5.3. Intermediate Data (Pre-processing Data)

In the process of creating a comprehensive 3D geological model, it is essential to handle and process the intermediate data. LoopStructural needs certain subsurface information as well as the Collars, Surveys and Stratigraphy tables to produce 3D models. These subsurface layers were derived from the collars and stratigraphy tables through the Python packages mentioned above and exported in CSV format. The fault data was derived from shapefiles and integrated into the modelling process by map2loop. The intermediate data play a crucial role in defining the geological framework and provide the necessary information to represent subsurface layers and structures accurately. The key intermediate data of the study is given in the section below.

5.3.1. Collar Polarity

The collar polarity is crucial data that provides valuable information about the stratigraphic order in the subsurface. It refers to the arrangement of stratigraphic layers encountered by each drillhole in the subsurface. The collar polarity data table has five primary columns: BHID, EAST, NORTH, ELEV, and POLARITY. The BHID column is a unique identifier for each drillhole. The EAST and NORTH columns refer to the geographical location of the collars, whereas the ELEV refers to the altitude of the collar in meters. POLARITY is the column where numerical values are stored, showing the polarity type of each collar.

Assuming that the top-down sequence is regular in the stratigraphic order given in Figure 3, the sequence in each well was determined to be regular, inverted, irregular, and unknown (Figure 7). The units were coded as a reference by incremental numbers from young to old using the stratigraphic chart shown in Figure 3 and saved into a CSV file called *strat_order* (Table 1).

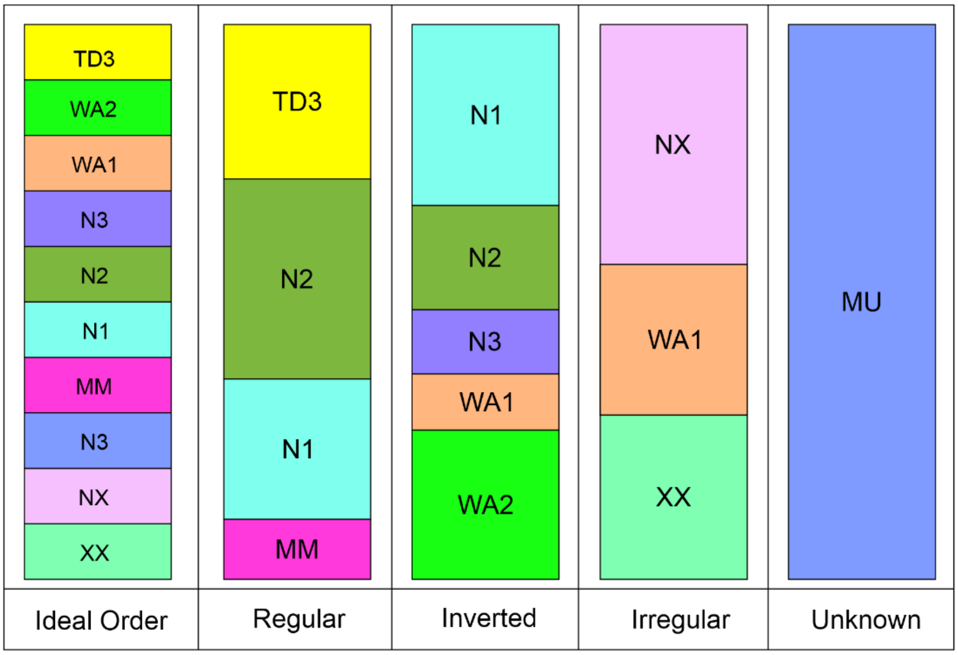

Figure 7. Schematic sample representation of regular, inverted, irregular and unknown polarity groups according to the ideal stratigraphic order.

A Python code was developed to determine the polarity of drillhole logs through a comparison with the stratigraphic order table. This process involves comparing the stratigraphic order of each drillhole log with the reference table. If a drillhole log exhibits the same stratigraphic order as the reference table, it is classified as "regular", and a polarity value of 1 is assigned to its POLARITY column. Conversely, if the stratigraphic order of a drillhole appears inverted, the log is labelled as "inverted" and assigned a polarity value of 0. In cases where the stratigraphic order is irregular, the log receives a designation of "irregular" and is given a polarity value of -1. Finally, if the stratigraphic order cannot be definitively determined, the log is categorized as "unknown" and assigned a polarity value of -2. This approach provides a systematic means of assessing and categorizing the polarity of drillhole logs based on their stratigraphic context (Figure 8).

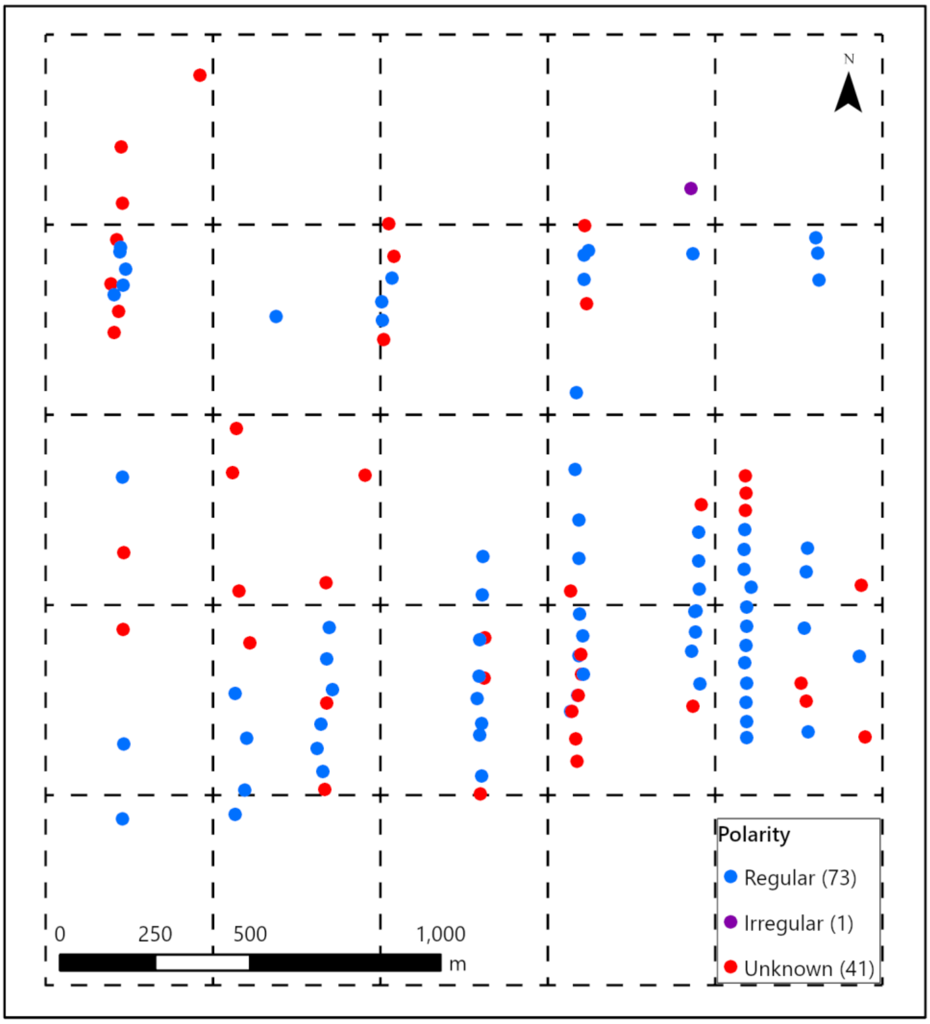

Figure 8. Spatial distribution of the collars and different polarity types. The numbers show the counts.

In other words, if the stratigraphic order numbers of the units encountered in a drillhole are in ascending order, the polarity of that drillhole is "regular"; if it is in descending order, it is inverted, and if it is in mixed order, it is irregular. If there is only one unit in a drillhole, its polarity is unknown since it is not comparable. The collar polarity table has 115 rows, including 73 regular, 1 irregular, and 41 unknown polarities (Figure 8).

5.3.2. Contact Locations

The contact locations table is another pre-processed data derived from the primary and intermediate datasets. A Python code was developed to determine the contact locations of the subsurface stratigraphic units within the study area. This code makes use of a multidimensional method that includes stratigraphic logs, stratigraphic orders and drillhole locations. Considering the polarities of the drillholes, stratigraphic units were correlated using log data and contact locations were determined by numerical calculations theoretically. For this calculation, EAST and NORTH coordinates of the drillhole, elevation, stratigraphic unit code, FROM and TO data of the stratigraphic units for thickness calculation and previously determined polarity data were used. The table consists of X and Y columns for coordinates, Z for elevations, Formation for stratigraphic unit codes and Source for reference drillhole names.

5.3.3. Dip and Dip Direction

In this study, two different dip and dip direction datasets were used. The first data was based on surface measurements derived from the geological maps published by GSWA and the dataset provided by BHP Group Ltd (Figure 4). The other dataset was obtained through calculations using drillhole log data, representing the subsurface layers' orientation. The dip and dip direction data of subsurface layers are crucial components derived from the primary data. These data provide essential insights into the orientation and structural characteristics of geological layers, enabling the accurate representation of subsurface structures.

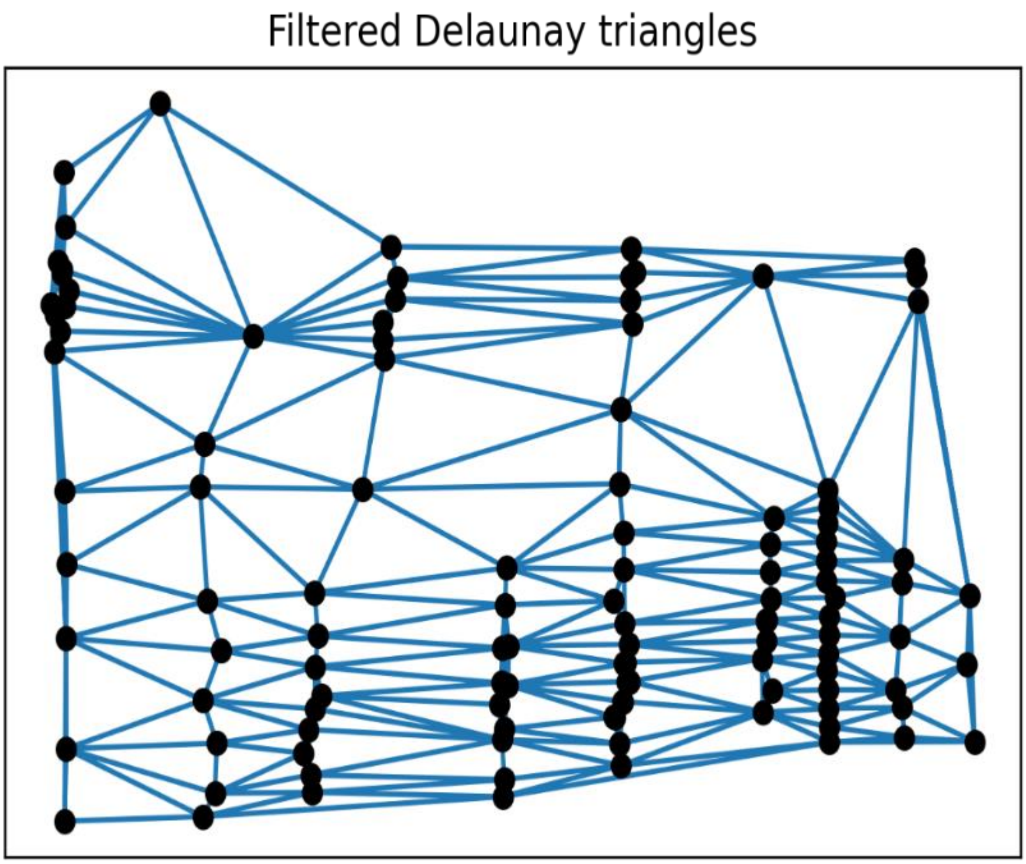

Figure 9. Delaunay Triangles from the study area. Black dots represent the collars.

The dip/dip direction measurements were derived using the 2D Delaunay Triangulation method on drillhole collar locations. This method includes forming non-overlapping triangles between the drillholes with the same polarity (regular, inverted) and estimating dip and dip direction at the centroid of resulting triangular surfaces. This methodology was applied where the inter-collar distance does not exceed 1000 meters (Figure 9). Python codes were employed to perform the intricate calculations required for obtaining the dip and dip direction measurements. The subsurface orientation of each layer was calculated by matching the stratigraphic units at the corners of each triangle with the same stratigraphic unit located at the other corners. Thus, 105 rows of dip/dip direction data were derived.

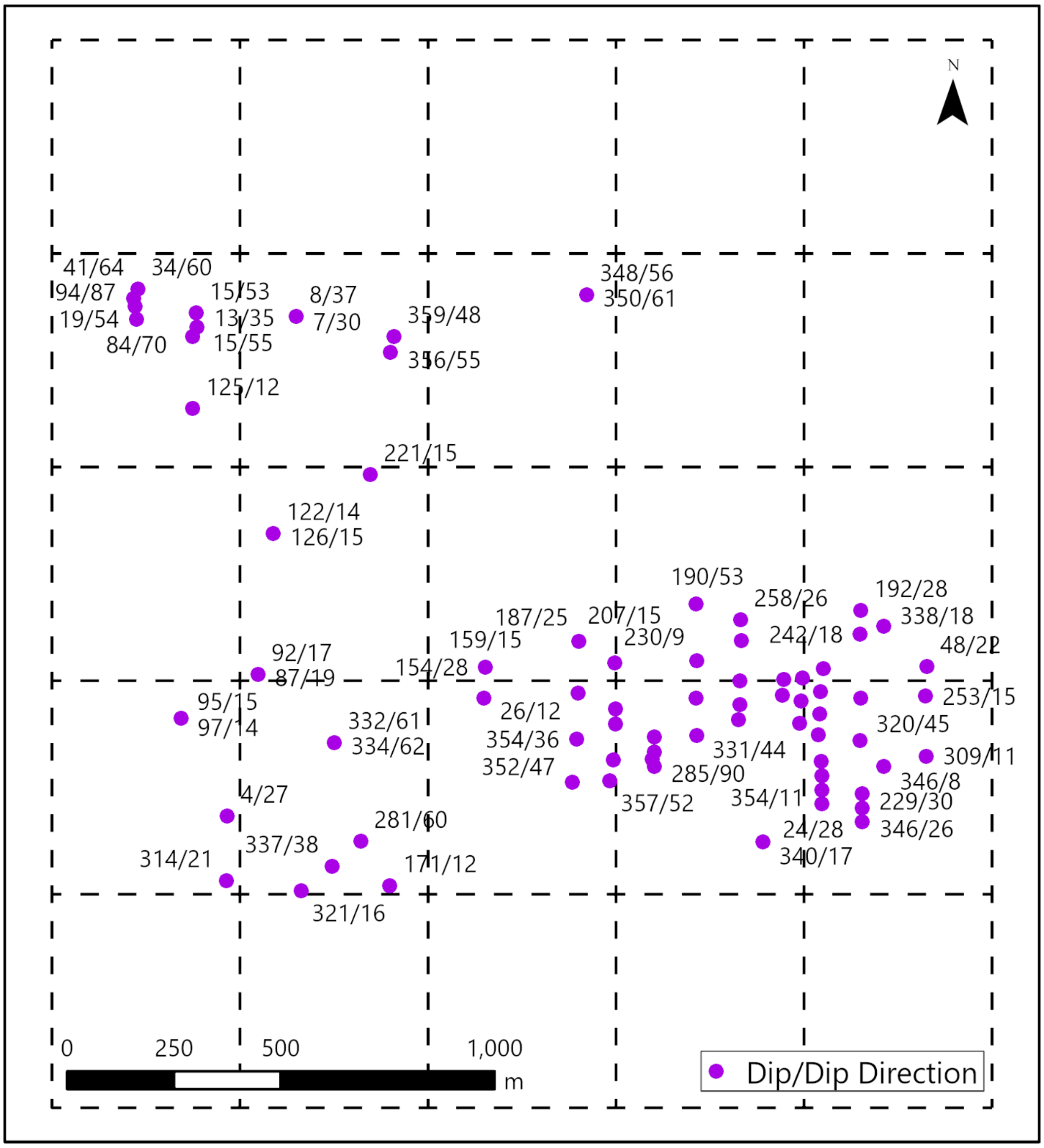

Figure 10. Dip/Dip direction measurement results through Delaunay Triangulation.

5.4. Modelling Method

5.4.1. Sampling

The sampling operations were conducted through QGIS, Postgresql and Python. To create the data sets, two distinct sampling methods were employed. In the first approach, sampling was carried out by dividing the study area into identical sub-areas (grids), ensuring well-distributed coverage. The study area was discretised into the grid of a 5x5 (440m x 500m), ensuring well-distributed coverage through QGIS tools 'Create Grid' tool, then the number collars were counted by the 'Count points in polygon' tool (Figure 11). Subsequently, the number of drillhole selected for each percentile subgroup was calculated via QGIS and assigned to the relevant column. In the second approach, the sampling process was carried out randomly and equiprobable, irrespective of location (Figure 12). In both methods, ten distinct samples were made. Thus, nine additional models with reduced data were generated to test the model and evaluate the effect of decreasing input data density on the resulting model. This strategy attempted to capture variations across the entire study area in an organised manner, ensuring a well-distributed sample for modelling. The original data includes 115 collar data. The original data set was reduced by 10% successively, from 100% to 10%, by the respective sampling method. In other words, 10% of the original data were removed from each subgroup for both approaches at each iteration.

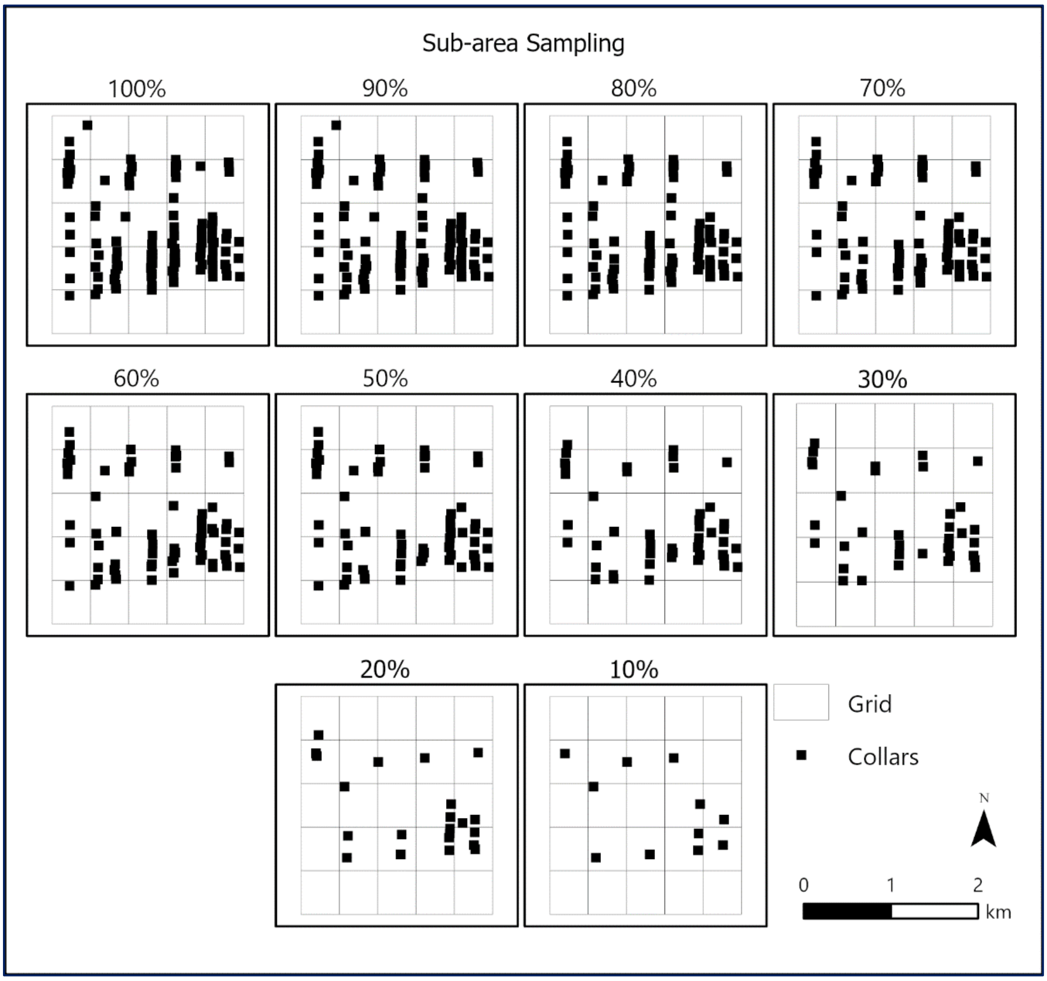

Figure 11. Distribution of collars selected by sub-area sampling method.

In the next step, GIS data was imported to Postgresql for selection. The purpose of using Postgresql was that its querying capabilities were more advanced than QGIS, and it was compatible with Python. Each sampling step resulted in the production of a specific collar (subgroup) file. Thus, each sample group was determined by selecting them in decreasing proportion as 10%. If the resulting subgroup size is not an integer, it would be rounded to the nearest integer for practicality. Values with a fractional part equal to or less than 0.5 have been rounded to the previous integer, while those with a fractional part greater than or equal to 0.5 have been rounded to the next integer. In instances where a sole collar is present within a given grid, it has been ascribed to the subgroup encompassing values equal to or surpassing 50% while being excluded from the subgroup associated with values falling below this threshold.

In contrast, the second method involved global sampling data from the study area. This approach aimed to reveal the effect of sampling, regardless of spatial arrangement, on modelling. Similar to the first approach, a 10% reduction in data points was implemented; nevertheless, the selection process was conducted irrespective of location. These data sets were also created through Postgresql queries, and Python code was developed.

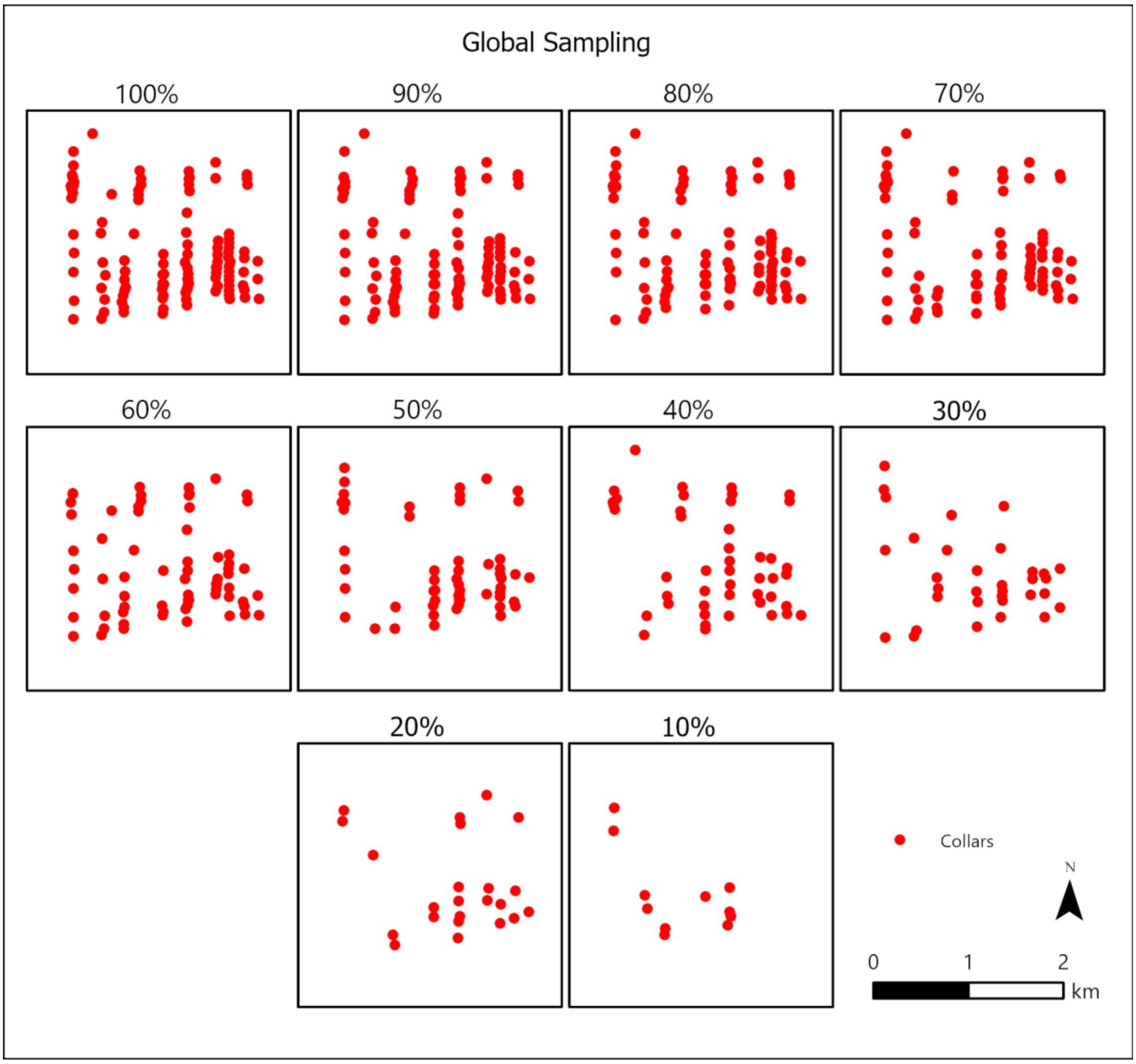

Figure 12. Distribution of collars selected by global sampling method.

Ultimately, employing both sampling strategies resulted in the development of ten separate 3D geological models for each strategy. These models were created to compare the results and evaluate the impact of various sampling procedures on the accuracy and reliability of the resulting geological models. These steps aimed to contribute to understanding the effective sampling techniques for 3D geological modelling.

5.4.2. Modelling

The modelling process includes chained data processing and transformation steps (Figure 13). These steps aim to create the appropriate input data for the LoopStructural module that is responsible for the modelling process. All processes were performed using Python, excluding sampling.

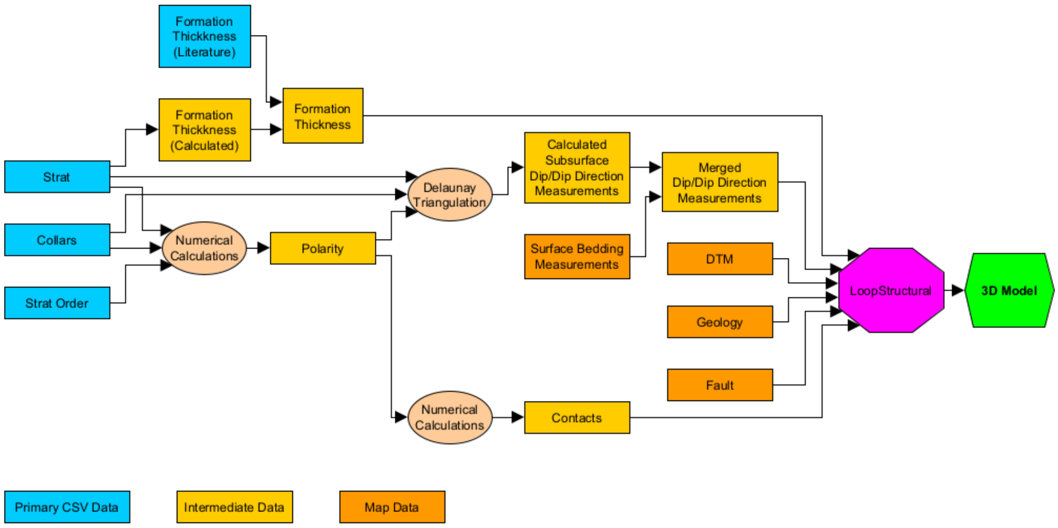

Figure 13. Modelling process flowchart.

By processing 20 groups of data sets obtained from both sampling approaches mentioned above, 20 different models were produced. Each model was produced with the relevant collar data.

6. RESULTS

6.1. Modelling Results

The effect of drilling density on modelling was revealed by comparing the geological models produced by both sampling methods with the models in their own group and in the other group (Figure 14, Figure 15). In addition, the top view of the reference model (produced by 100% of the drillhole data) and the geological maps, the 1:250000 scale Newman sheet published by GSWA and the 1:10000 scale geological map provided by BHP were compared to observe the model's reliability (Figure 16).

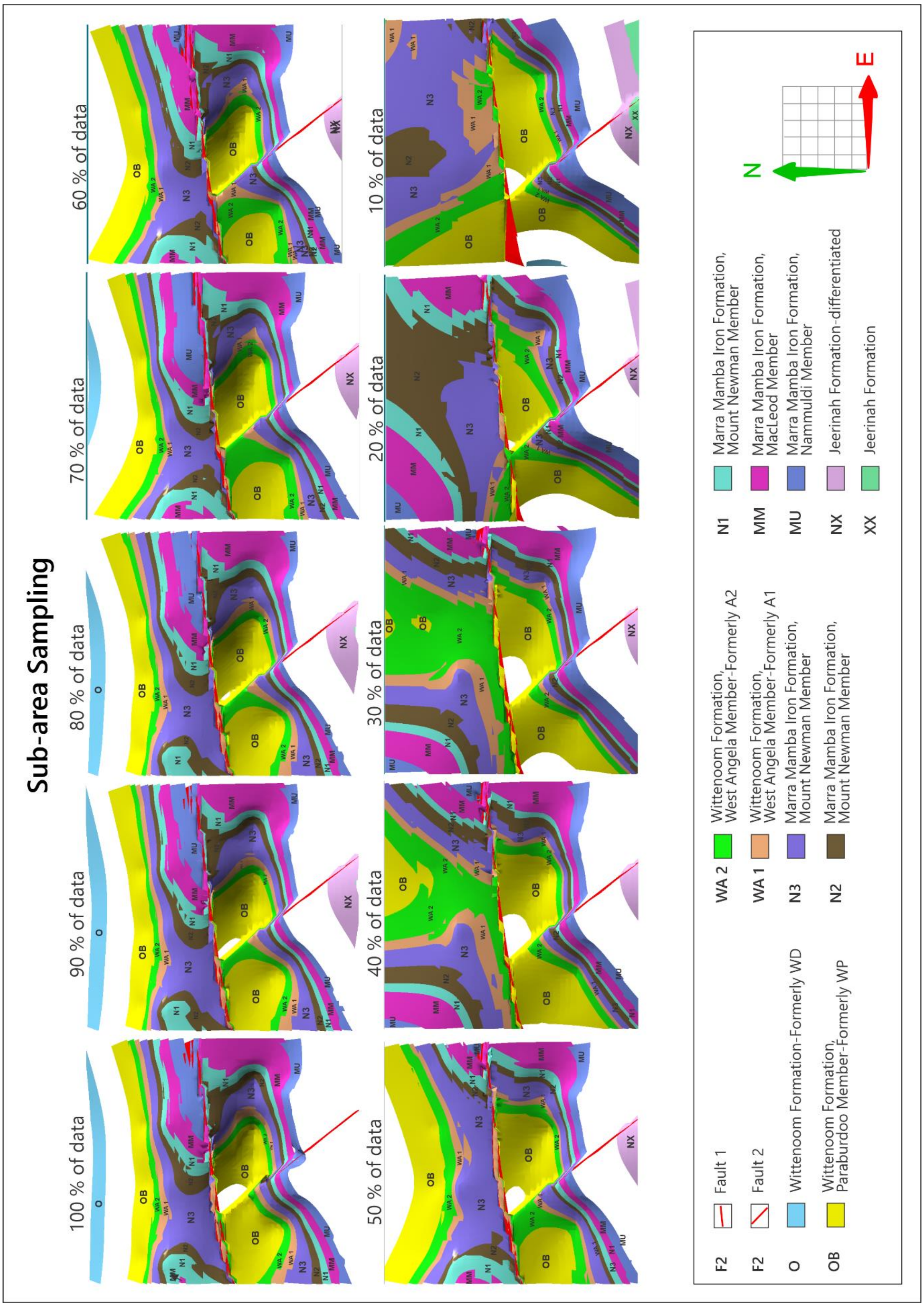

Figure 14. The models were generated with the sub-area sampling method.

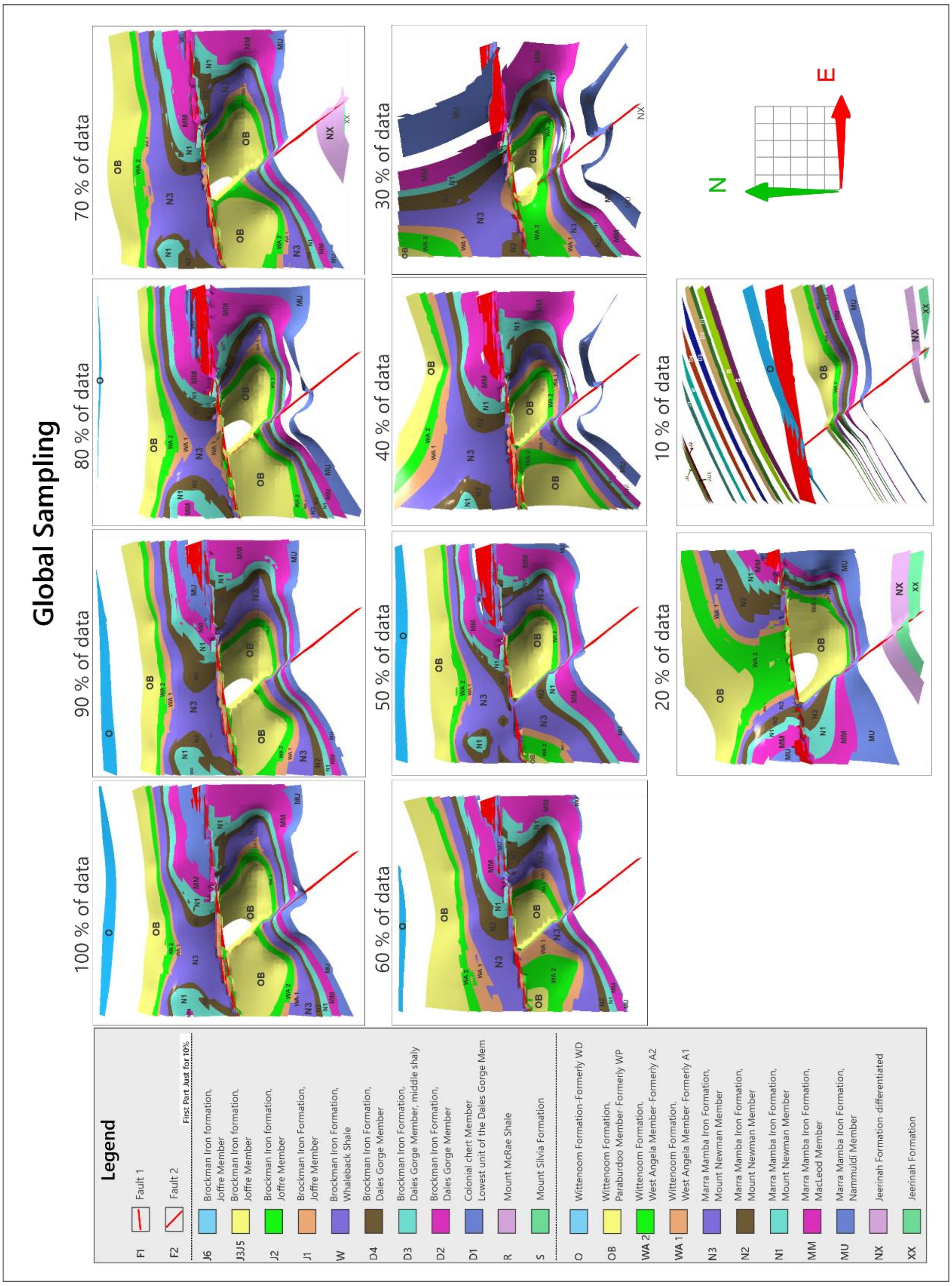

Figure 15. The models were generated with the global sampling method.

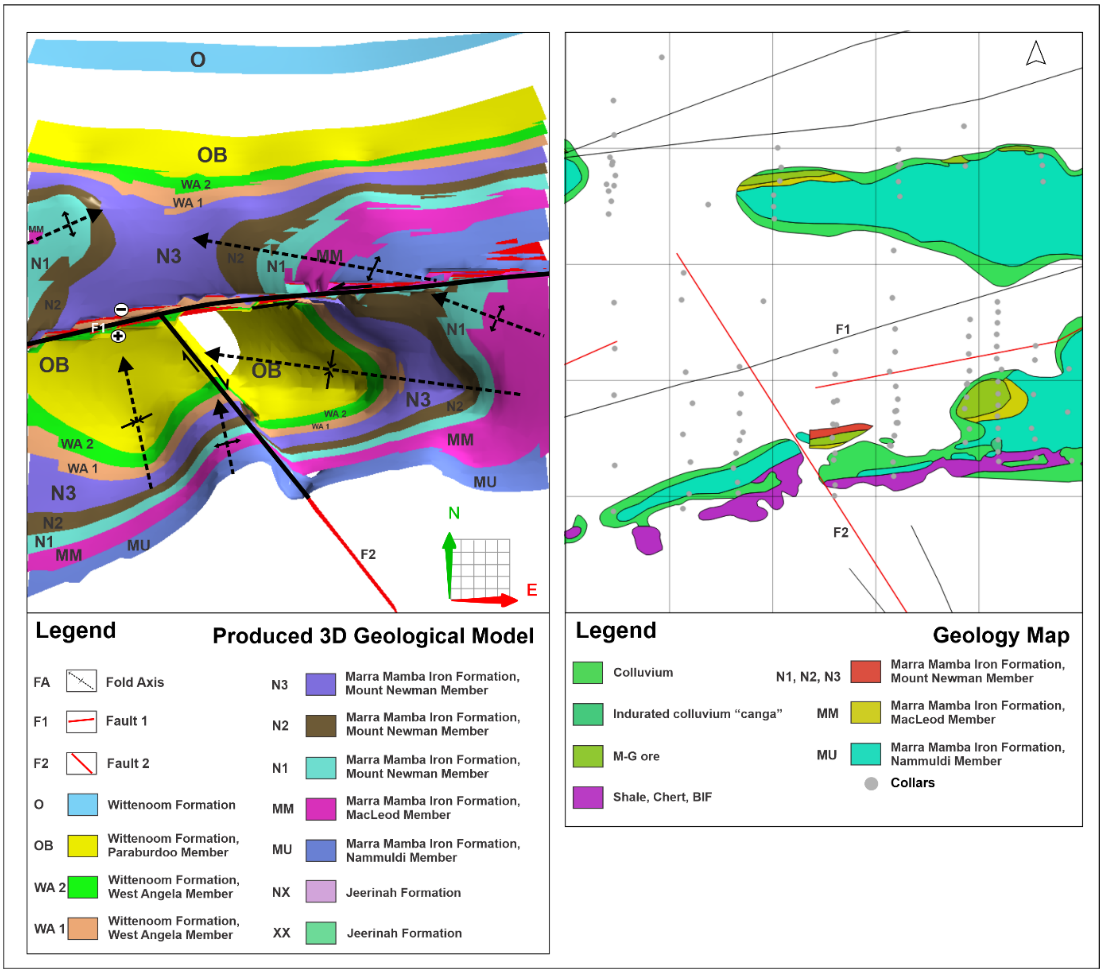

Figure 16. Comparison of the 3D model and the geology map of the study area.

First, geological interpretations were made according to the model produced, and then a comparison was presented. 3D views of the model were used to determine the types of structural elements, folds and faults in the study area (Figure 17).

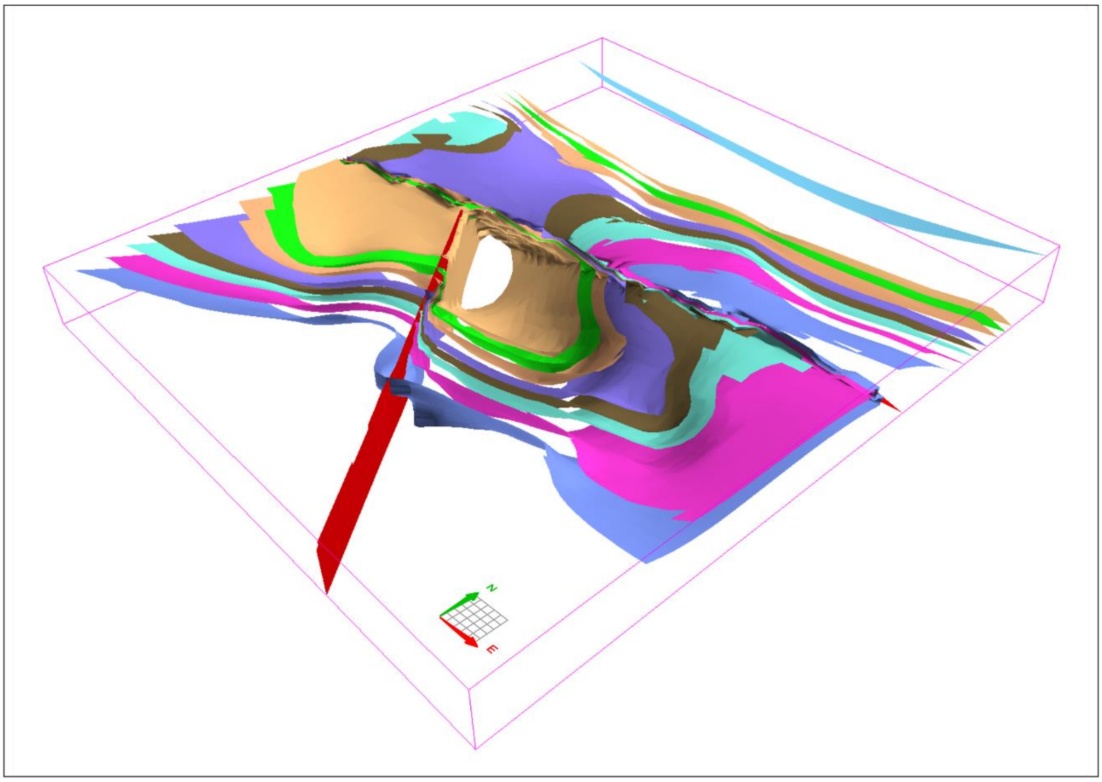

Figure 17. The reference 3D model of the study area is generated with the whole data.

For example, the bedding directions were observed towards the centre in both limbs of the syncline located southeast of the study area (Figure 17). It was also observed that the dip direction of the layers was towards the northwest. It is understood from the model that this fold is part of a fold belt that continues in the form of anticline-syncline throughout the study area. In addition, the small-scale folds observed around the NW-SE trending right-lateral strike-slip fault, with approximately 90 degrees of the plane, are considered faulting-related folds. It is observed that the anticline, located in the southeast of the map and in which MM, one of the oldest units, is located, was cut by the east-west trending fault in the centre and slipped (Figure 16).

Regarding the E-W fault, it has been observed that the units have horizontal and vertical displacement on both sides of this fault (Figure 16). Another observation of this fault is that the northern hanging wall, which dips approximately 60-70 degrees towards the North, is displaced relatively compared to the southern footwall block. In summary, this fault has movement along the fault line (left strike-slip) and parallel to its plane (a reverse fault). Thus, it is interpreted that the fault is formed under a transpressional regime.

As is demonstrated by the model, the region was deposited in regular chronological order, and all units were deposited in order of age. Subsequently, all units were folded and faulted by the two faults.

When the final model is compared with the existing geological map, the first striking detail is that the geology map based on detailed surface observations has fewer units than the units obtained in the model. Unexpectedly, about 11 units were distinguished in the model. The faults shown in red on the geological map were obtained from the Newman sheet (GSWA, 1990), and the black ones were obtained from GSWA (2006). Due to the inconsistency in the maps, there is uncertainty as to which fault dominates the other. For example, a geologically problematic intersection exists between the NW-SE trending fault coming to the centre from the south and the NE-SW trending fault relationship. However, this relationship is presented quite neatly in the model. In general, the East-West (NE-SW) trending fault and NW-SE trending fault appear compatible with each other in the model and the geological map.

6.2. Robustness of the Model

In order to test the robustness of the reference model (the model generated with 100% of the data) and to understand the effect of reduced input data density on this model, ten models were developed for each group (2 groups in total, sub-area and global) with gradually reduced input data. These groups emerged out of curiosity about the answers to two questions. The first question was how model changes would emerge when the study area is divided into 5x5 grids and a sub-area sampling is made with 10% decreases (Figure 14). The second question was how the model changes if the same number of collars are selected by the global sampling method without creating a grid (Figure 15). By understanding the answers to these two questions, it is thought that the relationship between the 3D geological model and the effect of drillhole density, which is the main subject of this study, will be better understood.

As elucidated in the preceding sections, the original dataset encompasses 115 drillholes, comprising calculated 106 dip/dip direction data (Figure 10) and 154 contact location data.

According to all these data, the results of the robustness are striking. As the number of data decreases, changes can be expected to occur, including layers that appear or disappear in each subgroup (Figure 14, Figure 15). While these changes were observed gradually in the sub-area sampling group, they are difficult to predict in the global sampling group. The reason for this is thought to be directly related to the sampling method and the distribution of the drillholes in the study area. In the global sampling method, selecting the points in each subgroup from a random location causes inconsistency in the models. The conspicuous parallels merit attention, as they demonstrate that the level of resemblance to the ultimate model, constructed using the entire dataset, persists even within the model generated using only 40% of the data in the sub-area sampling group.

In contrast, this resemblance reaches a rate of 60% within the group generated through global sampling. Despite the data reduction in the first group, it is clear that a good match was achieved (Figure 14). The reason for this is undoubtedly the conscious choice to represent the whole area and the units. Even in the southern part of the models, there is good consistency up to the model created with 10% of the data. With diminishing data, the coherence diminishes more rapidly in the northern segment of the model compared to the southern portion. The reason for this is that, as shown in the map showing the distribution of drillholes in the study area, the density of the drillhole in the North is much less and less dense than in the south (Figure 8). As a result, the reduction of the number of drillhole can be interpreted as affecting the south of the model less than the North.

As a result, as can be seen from the results of both sampling methods, the increase in the number of data increases the quality and robustness of the model. However, it should be considered that data quality is as important as the quantity of data. This is evident from the fact that the similarity with the most data-rich models is maintained up to 40% in the sub-area sampling group while only up to 70% in the global sampling group.

7. DISCUSSION

Although the data obtained from the field is used in the geological modelling processes, assumptions are also included in the process at some stages. The models were produced using the stratigraphic units in the study area in accordance with the stratigraphic order and unit thicknesses from stratigraphic logs. Although the unit thicknesses were calculated using drillhole logs, unit thicknesses compiled from the literature were used in areas where the number of drillholes is rare. Considering that the unit thicknesses vary within the basin, it is clear that the unit thicknesses compiled from previous studies cannot go beyond being a reference. Therefore, it is possible to observe an inconsistency in unit thicknesses depending on the drilling density in the produced models.

On the other hand, within the context of the sub-area sampling method, the study area undergoes a partitioning process into discrete grids. The data selected from each grid cell was selected in a certain number but randomly, although the final subgroups were assumed to represent the entire study area. Reducing the grid size can be considered to improve this technique to achieve a more uniform distribution in the sampling process.

The dip and dip direction data obtained through this process hold significant importance in the construction of a comprehensive and accurate 3D geological model. By incorporating this information into the model, various geological structures, such as tilted layers, folds, and faults, could be realistically represented. Additionally, these measurements act as a means to validate geological interpretations, ensuring the model's reliability and consistency with actual geological observations.

Another outcome of the study is the geological features of the study area. The investigation has revealed the intersection of two faults within the study area, indicating the presence of a shear zone. Furthermore, the slip plane of the NE-SW trending fault and the displacement of the layers were modelled. However, the reason why NE-SW trending faults observed in the North of the geological map are not observed in the 3D geological model may be related to the lack of sufficient drilling data in that region (Figure 11, Figure 16, Figure 17). As a result, the resolution and accuracy of the model are directly related to the drilling density. Observing the continuity of the units underground becomes possible as the data density increases. In addition, the ability to reveal structural elements such as faults and folds is directly related to the quantity of data.

8. CONCLUSION

This research examined the effect of drillhole density on the quality of 3D models. Using the stratigraphic log data of a total of 115 drillholes, the effect of the gradually reduced number of drillholes on model production was evaluated. This process started with the phase of dividing the sample group into two groups using sub-area and global sampling methods.

This research shows that the model quality is maintained when keeping at least 40% of the drillhole data in the sample group formed by sub-area sampling. This finding supports that sub-area sampling allows models to be formed with higher quality and that model performance increases steadily with increasing sample size.

In the second sample group, which was created by the global sampling method, it was observed that the fluctuations in the model quality were less because the selected samples did not represent the study area. It has been observed that the quality of the models produced with less than 70% of data has decreased. This shows that the global sampling method cannot provide a stable increase in model performance and may cause quality loss at low sample sizes.

This research shows that the sub-area sampling method should be preferred to obtain a relatively higher quality model with less material. Sub-area sampling can increase the probability of obtaining more stable results than global sampling. However, it was observed that the increase in sample size had a positive effect on model quality. Therefore, it may be possible to obtain more robust results using larger sample groups in future studies.

In conclusion, this study examines the effect of drillhole density on 3D model quality. It shows that sub-area sampling provides more stable results and that increasing sample size can improve overall model quality. These findings highlight the importance of sample selection and data density in geological modelling studies.

REFERENCES

Arndt, N. T., Nelson, D. R., Compston, W., Trendall, A. F., & Thorne, A. M. (1991). The age of the Fortescue Group, Hamersley Basin, Western Australia, from ion microprobe zircon U-Pb results. *Australian Journal of Earth Sciences*, *38*(3), 261-281. https://doi.org/10.1080/08120099108727971

Blockley, J. (1975). Hamersley Basin mineralisation. *Economic Geology of Australia and Papua New Guinea*, *1*, 413-415.

Blockley, J. (1979). A contribution to the stratigraphy of the Marra Mamba Iron Formation. *Geological Survey of Western Australia, Annual Report*, *1978*, 71-73.

Ewers, W. E., & Morris, R. C. (1980). *Chemical and mineralogical data from the uppermost section of the upper BIF member of the Marra Mamba Iron Formation*. CSIRO.

Finucane, K. J. (1939). *The blue asbestos deposits of the Hamersley Ranges, Western Australia* (Aerial Geol. Geophys. Survey North Australia Rept., Issue.

Forman, F. (1937). Notes on a reconnaissance of the Gascoyne, Ashburton and Fortescue Districts: West. Australia Geol. *Survey Ann. Rept*, 6-9.

Gole, M. J., & Klein, C. (1981). Banded iron-formations through much of Precambrian time. *The Journal of Geology*, *89*(2), 169-183.

Grose, L., Ailleres, L., Laurent, G., Caumon, G., Jessell, M., & Armit, R. (2021). Modelling of faults in LoopStructural 1.0. *Geosci. Model Dev.*, *14*(10), 6197-6213. https://doi.org/10.5194/gmd-14-6197-2021

Grose, L., Ailleres, L., Laurent, G., & Jessell, M. (2021). LoopStructural 1.0: time-aware geological modelling. *Geosci. Model Dev.*, *14*(6), 3915-3937. https://doi.org/10.5194/gmd-14-3915-2021

Horwitz, R. C. (1976). *Two unrecorded basal sections in older Proterozoic rocks of Western Australia*. CSIRO, Minerals Research Laboratories, Division of Mineralogy.

Howard, H. M., & Martin, D. (2018). Focusing on the Fortescue and Hamersley Basins. In *GSWA 2018 extended abstracts* (Vol. Record 2018, pp. 14-17). GSWA. www.dmp.wa.gov.au/GSWApublications

Jessell, M., Ogarko, V., de Rose, Y., Lindsay, M., Joshi, R., Piechocka, A., Grose, L., de la Varga, M., Ailleres, L., & Pirot, G. (2021). Automated geological map deconstruction for 3D model construction using map2loop 1.0 and map2model 1.0. *Geosci. Model Dev.*, *14*(8), 5063-5092. https://doi.org/10.5194/gmd-14-5063-2021

Kepert, D. A. (2018). The Mapped Stratigraphy and Structure of the Mining Area C Region, Hamersley Province. *GSWA*, *Report 185*, 282p.

Lascelles, D. F. (2000). Marra Mamba Iron Formation stratigraphy in the eastern Chichester Range, Western Australia. *Australian Journal of Earth Sciences*, *47*(4), 799-806. https://doi.org/10.1046/j.1440-0952.2000.00810.x

Lascelles, D. F. (2002). *A New Look at Old Rocks — An Alternative Model for the Origin of In Situ Iron Ore Deposits Derived From Banded Iron-Formation* Iron Ore Conference, Perth.

Loop3D. (2020). *Loop3D Website*. https://github.com/Loop3D

MacLeod, W. N. (1966). *The geology and iron deposits of the Hamersley Range area, Western Australia*. Western Australian Geological Survey Bullettin,

MacLeod, W. N., Hunty, L. E., Jones, W. R., & Halligan, R. (1962). *A Preliminary Report on the Hamersley Iron Province, Northwest Division* (Annual Report of the Geological Survey Branch for the year 1962, Issue. GSWA.

Maitland, A. G. (1904). *Preliminary report on the geological features and mineral resources of the Pilbara goldfield*. Geological Survey.

Maitland, A. G. (1905). *Further report on the geological features and mineral resources of the Pilbara goldfield*. Government Printer.

Maitland, A. G. (1906). *Third report on the geological features and mineral resources of the Pilbara goldfield*. Government Printer.

Maitland, A. G. (1908). *The geological features and mineral resources of the Pilbara goldfield*. Hudson & Kearns.

Maitland, A. G. (1909). *Geological investigations in the country lying 21°30' and 25°30'S lat. and 113°30' and 118°30'E long., embracing parts of the Gascoyne, Ashburton and West Pilbara Goldfields*. Government Printer. https://nla.gov.au/nla.cat-vn1537033

Maitland, A. G., & Jackson, C. F. V. (1903). *The mineral production of Western Australia up to the end of the year 1903*.

Miles, K. R. (1942). *The blue asbestos bearing banded iron formations of the Hamersley Ranges, Western Australia : Pt. 1 / by Keith R. Miles ; and Pt. II. The blue asbestos deposits of the Hamersley Range and their economic importance, by J.S. Foxall*. Government Printer.

Morris, R. C., & Horwitz, R. C. (1983). The origin of the iron-formation-rich Hamersley Group of Western Australia — deposition on a platform. *Precambrian Research*, *21*(3), 273-297. https://doi.org/https://doi.org/10.1016/0301-9268(83)90044-X

Perring, C., Crowe, M., & Hronsky, J. (2020). A New Fluid-Flow Model for the Genesis of Banded Iron Formation-Hosted Martite-Goethite Mineralization, with Special Reference to the North and South Flank Deposits of the Hamersley Province, Western Australia. *Economic Geology*, *115*(3), 627-659. https://doi.org/10.5382/econgeo.4734

Pirot, G., Joshi, R., Giraud, J., Lindsay, M. D., & Jessell, M. W. (2022). loopUI-0.1: indicators to support needs and practices in 3D geological modelling uncertainty quantification. *Geosci. Model Dev.*, *15*(12), 4689-4708. https://doi.org/10.5194/gmd-15-4689-2022

Putz, M., Stuwe, K., Jessell, M., & Calcagno, P. (2006). Three-dimensional model and late stage warping of the Plattengneis Shear Zone in the Eastern Alps. *Tectonophysics*, *412*(1-2), 87-103. https://doi.org/10.1016/j.tecto.2005.10.003

Shibuya, T., Aoki, K., Komiya, T., & Maruyama, S. (2010). Stratigraphy-related, low-pressure metamorphism in the Hardey Syncline, Hamersley Province, Western Australia. *Gondwana Research*, *18*(1), 213-221. https://doi.org/https://doi.org/10.1016/j.gr.2010.01.002

Simonson, B. M., Hassler, S. W., & Schubel, A. (1993). Lithology and proposed revisions in stratigraphic nomenclature of the Wittenoom Formation (Dolomite) and overlying formations, Hamersley Group, Western Australia. *Professional Papers*, *34*, 65-80.

Smith, R. E., Perdrix, J. L., & Parks, T. C. (1982). Burial Metamorphism in the Hamersley Basin, Western Australia. *Journal of Petrology*, *23*(1), 75-102.

Talbot, H. W. B. (1920). *The geology and mineral resources of the north-west, central and eastern divisions / by H.W.B. Talbot; Petrology by R.A. Farquharson*. Government Printer.

Thorne, A. M., & Trendall, A. F. (2001). *Geology of the Fortescue Group, Pilbara Craton, Western Australia*. Geological Survey of Western Australia.

Trendall, A. F. (1968). Three Great Basins of Precambrian Banded Iron Formation Deposition: A Systematic Comparison. *GSA Bulletin*, *79*(11), 1527-1544. https://doi.org/10.1130/0016-7606(1968)79[1527:Tgbopb]2.0.Co;2

Trendall, A. F. (1976). *Chemical composition of the Brockman Iron Formation*. Western Australia Geological Survey.

Trendall, A. F. (1979). A revision of the Mount Bruce Supergroup. *Geol. Surv. West. Aust. Ann. Rep.*, 48-53.

Trendall, A. F. (1983). Chapter 1 Introduction. In A. F. Trendall & R. C. Morris (Eds.), *Developments in Precambrian Geology* (Vol. 6, pp. 1-12). Elsevier. https://doi.org/https://doi.org/10.1016/S0166-2635(08)70040-9

Trendall, A. F. (1990). Hamersley Basin. In *Geology and Mineral Resources of Western Australia* (Vol. Memoir 3, pp. 163-189). GSWA.

Trendall, A. F., & Blockey, J. B. (1970). The iron formations of the Precambrian Hamersley Group, Western Australia with special reference to the associated crocidolite, by A. F. Trendall and J. B. Blockley. *GSWA Bullettin*, *119*(Accessed from https://nla.gov.au/nla.cat-vn2432399), 366p.

Trendall, A. F., Compston, W., Nelson, D. R., De Laeter, J. R., & Bennett, V. C. (2004). SHRIMP zircon ages constraining the depositional chronology of the Hamersley Group, Western Australia. *Australian Journal of Earth Sciences*, *51*(5), 621-644. https://doi.org/10.1111/j.1400-0952.2004.01082.x

Trendall, A. F., Nelson, D. R., De Laeter, J. R., & Hassler, S. W. (1998). Precise zircon U-Pb ages from the Marra Mamba Iron Formation and Wittenoom Formation, Hamersley Group, Western Australia. *Australian Journal of Earth Sciences*, *45*(1), 137-142. https://doi.org/10.1080/08120099808728374

Williams, G. E. (2023). Cyclic tidalites and seismites at a submarine hydrothermal system for a 2450 Ma banded iron formation, Hamersley Basin, Western Australia. *Australian Journal of Earth Sciences*, *70*(3), 323-331. https://doi.org/10.1080/08120099.2023.2150682

APPENDICES

1. Sampling Codes

The Python codes used for drillhole data sampling:

```
import psycopg2
import pandas as pd

try:
    connection = psycopg2.connect(database="loop", #Postgresql
Credentials
                                  host="localhost",
                                  user="username",
                                  password="password",
                                  port="5432")
    # Get Grid Count
    cursor2 = connection.cursor()
    cursor3 = connection.cursor()
    sql = 'SELECT COUNT(*) from grid;' #Getting total number of
grid
    total_collar_query = 'SELECT COUNT(*) FROM collars;' #
#Getting total number of collars
    total_collars=[]
    cursor3.execute(total_collar_query,total_collars)
    data=[]
    cursor2.execute(sql,data)
    collar_count = cursor3.fetchone()
    for total_collar in collar_count:
            print("Total collars = ", total_collar)
    grid_count = cursor2.fetchone()
    for row in grid_count:
            print("Total grids = ", row)
except (Exception, psycopg2.Error) as error:
    print("Error while fetching data from PostgreSQL", error)
finally:
    if connection:
        cursor2.close()
        connection.close()
        print("PostgreSQL connection is closed")

#Sub-area Sampling
for perc in range(10, total_collar, 10):
    output = "collars_" + str(perc) + ".csv"
    print(output)
    try:
        connection = psycopg2.connect(database="loop",
#Postgresql Credentials
                                  host="localhost",
                                  user="username",
```

```
                        password="password",
                        port="5432")
        cursor = connection.cursor()
        results = []
        for x in range(1, row):
            sql_query = "SELECT bhid, east, north, elev, depth
FROM collars where grid_id=" + str(
                x) + " order by bhid limit (select AVG(perc" +
str(
                percentage) + ") from collars where grid_id=" +
str(x) + ")" #subsampling query
            cursor.execute(sql_query)
            collars = cursor.fetchall()
            if len(collars) > 0:
                data = pd.DataFrame(collars, columns=['BHID',
'EAST', 'NORTH', 'ELEV', 'DEPTH'])
                results.append(data)
            else:
                continue
        print(sql_query)
        df = pd.concat(results)
        df.to_csv(output)
        print(output + " exported")
    except (Exception, psycopg2.Error) as error:
        print("Error while fetching data from PostgreSQL", error)
    finally:
        if connection:
            cursor.close()
            connection.close()
            print("PostgreSQL connection is closed")
#Global Sampling
print("Total number of collars: " + str(total_collar))
for perc in range(10, total_collar, 10):
    percentage = int(perc * total_collar / 100)
    print("Number of selected collars: "+str(percentage))
    print("Percentage of selected collars: " + str(perc))
    routput = "collars_" + str(perc) + "_random.csv"
    try:
        cursor4 = connection.cursor()
        rresults = []
        rsql_query = "SELECT bhid, east, north, elev, depth FROM
collars ORDER BY RANDOM() LIMIT " + str(
            percentage) + ";"
        print(rsql_query)
        cursor4.execute(rsql_query)
        rcollars = cursor4.fetchall()
        if len(rcollars) > 0:
            rdata = pd.DataFrame(rcollars, columns=['BHID',
'EAST', 'NORTH', 'ELEV', 'DEPTH'])
```

```
            rresults.append(rdata)
        else:
            continue
        print(sql_query)
        rdf = pd.concat(results)
        rdf.to_csv(routput)
        print(routput + " exported")
    except (Exception, psycopg2.Error) as error:
        print("Error while fetching data from PostgreSQL", error)
    finally:
        if connection:
            cursor4.close()
            connection.close()
            print("PostgreSQL connection is closed")
```

2. Pre-processing Data and Modelling Codes

The Python codes used for data pre-processing and modelling:

```
# Set Output folder name
from datetime import datetime
import time
nowtime = datetime.now().isoformat(timespec='minutes')
model_name = nowtime.replace(":", "-").replace("T", "-")
model_name = "Output-" + model_name

import pandas as pd
import numpy as np
from array import *
from math import degrees, asin, atan2, sqrt
from scipy.spatial import Delaunay
for perc in range(10, 110, 10):
    def check_polarity():
        strat = pd.read_csv('strat.csv')
        collars = pd.read_csv('collars_' + str(perc) + '.csv')
        strat_order = pd.read_csv('strat_order.csv')

        merged = pd.merge(collars, strat, on="BHID", how="outer",
suffixes=("_l", "_r"))
        merged = merged[merged.duplicated(subset=["BHID", "STRAT"])
== False]
        merged = merged.dropna(axis=0, subset=['EAST'])

        # merging strata_order number with existing merged data
(collars+strat)
        merged_order = pd.merge(merged, strat_order, on="STRAT",
how="outer", suffixes=("_l", "_r"))
        merged_order =
merged_order[merged_order.duplicated(subset=["BHID", "STRAT"]) ==
False]
        merged_order = merged_order.dropna(axis=0, subset=['EAST'])
        polarity = []
        for borehole in collars["BHID"]:
            BH = merged_order[merged_order["BHID"] == borehole]
            BH = BH[BH["num_strat"] > 0]
            if len(BH) < 2:
                polarity.append(-2)   # no strat contacts
```

```
                continue

            BH_sort_to = BH.sort_values("TO", ascending=True)
            BH_sort_num_strat_desc = BH.sort_values("num_strat",
ascending=False)
            BH_sort_num_strat_asc = BH.sort_values("num_strat",
ascending=True)
            BH_sort_to_arr = BH_sort_to["num_strat"].to_numpy()
            BH_sort_numstrat_asc_arr =
BH_sort_num_strat_asc["num_strat"].to_numpy()
            BH_sort_numstrat_desc_arr =
BH_sort_num_strat_desc["num_strat"].to_numpy()

            test1 = BH_sort_to_arr - BH_sort_numstrat_asc_arr
            test2 = BH_sort_to_arr - BH_sort_numstrat_desc_arr

            if (not np.any(test1)):
                polarity.append(1)  # regular
            elif (not np.any(test2)):
                polarity.append(0)  # inverted
            else:
                polarity.append(-1)  # unknown

        collars["polarity"] = np.array(polarity)
        collars.to_csv("collars_polarity_" + str(perc) + ".csv")

    def save_contacts(merged):
        df = pd.DataFrame(
            columns=['index', 'X', 'Y', 'Z', 'formation', 'source'])
        i = 0
        collars = pd.read_csv("collars_polarity_" + str(perc) +
".csv")

        collars = collars[collars["polarity"] > -1]
        collars = collars.set_index("BHID")

        for bhid in collars.index:
            BH = merged[merged["BHID"] == bhid]
            if collars.loc[bhid].polarity == 1:
                for ind, bh in BH[:-1].iterrows():
                    tmp_data = [i, bh.EAST, bh.NORTH,
collars.loc[bhid].ELEV - bh.TO, bh.STRAT, bhid]
                    df.loc[i] = tmp_data
                    i = i + 1
            else:
                for ind, bh in BH[1:].iterrows():
                    tmp_data = [i, bh.EAST, bh.NORTH,
collars.loc[bhid].ELEV - bh.FROM, bh.STRAT, bhid]
                    df.loc[i] = tmp_data
                    i = i + 1
        df.to_csv("dh_contacts_both.csv", index=False)
    def common_elements(list1, list2, list3):
        return list(set(list1) & set(list2) & set(list3))

    def dircos2ddd(l, m, n):
        dipdir = degrees(atan2(l, m)) % 360
        dip = 90 - degrees(asin(n))
        if dip > 90:
            dip = 180 - dip
            dipdir = dipdir + 180
        dipdir = dipdir % 360
```

```
        return (dip, dipdir)

    def calc_dd_dip(merged, filtered_triangles):
        # display(filtered_triangles)
        collars = pd.read_csv("collars_polarity_"+ str(perc) +
".csv")
        collars = collars.set_index('BHID')
        df = pd.DataFrame(
            columns=['X', 'Y', 'Z', 'azimuth', 'dip', 'polarity',
'formation', 'source'])
        if (len(filtered_triangles) == 0):
            return (df)
        i = 0
        for ft in filtered_triangles:

            BH1 = merged[merged["BHID"] == ft[0]]
            BH2 = merged[merged["BHID"] == ft[1]]
            BH3 = merged[merged["BHID"] == ft[2]]

            if BH1.iloc[0].polarity == BH2.iloc[0].polarity and
BH1.iloc[0].polarity == BH3.iloc[0].polarity:
                if BH1.iloc[0].polarity == 1:
                    same = (common_elements(BH1.STRAT[:-
1].values.tolist(), BH2.STRAT[:-1].values.tolist(),
                                            BH3.STRAT[:-
1].values.tolist()))
                else:
                    same =
(common_elements(BH1.STRAT[1:].values.tolist(),
BH2.STRAT[1:].values.tolist(),

BH3.STRAT[1:].values.tolist()))
                BH1 = BH1.set_index("STRAT")
                BH2 = BH2.set_index("STRAT")
                BH3 = BH3.set_index("STRAT")
                for s in same:
                    # display(BH1)
                    if BH1.loc[s].polarity == 1:
                        p1 = np.array([BH1.loc[s].EAST,
BH1.loc[s].NORTH, collars.loc[ft[0]].ELEV - BH1.loc[s].TO])
                        p2 = np.array([BH2.loc[s].EAST,
BH2.loc[s].NORTH, collars.loc[ft[1]].ELEV - BH2.loc[s].TO])
                        p3 = np.array([BH3.loc[s].EAST,
BH3.loc[s].NORTH, collars.loc[ft[2]].ELEV - BH3.loc[s].TO])
                    else:
                        p1 = np.array([BH1.loc[s].EAST,
BH1.loc[s].NORTH, collars.loc[ft[0]].ELEV - BH1.loc[s].FROM])
                        p2 = np.array([BH2.loc[s].EAST,
BH2.loc[s].NORTH, collars.loc[ft[1]].ELEV - BH2.loc[s].FROM])
                        p3 = np.array([BH3.loc[s].EAST,
BH3.loc[s].NORTH, collars.loc[ft[2]].ELEV - BH3.loc[s].FROM])

                    # These two vectors are in the plane
                    v1 = p3 - p1
                    v2 = p2 - p1

                    # the cross product is a vector normal to the
plane
                    cp = np.cross(v1, v2)
                    cp = cp / np.sqrt(np.sum(cp ** 2))
                    a, b, c = cp
```

```

                    # This evaluates a * x3 + b * y3 + c * z3 which equals d
                    d = np.dot(cp, p3)
                    dip, dipdir = dircos2ddd(a, b, c)
                    p_avg = (p1 + p2 + p3) / 3
                    tmp_data = [p_avg[0], p_avg[1], p_avg[2], dipdir, dip, int(BH1.iloc[0].polarity), s,
                                str(ft[0]) + '_' + str(ft[1]) + '_' + str(ft[2])]
                    df.loc[i] = tmp_data
                    i = i + 1
        return (df)

    def filter_delaunay_triangles(points_df, max_angle_degree=None, max_length=None, polarity=1):
        points_df = points_df[points_df["polarity"] == polarity]
        print('polarities', polarity, len(points_df))
        if (len(points_df) == 0):
            return ([])
        points = np.array(points_df[["EAST", "NORTH"]])

        tri = Delaunay(points, furthest_site=False)
        filtered_tri_simplices = []
        for t in tri.simplices:

            A, B, C = points[t[0]], points[t[1]], points[t[2]]

            if not (max_length is None):
                l1 = (np.sum((B - A) ** 2)) ** 0.5
                l2 = (np.sum((C - A) ** 2)) ** 0.5
                l3 = (np.sum((B - C) ** 2)) ** 0.5
                edge_lengths = np.array([l1, l2, l3])

                if np.any(edge_lengths > max_length):
                    continue
            if not (max_angle_degree is None):
                e1 = B - A
                e2 = C - A
                num = np.dot(e1, e2)
                denom = np.linalg.norm(e1) * np.linalg.norm(e2)
                d1 = np.rad2deg(np.arccos(num / denom))
                e1 = C - B
                e2 = A - B
                num = np.dot(e1, e2)
                denom = np.linalg.norm(e1) * np.linalg.norm(e2)
                d2 = np.rad2deg(np.arccos(num / denom))
                d3 = 180 - d1 - d2
                degs = np.array([d1, d2, d3])
                if np.any(degs > max_angle_degree):
                    continue
                print(points_df.iloc[t[0]]["BHID"], points_df.iloc[t[1]]["BHID"], points_df.iloc[t[2]]["BHID"])

            filtered_tri_simplices.append(
                [points_df.iloc[t[0]]["BHID"], points_df.iloc[t[1]]["BHID"], points_df.iloc[t[2]]["BHID"]])

        return filtered_tri_simplices

    import pandas as pd
```

```
    check_polarity()
    strat = pd.read_csv('strat.csv')
    points = pd.read_csv("collars_polarity_" + str(perc) + ".csv")
 # if NOT hand filtered
    merged = pd.merge(points, strat, on="BHID", how="outer",
suffixes=("_l", "_r"))
    merged = merged[merged.duplicated(subset=["BHID", "STRAT"]) ==
False]
    merged = merged.dropna(axis=0, subset=['EAST'])
    strat_order = pd.read_csv('strat_order.csv')
    merged = pd.merge(merged, strat_order, on="STRAT", how="outer",
suffixes=("_l", "_r"))
    merged = merged[merged["num_strat"] > 0]
    save_contacts(merged)
    print(len(merged))

    max_angle_degree = None  # None or a number in degree 100
    max_length = 1000  # None or a number 120

    filtered_triangles = filter_delaunay_triangles(merged,
max_angle_degree, max_length, polarity=1)
    df_normal = calc_dd_dip(merged, filtered_triangles)

    filtered_triangles = filter_delaunay_triangles(merged,
max_angle_degree, max_length, polarity=0)
    df_inverted = calc_dd_dip(merged, filtered_triangles)

    df = pd.concat([df_normal, df_inverted])
    df.to_csv('drillhole_dipd_both.csv', index=False)

    import pandas as pd
    import numpy as np
    from LoopStructural import GeologicalModel
    from LoopStructural.visualisation import LavaVuModelViewer
    import os
    from os import listdir
    from os.path import isfile, join
    from pathlib import Path
    import random
    def common_elements(list1, list2):
        return list(set(list1) & set(list2))

# define your own coorninates here
    minx = #minx coordinate
    miny = #miny_coordinate
    maxx = #maxx_coordinate
    maxy = #maxy_coordinate
    base = #base_elevation
    top = #base_elevation

    vtk_path = model_name
    print(vtk_path)

    print('x_length: ' + str((maxx - minx) / 1000) + 'km')
    print('y_width: ' + str((maxy - miny) / 1000) + 'km')
    print('z_depth: ' + str((top - base) / 1000) + 'km')

    # Estimation of the thickness between two consecutive
stratigraphic boundaries
    def estimate_thickness_btw_bdies(dh_contacts, strat_bdies):
        # input: drillhole basal contacts, stratigraphic boundaries
```

```
of the considered geological unit
        # find drillholes containing both boundaries
        dh_ix_yng = dh_contacts[
            dh_contacts['formation'] ==
strat_bdies[0]].source.values  # drillholes containing the top
contact
        dh_ix_old = dh_contacts[
            dh_contacts['formation'] ==
strat_bdies[1]].source.values  # drillholes containing the bottom
contact
        dh_ix_yng_and_old = common_elements(dh_ix_yng, dh_ix_old)  #
drillholes containing both top & bottom contact
        if len(dh_ix_yng_and_old) > 0:
            # get top boundary elevations
            ztop = dh_contacts[
                (dh_contacts['formation'] == strat_bdies[0]) &
(dh_contacts['source'].isin(dh_ix_yng_and_old))].Z.values
            # get bottom boundary elevations
            zbtm = dh_contacts[
                (dh_contacts['formation'] == strat_bdies[1]) &
(dh_contacts['source'].isin(dh_ix_yng_and_old))].Z.values
            # estimate thickness by averaging absolute values
            thickness = np.mean(np.abs(ztop - zbtm))
        else:
            thickness = np.nan
        return thickness

    # Estimation of all stratigraohic unit thicknesses
    def estimate_thickness_strat(strat_order, dh_contacts):
        # order by relative age
        strat_order.sort_values('num_strat', inplace=True,
ignore_index=True)
        # keep stratigraphic units whose relative os greater than 0
(0 means sediments or not of interest)
        strat_list = strat_order[strat_order['num_strat'] >
0].STRAT.values
        # initialize
        strat_order['thickness'] = np.nan
        # iterate for each stratigraphic unit
        for i in range(len(strat_list) - 1):
            # get the stratigraphic unit boundaries
            strat_bdies = strat_list[i:i + 2]  # from youngest to
oldest
            # get index of stratigraphic unit of interest
            ix = np.asarray(np.where(strat_order["STRAT"] ==
strat_list[i + 1])).flatten()
            # estimate thickness for tratigraphic unit of interest
            strat_order.loc[ix, "thickness"] =
estimate_thickness_btw_bdies(dh_contacts, strat_bdies)
        # if estimation not possible, try to get the default value
        if 'default_thickness' in strat_order.columns:
            print('default_thickness exists')
            strat_order.loc[strat_order['thickness'].isna(),
['thickness']] = strat_order.loc[
                strat_order['thickness'].isna(),
['default_thickness']].values
        # if no default thickness exist then define new default
value as mean of what has been estimated
        default_thickness = strat_order['thickness'].mean()
        strat_order.loc[strat_order['thickness'].isna(),
['thickness']] = default_thickness
```

```
            # compute pseudo-depth
            strat_order["pseudo_depth"] = -
strat_order.thickness.cumsum()
            return
        # # Download data and remove duplicates

    contacts_clean = pd.read_csv("./contacts_clean.csv")
    contacts_clean.drop(columns=["level_0", "index"], inplace=True)
    contacts_clean.drop_duplicates(inplace=True, ignore_index=True)

    dh_contacts_both = pd.read_csv("./dh_contacts_both.csv")
    dh_contacts_both.drop(columns=["index"], inplace=True)
    dh_contacts_both.drop_duplicates(inplace=True,
ignore_index=True)

    drillhole_dipd_both = pd.read_csv("./drillhole_dipd_both.csv")
    drillhole_dipd_both.drop_duplicates(inplace=True,
ignore_index=True)

    fault_dimensions = pd.read_csv("./fault_dimensions.csv")
    fault_dimensions.drop_duplicates(inplace=True,
ignore_index=True)

    fault_orientations = pd.read_csv("./fault_orientations.csv")
    fault_orientations.drop_duplicates(inplace=True,
ignore_index=True)

    faults = pd.read_csv("./faults.csv")
    faults.drop_duplicates(inplace=True, ignore_index=True)

    strat_order = pd.read_csv("./strat_order.csv")
    strat_order.drop_duplicates(subset=['STRAT'], inplace=True,
ignore_index=True)

    dtm = pd.read_csv("./dtm.csv", header=None, sep=' ', names=["X",
"Y", "Z"])
    ix2drop = np.asarray(np.where(dtm["Z"].values == 0)).flatten()

    from scipy.interpolate import RegularGridInterpolator, griddata

    dx = dy = 100
    xvec = np.linspace(minx, maxx, int(np.round((maxx - minx) /
dx)))
    yvec = np.linspace(miny, maxy, int(np.round((maxy - miny) /
dy)))
    xx, yy = np.meshgrid(xvec, yvec, indexing='ij')
    zz_dtm = griddata(dtm[["X", "Y"]].values, dtm["Z"].values, (xx,
yy), method='linear')
    zz_dtm[0, :] = zz_dtm[1, :]
    zz_dtm[:, -1] = zz_dtm[:, -2]

    dtm_interpolator = RegularGridInterpolator((xvec, yvec), zz_dtm)

    [xmin, ymin, zmin, xmax, ymax, zmax] = [minx, miny, base, maxx,
maxy, top]
    [xmin, ymin, zmin, xmax, ymax, zmax]

    # Check if data inside bounding box
    inside_bb_contacts_clean = ((contacts_clean['X'].min() > xmin) &
(contacts_clean['X'].max() < xmax) &
                                (contacts_clean['Y'].min() > ymin) &
```

```
(contacts_clean['Y'].max() < ymax) &
                                (contacts_clean['Z'].min() > zmin) &
(contacts_clean['Z'].max() < zmax))

    inside_bb_dh_contacts_both = ((dh_contacts_both['X'].min() >
xmin) & (dh_contacts_both['X'].max() < xmax) &
                                  (dh_contacts_both['Y'].min() >
ymin) & (dh_contacts_both['Y'].max() < ymax) &
                                  (dh_contacts_both['Z'].min() >
zmin) & (dh_contacts_both['Z'].max() < zmax))

    inside_bb_dh_dipd_both = ((drillhole_dipd_both['X'].min() >
xmin) & (drillhole_dipd_both['X'].max() < xmax) &
                              (drillhole_dipd_both['Y'].min() >
ymin) & (drillhole_dipd_both['Y'].max() < ymax) &
                              (drillhole_dipd_both['Z'].min() >
zmin) & (drillhole_dipd_both['Z'].max() < zmax))

    inside_bb_fault_orientations = ((fault_orientations['X'].min() >
xmin) & (fault_orientations['X'].max() < xmax) &
                                    (fault_orientations['Y'].min() >
ymin) & (fault_orientations['Y'].max() < ymax) &
                                    (fault_orientations['Z'].min() >
zmin) & (fault_orientations['Z'].max() < zmax))

    inside_bb_faults = ((faults['X'].min() > xmin) &
(faults['X'].max() < xmax) &
                        (faults['Y'].min() > ymin) &
(faults['Y'].max() < ymax) &
                        (faults['Z'].min() > zmin) &
(faults['Z'].max() < zmax))

    inside_bb = (inside_bb_contacts_clean &
inside_bb_dh_contacts_both & inside_bb_dh_dipd_both &
                 inside_bb_fault_orientations & inside_bb_faults)

    print('All data inside model boundaries: ' + str(inside_bb))
    if not inside_bb:
        print('All contacts_clean data inside model boundaries: ' +
str(inside_bb_contacts_clean))
        print('All dh_contacts_both data inside model boundaries: '
+ str(inside_bb_dh_contacts_both))
        print('All dh_dipd_both data inside model boundaries: ' +
str(inside_bb_dh_dipd_both))
        print('All fault_orientations data inside model boundaries:
' + str(inside_bb_fault_orientations))
        print('All faults data inside model boundaries: ' +
str(inside_bb_faults))

    (dh_contacts_both['Z'].min() > zmin) &
(dh_contacts_both['Z'].max() < zmax)

    # Estimate stratigraphic unit thicknesses if possible, otherwise
apply default values or default rule (mean)
    estimate_thickness_strat(strat_order, dh_contacts_both)
    strat_order.rename(columns={"STRAT": "formation"}, inplace=True)
    print(strat_order[["formation", "thickness"]])

    # # Prepare basal contacts

    df_contacts = pd.DataFrame(columns=["X", "Y", "Z",
```

```
"feature_name", "val", "nx", "ny", "nz", "formation"])
    df_contacts[["X", "Y", "Z", "formation"]] =
pd.concat([contacts_clean[["X", "Y", "Z", "formation"]],

dh_contacts_both[["X", "Y", "Z", "formation"]]
                                                                  ])
    df_contacts["feature_name"] = 'strat'
    df_contacts = (df_contacts.set_index('formation')).join(
        (strat_order[["formation",
"pseudo_depth"]]).set_index('formation')).reset_index(drop=True)
    df_contacts["val"] = df_contacts["pseudo_depth"].values
    df_contacts.drop(columns=["pseudo_depth"], inplace=True)
    df_contacts.head()

    print(df_contacts["val"].sort_values().unique())

    # ## Prepare contact orientations
    from LoopStructural.utils.helper import strike_dip_vector

    df_orientations = pd.DataFrame(columns=["X", "Y", "Z",
"feature_name", "val", "nx", "ny", "nz"])
    for i in range(drillhole_dipd_both.shape[0]):
        df_orientations.loc[i, "X"] = drillhole_dipd_both.loc[i,
"X"]
        df_orientations.loc[i, "Y"] = drillhole_dipd_both.loc[i,
"Y"]
        df_orientations.loc[i, "Z"] = drillhole_dipd_both.loc[i,
"Z"]
        df_orientations.loc[i, ["nx", "ny", "nz"]] =
strike_dip_vector(
            np.asarray(drillhole_dipd_both.loc[i, "azimuth"] -
90).flatten(),
            np.asarray(drillhole_dipd_both.loc[i, "dip"]).flatten())

    df_orientations["feature_name"] = 'strat'
    df_orientations.head()

    # ## Prepare Fault data

    df_fault_contacts = faults.rename(columns={"formation":
"feature_name"})
    df_fault_contacts["val"] = 0
    df_fault_contacts["coord"] = 0
    df_fault_contacts.head()

    fault_orientations  # .head()

    # RE-ESTIMATE FAULT ORIENTATIONS AS GIVEN DipDirection is -999.0
!!!
    faultid = fault_dimensions["Fault"].unique()
    nfaults = faultid.size
    df_fault_orientations = pd.DataFrame(columns=["X", "Y", "Z",
"feature_name", "val", "nx", "ny", "nz", "coord"])
    for f in range(nfaults):
        df_tmp = pd.DataFrame(columns=["X", "Y", "Z",
"feature_name", "val", "nx", "ny", "nz", "coord"])
        ix = np.asarray(np.where(df_fault_contacts["feature_name"]
== faultid[f])).flatten()
        tmp_X = faults.loc[ix, "X"].values
        tmp_Y = faults.loc[ix, "Y"].values
        tmp_Z = faults.loc[ix, "Z"].values
```

```
        tmp_nx = np.ones(len(tmp_X) - 1)
        tmp_ny = -(tmp_X[1:] - tmp_X[:-1]) / (tmp_Y[1:] - tmp_Y[:-
1]) * tmp_nx
        tmp_no = np.sqrt(tmp_nx ** 2 + tmp_ny ** 2)
        tmp_nx = tmp_nx / tmp_no
        tmp_ny = tmp_ny / tmp_no
        ixn = np.asarray(np.where(np.abs(tmp_ny) ==
np.inf)).flatten()
        if len(ixn) > 0:
            tmp_nx[ixn] = 0
            tmp_ny[ixn] = 1.0
        tmp_X = (tmp_X[1:] + tmp_X[:-1]) / 2
        tmp_Y = (tmp_Y[1:] + tmp_Y[:-1]) / 2
        tmp_Z = (tmp_Z[1:] + tmp_Z[:-1]) / 2

        df_tmp["X"] = tmp_X
        df_tmp["Y"] = tmp_Y
        df_tmp["Z"] = tmp_Z
        df_tmp["feature_name"] = faultid[f]
        # df_tmp["nz"] = 0.0 # given that all dips are 90 degrees in
our specifi case!
        print(faultid[f])
        if (faultid[f] == 'Fault_2'):
            a = np.sqrt(2 * (tmp_nx ** 2 + tmp_ny ** 2))
            df_tmp["nx"] = tmp_nx / a
            df_tmp["ny"] = tmp_ny / a
            df_tmp["nz"] = -0.707
            print(tmp_nx, tmp_ny)
        else:
            df_tmp["nx"] = tmp_nx
            df_tmp["ny"] = tmp_ny
            df_tmp["nz"] = 0.0

        df_fault_orientations = pd.concat([df_fault_orientations,
df_tmp])
    df_fault_orientations["coord"] = 0

    df_fault_orientations  # .head()

    # # Gather into data

    data = pd.concat([df_contacts, df_orientations,
df_fault_contacts, df_fault_orientations])  # ,dtm
    data.reset_index(drop=True, inplace=True)
    data.to_csv('data.csv')
    print('Before')
    print(data.dtypes)
    # using dictionary to convert specific columns
    convert_dict = {'X': float,
                    'Y': float,
                    'Z': float,
                    'feature_name': str,
                    'nx': float,
                    'ny': float,
                    'nz': float
                    }

    data = data.astype(convert_dict)

    print('After')
    print(data.dtypes)
```

```
    data.head()

    # # Define the stratigraphic column
    # assuming a conformable stratigraphy
    stratigraphic_column = {}
    stratigraphic_column['strat'] = {}
    stratigraphic_column['strat']['basement'] = {'min': -np.inf,
'max': strat_order.loc[:, "pseudo_depth"].values[-1],
                                                  'id': 0}  #
,'colour':[0.1,0.5,0]

    nstrat = (strat_order["num_strat"] > 0).sum()
    for i in range(nstrat - 1):
        stratigraphic_column['strat'][strat_order.loc[:,
"formation"].values[-(1 + i)]] = {
            'min': strat_order.loc[:, "pseudo_depth"].values[-(1 +
i)],
            'max': strat_order.loc[:, "pseudo_depth"].values[-(2 +
i)],
            'id': i + 1}  # ,'colour':[0.8,0.1,0.8]
    stratigraphic_column['strat'][strat_order.loc[:,
"formation"].values[-(2 + i)]] = {
        'min': strat_order.loc[:, "pseudo_depth"].values[-(2 + i)],
        'max': np.inf,
        'id': i + 2}  # ,'colour':[0.1,0.5,0]

    stratigraphic_column

    # # Run LoopStructural
    surface_verts = {}

    def function(xyz, tri, name):  # for saving out vtk files
        xyz = np.copy(xyz)
        tri = np.copy(tri)
        nanmask = np.all(~np.isnan(xyz), axis=1)
        vert_idx = np.zeros(xyz.shape[0], dtype=int) - 1
        vert_idx[nanmask] = np.arange(np.sum(nanmask))
        tri[:] = vert_idx[tri]
        tri = tri[np.all(tri > -1, axis=1), :]
        xyz = xyz[nanmask, :]
        surface_verts[name] = (xyz, tri)

    def mask(xyz):  # for clipping strat to surface dtm
        from map2loop.map import MapUtil
        import rasterio
        import os
        dtm_map = MapUtil(proj.config.bbox_3d,
dtm=rasterio.open(os.path.join(dtm_path, 'dtm_rp.tif')))
        xyz = model.rescale(xyz, inplace=False)
        dtmv = dtm_map.evaluate_dtm_at_points((xyz[:, :2]))
        return xyz[:, 2] < dtmv

    filename = vtk_path + '/' + 'surface_name_{}.vtk'

    # Check whether the specified path exists or not
    isExist = os.path.exists(vtk_path)
    perc_dir = vtk_path + "/" + str(perc)
    isperc_dir = os.path.exists(perc_dir)
    print(perc_dir)
    if isExist:
        print(vtk_path + " is already exist!")
```

```
        if isperc_dir:
            print(perc_dir + " is already exist!")
        else:
            os.makedirs(vtk_path + "/" + str(perc))
            print(perc_dir + " has been created!")
    else:
        # Create a new directory because it does not exist
        os.makedirs(vtk_path)
        os.makedirs(vtk_path + "/" + str(perc))
        print("Directory " + vtk_path + " has been created!")
        print(perc_dir + " has been created!")

    model = GeologicalModel([xmin, ymin, zmin], [xmax, ymax, zmax])
    model.set_model_data(data)

    nelements = 3000
    flts = {}
    for f in range(fault_dimensions.shape[0]):  # [0]:#

        flt_name = fault_dimensions.loc[f, "Fault"]
        if (f == 0):
            flt_disp = 100
        else:

            flt_dsp = -10
        flt_center_xyz =
np.array([df_fault_contacts.loc[df_fault_contacts["feature_name"] ==
flt_name, "X"].mean(),

df_fault_contacts.loc[df_fault_contacts["feature_name"] == flt_name,
"Y"].mean(),

df_fault_contacts.loc[df_fault_contacts["feature_name"] == flt_name,
"Z"].mean()])
        flt_ext = fault_dimensions.loc[f, "HorizontalRadius"] * 5
        flt_infl = fault_dimensions.loc[f, "InfluenceDistance"] * 5
        flt_vera = fault_dimensions.loc[f, "VerticalRadius"] * 5

        flts[flt_name] = model.create_and_add_fault(flt_name,
                                                    flt_disp,

fault_center=flt_center_xyz,  #
np.array([xf.mean(),yf.mean(),zmax]),

fault_extent=flt_ext,

fault_influence=flt_infl,

fault_vertical_radius=flt_vera,

nelements=nelements,
                                                    steps=4,

interpolatortype='FDI',
                                                    buffer=0.3)

    # define fault intersection relationship
    flts['Fault_3'].add_abutting_fault(flts['Fault_2'],
positive=True)

    # IMPORTANT: FROM YOUNGEST TO OLDEST!
```

```
    strati = model.create_and_add_foliation('strat',
interpolatortype='FDI', solver='pyamg', buffer=0.5)

    dtm_0 = model.dtm = lambda xyz: dtm_interpolator(xyz[:, :2])

    model.set_stratigraphic_column(stratigraphic_column)


    # View stratigraphic surfaces and possibly faults
    view = LavaVuModelViewer(model)  # ,vertical_exaggeration=4
    view.interactive()
    view.add_model_surfaces(faults=False, function=function,
paint_with=strati)  # faults=False

    view.add_isosurface(flts['Fault_2'], isovalue=0,
function=function)
    view.add_isosurface(flts['Fault_3'], isovalue=0,
function=function)

    [[xminLSbb, yminLSbb, zminLSbb], [xmaxLSbb, ymaxLSbb, zmaxLSbb]]
= model.bounding_box
    xLS = np.linspace(xminLSbb, xmaxLSbb, nnx)
    yLS = np.linspace(yminLSbb, ymaxLSbb, nny)
    zLS = np.linspace(zminLSbb, zmaxLSbb, nnz)
    xxxLS, yyyLS, zzzLS = np.meshgrid(xLS, yLS, zLS, indexing='ij')
# build mesh
    xyzLS = np.array(
        [xxxLS.flatten(), yyyLS.flatten(), zzzLS.flatten()]).T  #
build array for LS lithocode evaluation function

    reggrid_model = model.evaluate_model(xyzLS, scale=False)

    with open(datafilepath, 'wb') as f:
        pickle.dump([reggrid_model, xLS, yLS, zLS
                     ], f)

    onlyfiles = [f for f in listdir(vtk_path) if
isfile(join(vtk_path, f))]
    print(onlyfiles)

    for layer in surface_verts:
        f = open(perc_dir + '/' + layer.replace("_iso_0.000000", "")
+ '.obj', 'w')
        vert = surface_verts[layer][0].shape[0]
        tri = surface_verts[layer][1].shape[0]
        # print(layer.replace("_iso_0.000000",""),vert,tri)
        for v in range(0, vert):
            ostr = "v {} {} {}\n" \
                .format(surface_verts[layer][0][v][0],
surface_verts[layer][0][v][1], surface_verts[layer][0][v][2])
            f.write(ostr)
        for t in range(0, tri):
            ostr = "f {} {} {}\n" \
                .format(surface_verts[layer][1][t][0] + 1,
surface_verts[layer][1][t][1] + 1,
                        surface_verts[layer][1][t][2] + 1)
            f.write(ostr)
        first = False
        f.close()

    # view scalarfield
```

```
    viewer = LavaVuModelViewer(model)  # ,vertical_exaggeration=4
    viewer.interactive()
    viewer.add_scalar_field(strati)

    from geoh5py.objects import BlockModel
    from geoh5py.workspace import Workspace
    from geoh5py.objects import Surface

    def hextofloats(h):
        return tuple(int(h[i:i + 2], 16) / 255. for i in (1, 3, 5))
    def geoh5_create_surface_data(vtk_path, colour_path):
        h5file_path = perc_dir + "/" + str(perc) + ".geoh5"
        print("Geoh5 model saved to " + h5file_path)
        try:
            os.remove(h5file_path)
        except OSError:
            pass

        workspace = Workspace(h5file_path, version=2.0)
        onlyfiles = [f for f in listdir(perc_dir) if isfile(join(perc_dir, f))]
        colour_index = 0
        for file in onlyfiles:
            if ('.obj' in file):
                obj = pd.read_csv(perc_dir + '/' + file, sep=' ', names=["code", "X", "Y", "Z"])
                indices = obj[obj['code'] == 'f']
                vertices = obj[obj['code'] == 'v']
                vertices = vertices.drop(['code'], axis=1)
                indices = indices[list("XYZ")].astype(int)
                i = indices.to_numpy() - 1
                v = vertices.to_numpy()
                if (len(i) > 0 and len(v) > 0):
                    # Create a geoh5 surface
                    surface = Surface.create(
                        workspace, name=file.replace('.obj', ''), vertices=v, cells=i
                    )
                    if ('Fault_' in file or 'dtm' in file):
                        colours = np.ones(surface.n_cells) * 99
                    else:
                        colours = np.ones(surface.n_cells) * colour_index

                        colour_index = colour_index + 1

                    surface.add_data({
                        "colour_index": {
                            "association": "CELL",
                            "values": colours
                        }
                    })

                    workspace.save_entity(surface)
        workspace.close()
    save_GA = True
    tmp_path = True
    if (save_GA):
        geoh5_create_surface_data(perc_dir, tmp_path)

    # code to take a LoopStructural voxel model and save it out
```

```
    # as a *.geoh5 GeoscienceAnalyst model
    # Requires installation of
https://github.com/MiraGeoscience/geoh5py

    from pathlib import Path
    import numpy as np

    voxel_size = 10
    sizex = int((maxx - minx) / voxel_size)
    sizey = int((maxy - miny) / voxel_size)
    sizez = int((top - base) / voxel_size)
    nsteps = [sizex, sizey, sizez]
    h5file_path = perc_dir + "/" + str(perc) + "_block.geoh5"

    def create_geoh5_block_model_data(model, voxel_size, minx, miny,
maxx, maxy, model_base, model_top, output_dir, nsteps):
        voxels =
model.evaluate_model(model.regular_grid(nsteps=(nsteps[0],
nsteps[1], nsteps[2]), shuffle=False),
                                      scale=False)
        voxels = voxels.astype(float)

        name = "MyLoopBlockModel"

        # Generate a 3D array

        nodal_x = np.arange(0, maxx - minx + 1, voxel_size)
        nodal_y = np.arange(0, maxy - miny + 1, voxel_size)
        nodal_z = np.arange(model_top - model_base + 1, 0, -
voxel_size)

        try:
            os.remove(h5file_path)
        except OSError:
            pass

        # Create a workspace
        workspace = Workspace(h5file_path, version=2.0)

        grid = BlockModel.create(
            workspace,
            origin=[minx + (voxel_size / 2), miny + (voxel_size /
2), model_base + (voxel_size / 2)],
            u_cell_delimiters=nodal_x,
            v_cell_delimiters=nodal_y,
            z_cell_delimiters=nodal_z,
            name=name,
            rotation=0,
            allow_move=False,
        )
        data = grid.add_data(
            {
                "DataValues": {
                    "association": "CELL",
                    "values": (
                        voxels.reshape((nodal_x.shape[0] - 1,
nodal_y.shape[0] - 1, nodal_z.shape[0] - 1)).transpose(
                            (1, 0, 2))
                    ),
                }
            }
```

```
        )
        workspace.save_entity(grid)
        workspace.close()

    # output_dir= ''   # output directory to save geoh5 format voxel
mdoel
    create_geoh5_block_model_data(model, voxel_size, minx, miny,
maxx, maxy, base, top, perc_dir, nsteps)
    print("Voxet model saved to " + h5file_path)
```