Reliable Prospection and Exploration

Adam Kosík Michael Schreiner Simon Wiesinger Djamel Ziane

19.01.2023

Inhaltsverzeichnis

KAPITEL 1

Einführung

Wir befassen uns mit der Kosten-Nutzen-Bewertung von Prospektionsverfahren. Dazu soll ein Software-Tool entwickelt werden, das bei der Planung der Prospektion die Verfahren hinsichtlich Kosten und Nutzen bei der Planung der Prospektion bewertet. Das Tool soll die potentiellen Methoden beinhalten, aber auch die Einbeziehung anderer Methoden ermöglichen.

Die Motivation ist eine Beurteilung, welchen Beitrag eine Messmethode oder eine Kombination von Messmethoden liefern kann, bevor die Messung durchgeführt wurde. In diesem Projekt soll versucht werden, mit einem quantitativen Verfahren für ein Geothermievorhaben zu bewerten, welche Methode, mit welchen Parametern, in welcher Kombination, welchen Beitrag liefern kann. Für die geothermische Prospektion/Exploration gibt es zahlreiche verschiedene Messmethoden. In der Praxis ist es häufig unklar, welche Methoden für ein geplantes Geothermievorhaben geeignet ist. Es fehlt typischerweise an validierten Aussagen über

- die Aussagekraft der einzelnen Methoden für die Prospektion mit einer quantitativen Bewertung der Unsicherheiten der Methoden im geologischen Kontext,
- den Einfluss der Messparameter auf die Aussagekraft und Unsicherheiten der einzelnen Messmethoden,
- ein optimales Vorgehen für die Messungen im Hinblick auf Kosten und Nutzen für die Investoren.

Ziel dieses Projektes ist die Entwicklung einer Methode und eines Software-Tools, um für Förderagenturen oder einen Investor die Erfolgsaussichten einer Prospektion aussagekräftige und belastbare Erkenntnisse für die weiterführende Exploration zu liefern und damit die Projektrisiken zu reduzieren. Dabei gehen synergetisch sowohl Expertenwissen vor den Messungen als auch die erzielten Messwerte ein. Die wissenschaftlich fundierten Resultate sollen in verständlicher, z. B. graphisch basierter Form dargeboten werden.

Es werden folgende Schritte angestrebt:

- Für jedes Prospektions/Explorationsverfahren werden eine grosse Klasse an synthethischen geologischen Modellen automatisch erzeugt. Hierfür geht sowohl geologisches als auch numerisches Expertenwissen ein. Diese Daten werden mit «virtuellen Sensoren» vermessen. Es werden also Daten erzeugt, die den Daten entsprechen, die reale Sensoren in der entsprechenden Situation erkennen würden.
- Diese konzeptuellen Daten werden systematisch mit den zur Verfügung stehenden Prospektions-/Explorationsverfahren analysiert. Hierbei werden auch Parameter der Verfahren einbezogen, wie z.B. die lokale Dichte der Sensoren, die Qualität der Sensoren, die Anzahl der Messwiederholungen, etc.

- Für diese Prospektions-/Explorationsverfahren müssen Bewertungskriterien entwickelt werden, mit denen die Vorhersagequalität der einzelnen Verfahren bestimmt werden kann. Konkret muss es möglich sein, beide Fehlerarten, nämlich das fehlerhafte Verwerfen eines für die Geothermie geeigneten Gebiets als auch das fehlerhafte zu positive Bewerten eines Gebietes zu quantifizieren.
- Wenn man für die einzelnen Verfahren auch die Kosten in Abhängigkeit der einzelnen Parameter kennt, ist es möglich, für bestimmte geologische Situationen eine umfassende Risikobewertung zu erstellen, aus der ebenso optimale Mess-Methoden und -Kampagnen abgeleitet werden können.
- Am Ende müssen die Resultate in eine Software verpackt werden, die auch schon in der Konzeptionsphase für die Kosten-Risikoabschätzung von Investoren genutzt werden kann.

Durch dieses Vorgehen wird es möglich sein, schon in der Planungsphase die verschiedenen Verfahren gegeneinander abzuwägen, und die Risiken und Chancen statistisch sauber zu bewerten.

Für die Entwicklung eines Tools zur Risikoabschätzung von Prospektionsverfahren wurde ein Python-Programm geschrieben. Dieses dient zur Automatisierung der benötigten Schritte, also

- Generierung von synthetischen Untergrund-Modellen
- Vorwärts-Simulationen mit verschiedenen physikalischen Effekten
- Auswerten von virtuellen Sensordaten
- Einbinden von Prospektions- bzw. Inversionsverfahren
- Evaluation und Bewertung

Das Programm ist so konzipiert, dass möglichst einfach verschiedene externe Tools eingebunden werden können, wie zum Beispiel die weiter unten erwähnten Methoden.

In der Geothermie geht es darum, die Energie, die im warmen Erdinnern steckt, für die Stromerzeugung oder für das Heizen zu verwenden. In der Schweiz wird die tiefe Geothermie (Tiefen bis 10 km) eingesetzt, um grössere Quartiere mit Fernwärme zu bedienen. Gemeinsam mit der Hochschule Freiberg und der CBM GmbH in Bexbach - und finanziert durch das Bundesamt für Energie - arbeitet das Institut für Computational Engineering ICE daran, das Fündigkeitsrisiko bei der Erschliessung eines Geothermiegebietes zu reduzieren. Dazu werden die Verfahren, die zum Aufspüren des Erdinneren zum Einsatz kommen (Bohrlöcher, Seismik, Gravimetrie, elektromagnetische Methoden), im Detail mit physikalischen Simulationen sowie Methoden der Data Sciences analysiert. Als Ziel des Projektes soll bereits vor der der Untersuchung eines neuen Geothermiegebietes quantifiziert werden können, welche Kombination von Prospektionsmethoden zu einem minimalen Erkundungsrisiko (bei gegebenen Kosten) führt. Wir haben die in [2] und [4] beschriebenen grundlegenden Erkenntnisse für die Anwendung von Prospektionsmethoden für tiefe geothermische Anwendungen herangezogen. In enger Zusammenarbeit mit Swisstopo, der Uni Genf und der ETH haben wir die Entwicklung der Methodik diskutiert und ihre Arbeiten, z.B. [1], [5], [6], [7] und [9] besprochen.

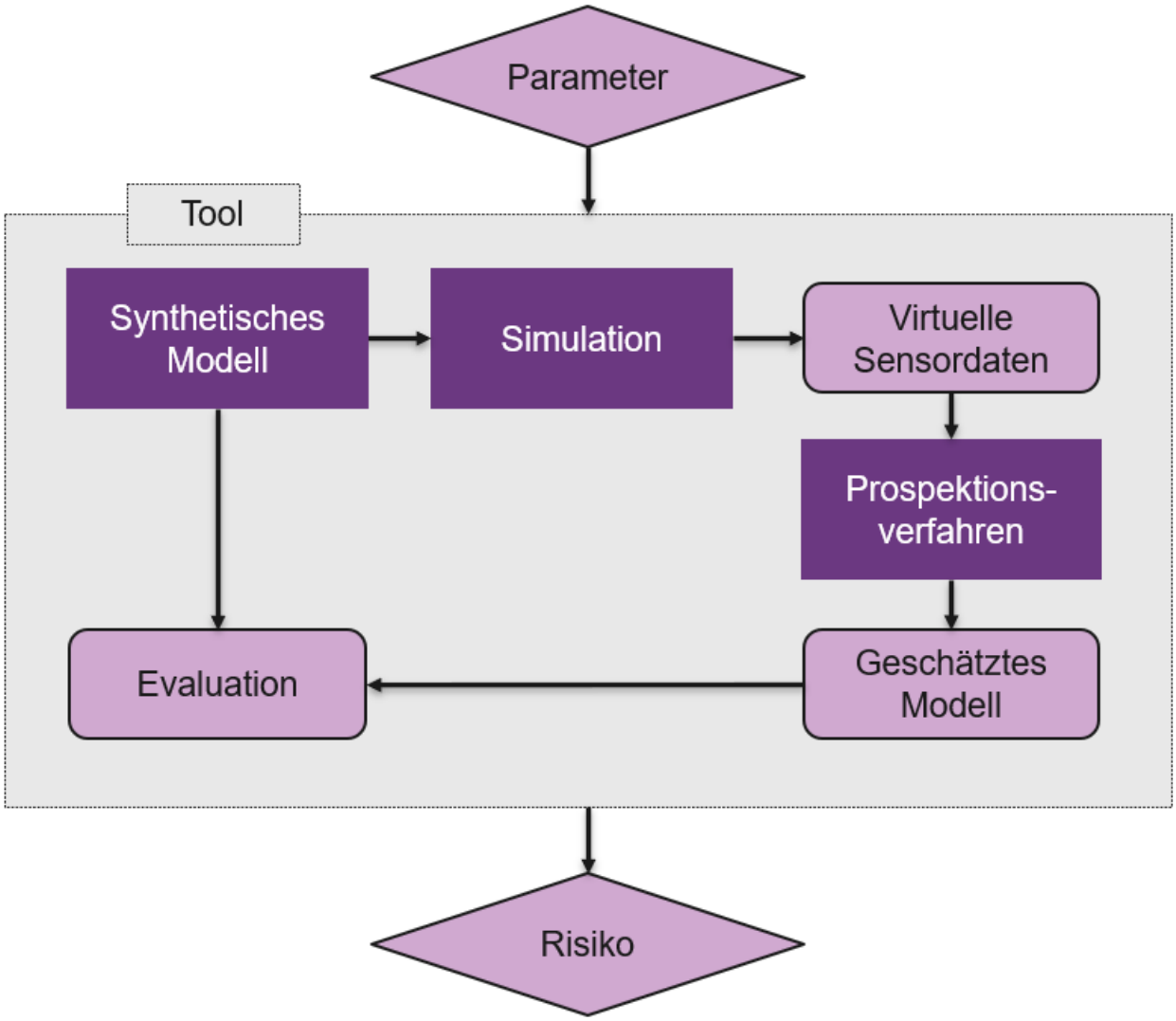

Abb. 1.1: Diagramm der Tool-Struktur.

KAPITEL 2

Aufbau geologische Modelle

„Wofür die Natur Millionen von Jahren brauchte, brauchen wir nur ein paar Sekunden.“

Das Ziel in diesem Teil war die Erstellung von parametrisierbaren geologischen 3D-Modellen für die Anwendung von geophysikalischen Methoden.

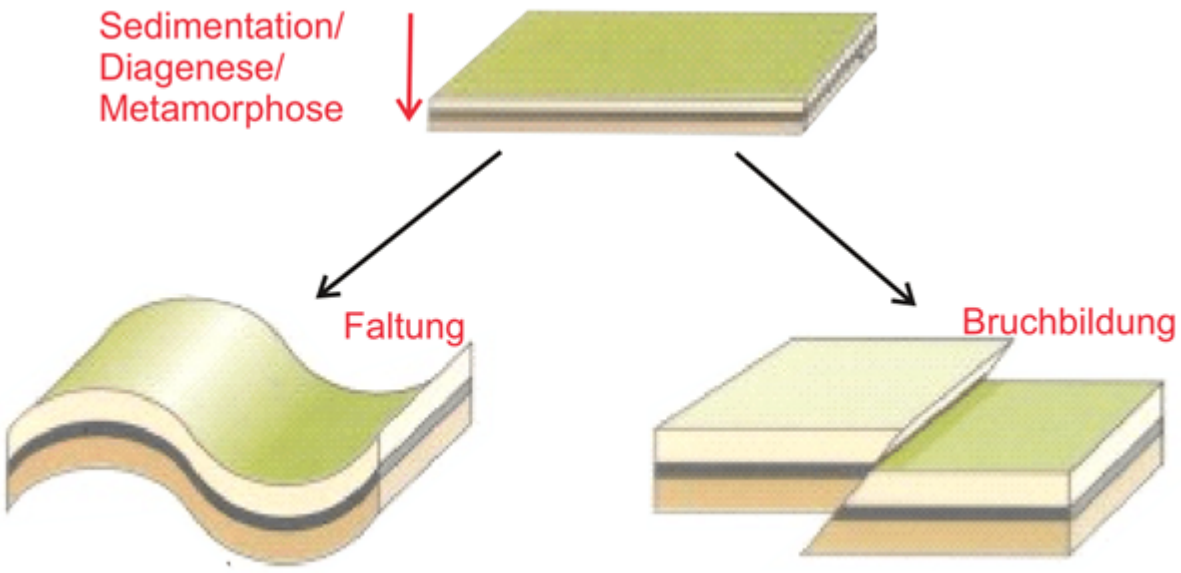

Abb. 2.1: Schema der Entstehung von geologischen Schichten und Verwerfungen.

Für den Aufbau geologischer 3D Modelle haben wir zwei verschiedene Modellgeneratoren implementiert, die leicht unterschiedliche Ziele verfolgen. Auf der einen Seite haben wir den „Flat Layer Generator (FLG)“, der wie der Name schon andeutet, für simple Modelle verwendet werden kann, mit flachen Schichten und Verwerfungen mit beliebiger geometrischer Anordnung (siehe Abbildung 2 links). Der Vorteil des FLGs ist, dass er unabhängig von externen Tools ist und dass die Modelle sehr schnell und einfach erzeugt werden können.

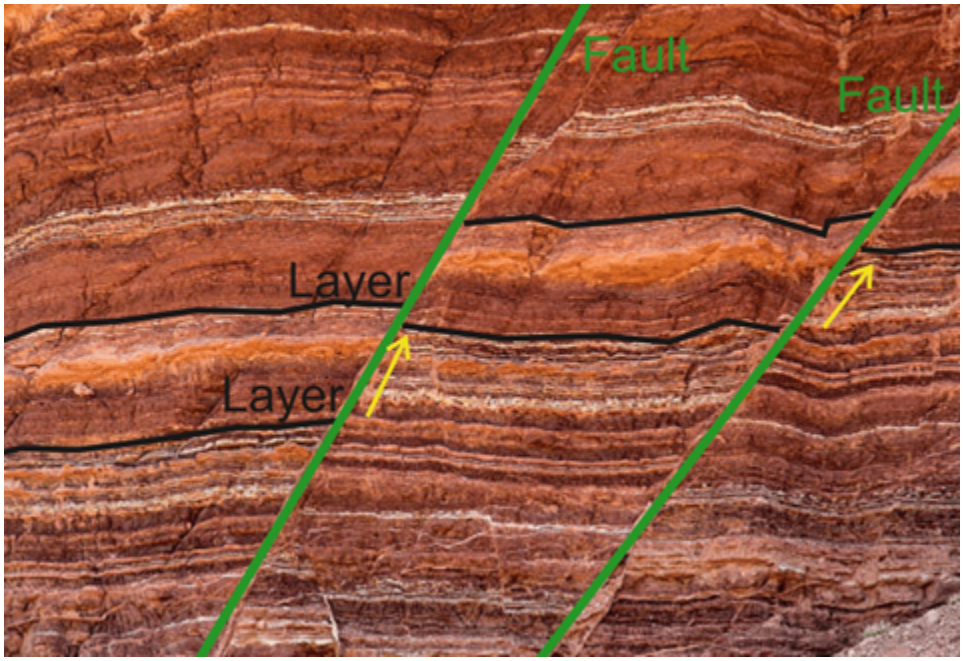

Abb. 2.2: Schnitt durch tektonisch gestörte geologische Schichtfolge.

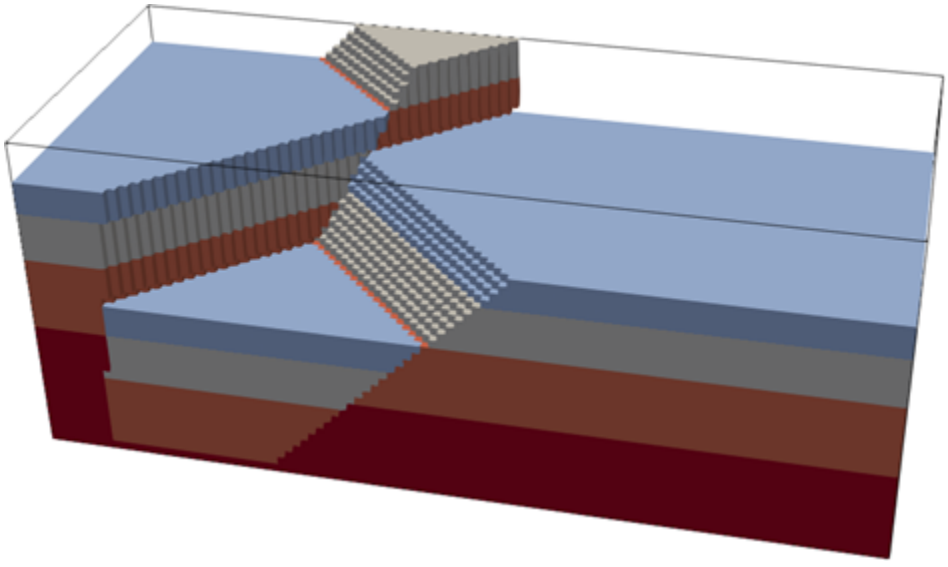

Abb. 2.3: Beispielmodell: Flat Layer Generator

Auf der anderen Seite haben wir den „GemPy Generator (GPG)“, der eine beliebig komplizierte Schichtung und Bruchgeometrie erlaubt. Dazu verwenden wir das externe Python-Paket Gempy [8]. GemPy ermöglicht geologische Modelle durch Interpolation von Punkten an Schichtgrenzen und Verwerfungsstellen zu erstellen. Wir haben diese Bibliothek für die Zwecke unseres Tools mit einer ähnlichen Parametrisierung wie beim ersten Generator erweitert. Wir verändern das ursprüngliche GemPy-Modell, indem wir seine Startpunkte verschieben und drehen.

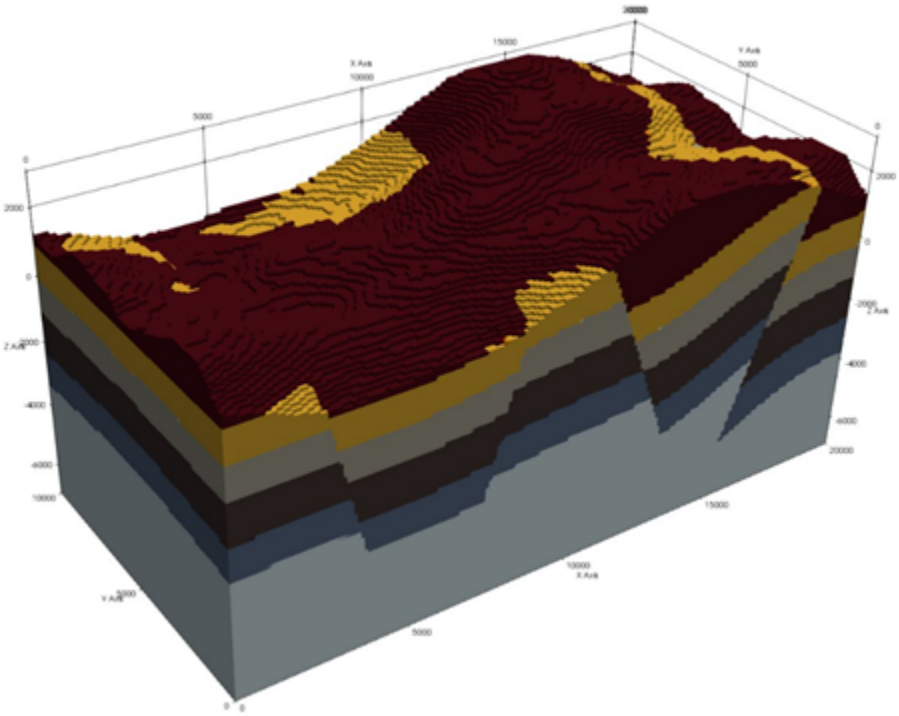

Abb. 2.4: Beispielmodell: GemPy Generator

Die modellbasierte Parametrisierung ermöglicht es uns, eine Reihe von Variationen des Ausgangsmodells zu erstellen. Auf jedes so erstellte Modell können geophysikalische Methoden zur Simulation von Messungen angewendet werden. Das erlaubt eine sehr einfache Anwendung für den Nutzer, und eine automatisierte Manipulation der Schicht- und Bruch Geometrien während der Programmausführung. Somit können zur Laufzeit Modelle erzeugt und verändert werden, die dann für weitere Anwendungen verwendet werden können. Das Ziel ist eine Sensitivitätsanalyse dieser Methoden. Dieser Vorgang wird in den folgenden Kapiteln ausführlich beschrieben.

2.1 Flat Layer Generator

Das Vorgehen basiert auf einer einfachen Parametrisierung der geologischen Schichten. Die Tiefe und die Neigung dieser Schichten können mit mehreren Parametern definiert werden. Zusätzlich zu den Gesteinsschichten können wir auch Verwerfungen modellieren. Ähnlich wie bei den Schichten geben wir ihre Lage und Neigung an.

Das modellierte geologische Gebiet ist immer rechteckig. Die Länge, Breite und Tiefe werden in den Einstellungen eingegeben.

```
"model_domain": {
    "X": 10000,
    "Y": 4000,
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    "Z": 4000
}
```

Nachfolgend werden die einzelnen geologischen Schichten beschrieben.

```
"layers": [
    {
        "name": "sediment",
        "series": "layer_1",
        "material properties": {
            "rho": 2.4,
            "vs": 3.4,
            "vp": 4.7
        },
        "thickness": 1850
    }
]
```

Schliesslich die einzelnen Verwerfungen, die wir anhand eines einzelnen Punktes auf der Oberfläche und einer Neigung definieren. Die Neigung bestimmt den Azimut und den Neigungswinkel. Die Verschiebung der Schichten an den Seiten der Verwerfung bestimmt der Versatz.

```
"faults": [
    {
        "surface point": [
            4000,
            2000
        ],
        "azimuth angle": 0.0,
        "dip angle": 7.5,
        "offset": 400
    }
]
```

2.2 GemPy Generator

Dieser Generator basiert auf einem externen Tool zur Erzeugung von geologischen 3D-Strukturmodellen in Python. Für weitere Informationen siehe die Projektseite GemPy, [8].

Die Modellerstellung mit GemPy ist in unser Tool integriert. Wir können direkt in der Einstellungsdatei die Punkte angeben, die die Schnittstelle der verschiedenen geologischen Schichten und auch die Verwerfungen definieren. Die einzelnen Schritte können anhand der im Code enthaltenen Beispiele nachvollzogen werden.

```
"site_name": "default",
"input_directory_path": "model",
"model_name": "default",
"extent": [0, 1000, -500, 500, -500, 0],
"resolution": [100, 50, 100],
"quantities": [
    {
        "name": "rho",
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        "values": [0.0, 2.0, 2.6, 3.3]
    }
]
```

Für den Flat Layer Generator können wir das Setup leicht parametrisieren. Wir haben die gleiche Option für das GemPy Modell. Mithilfe der Modifikationseinstellungen können die Punkte jeder Schichtgrenze oder Verwerfung gedreht oder verschoben werden.

```
"modifications": [
    {
        "layer": "Layer_1",
        "translation": [1.0, 0.0, 0.0],
        "rotation": {
            "origin": [0.0, 0.0, 0.0],
            "axis":   [0.0, 0.0, 0.0],
            "angle":  10
        }
    }
]
```

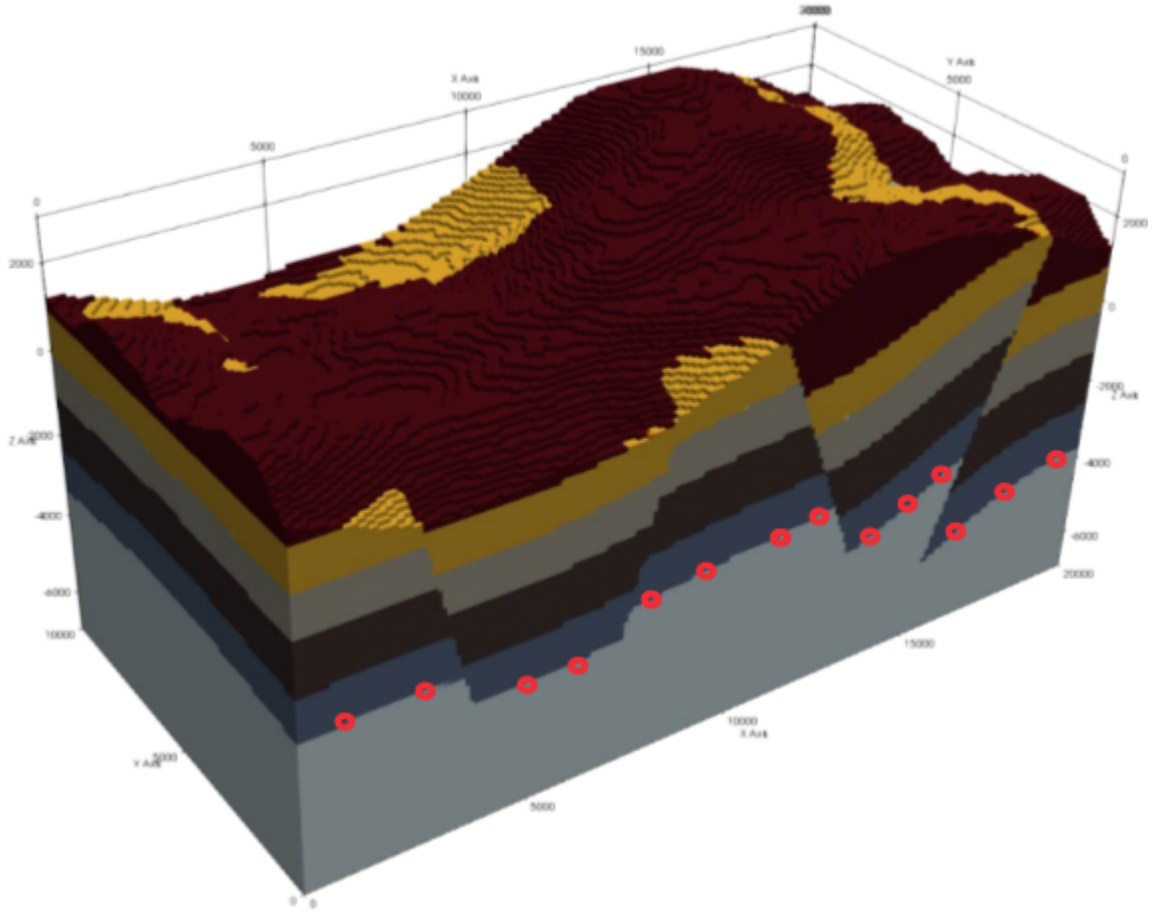

Abb. 2.5: Beispielmodell: GemPy Generator, Punkte an der Schichtgrenze.

KAPITEL 3

Gravimetrie

Wir verwenden externe Programme zur Simulation von Prospektionsverfahren. Wir geben zunächst ein Beispiel für eine gravimetrische Simulation mit SimPEG, [3]. SimPEG bietet eine Sammlung von geophysikalischen Simulations- und Inversionswerkzeugen, die in einem einheitlichen Rahmen aufgebaut sind. Ausser Gravimetrie auch Magnetik oder Magnetotellurik. Ähnlich wie bei SimPEG ist es möglich, unser Tool mit anderen Softwarepaketen zu erweitern. Es ist notwendig, die Simulationseinstellungen in der Software mit den REX-Einstellungen entsprechend zu verknüpfen. Im nächsten Kapitel beschreiben wir die Anknüpfung an die seismische Simulation unter Verwendung der WAVE-Bibliothek.

Das SimPEG-Simulationsverfahren läuft wie folgt ab. Zunächst wird das Ausgangsmodell in Volumen - Voxel - unterteilt, auf denen die geophysikalischen Grössen diskretisiert werden. Derzeit betrachten wir nur ein uniformes Netz und definieren daher seine Auflösung in der Richtung jeder Achse. Als Nächstes geben wir die Messpunkte an, an denen die Gravitationskraft ausgewertet werden soll. Die derzeitige Implementierung erlaubt es uns, ein uniformes Netz zu definieren, d.h. ähnlich wie bei den Volumenelementen legen wir seine Auflösung fest. In den Einstellungen geben wir alles unter dem Stichwort *simulation* ein.

```
"simulation": {
    "problem": "gravity",
    "options": {
        "mesh resolution": [100, 100, 70],
        "mesh padding": 0.0,
        "basement density": 3.3,
        "observation points": {
            "extent": [1000., 9000., 1000., 9000.],
            "resolution": [18, 18],
            "distance to top": 1001.0
        }
    }
}
```

Das Ergebnis ist die in vertikaler Richtung berechnete Schwerkraft an jedem Punkt und wird im *VTK*-Format gespeichert. Die so gespeicherten Ergebnisse können mit der ParaView-Software angezeigt oder mit vordefinierten Funktionen automatisch verarbeitet werden. Die automatische Verarbeitung ist besonders bei einer Reihe von Berechnungen sinnvoll und kann in den Einstellungen unter dem Stichwort *evaluation* angegeben werden.

```
"evaluation": [
    {
        "method": "plot gravity",
        "options": {}
    },
    {
        "method": "visualize density",
        "options": {}
    }
]
```

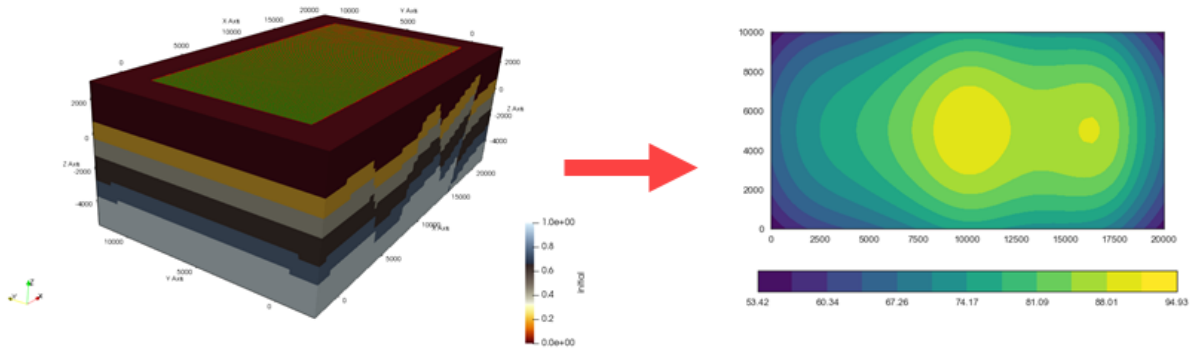

Abb. 3.1: Visualisierung der Simulation der gravimetrischen Prospektion an einem einfachen Beispiel.

KAPITEL 4

Seismik

Zur Simulation von seismischen Wellen verwenden wir den Open-Source-Löser *WAVE*, eine Entwicklung des KIT in Zusammenarbeit mit mehreren Universitäts-Instituten und einer deutschen Firma für Kohlenwasserstoff-Exploration.

Zur Einbindung von *WAVE* in das angestrebte Verfahren im REX Projekt wurde analog zum Vorgehen in der Gravimetrie ein Wrapper geschrieben, wodurch auch für die Seismik automatisiert Simulationen für einfache und komplexe geologische Modelle durchgeführt werden können.

Die Einstellungen des *WAVE*-Solvers geben wir auch unter dem Stichwort *simulation* ein. Wie bei den gravimetrischen Simulation wird das Ausgangsmodell in Volumen - Voxel - unterteilt.

```
"simulation": {
    "problem": "seismic",
    "options": {
        "wave_options": {
            "T": 3,
            "DT": 3e-3,
            "dimension": "3D",
            "equationType": "elastic",
            "BoundaryWidth": 5,
            "n_proc": 4,
            "NX": 100,
            "NY": 40,
            "NZ": 40,
            "saveSnapshots": 0,
            "tincSnapshot": 2e-1,
            "tlastSnapshot": 2,
            "tFirstSnapshot": 0,
        },
    }
}
```

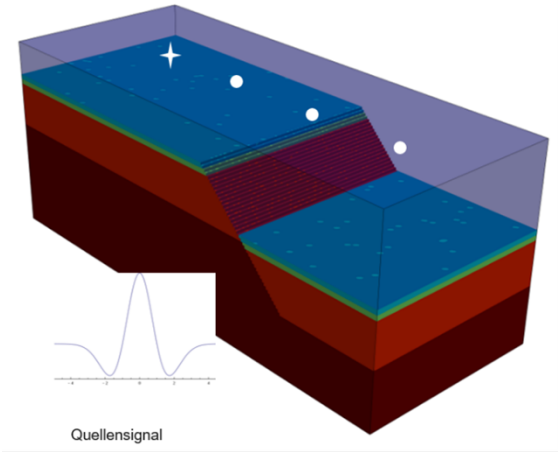

Abb. 4.1: Simuliertes Gebiet der Seismik-Simulation. Der Stern bezeichnet die Quellenposition, die Punkte die virtuellen Seismographen. An der Quelle wird ein Rickerwavelet als Signal vorgegeben.

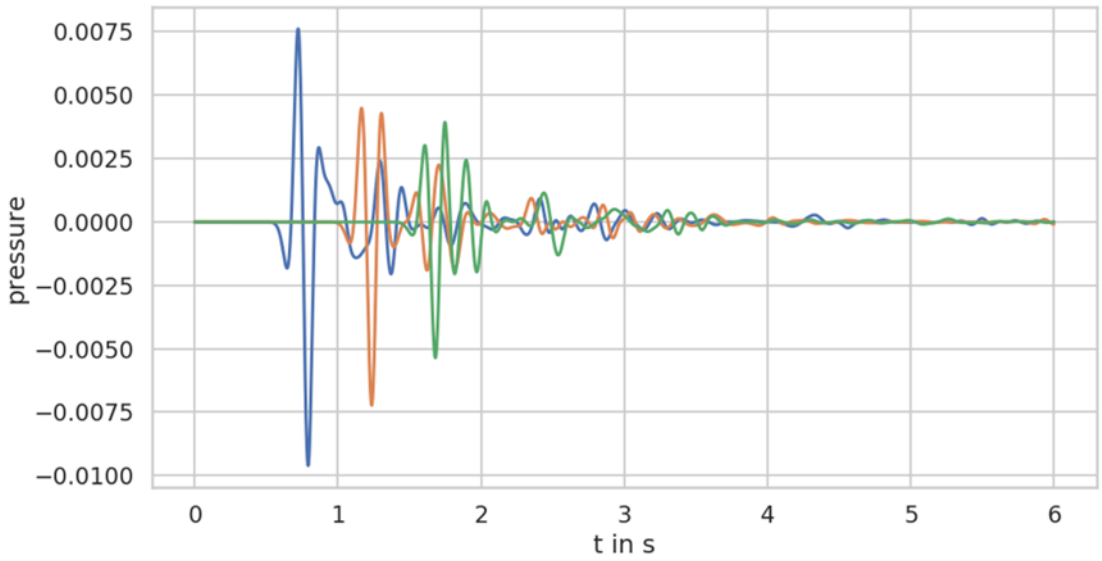

Abb. 4.2: Zeitverlauf des gemessenen Drucks an den drei virtuellen Seismographen.

KAPITEL 5

Sensitivitätsanalyse

Unsere Methode basiert auf der Parametrisierung der komplexen Topologie der untersuchten Gebiete und einer anschliessenden Sensitivitätsanalyse. Damit soll die Möglichkeit der Quantifizierung von Prospektionsmethoden aufgezeigt werden. Diese Methoden führen zu inversen Problemen. Wir gehen von zwei Grundsätzen aus. Erstens eine ausreichend komplexe, aber vereinfachte Parametrisierung des Untersuchungsgebiets. Zweitens die bestmögliche Schätzung der Genauigkeit der Prospektionsmethode.

Wir fragen, welche Abweichungen von den Untergrundbedingungen zu welchen Veränderungen in den Messdaten beitragen. Bei der Untersuchung des Modells führen wir eine Trennung zwischen Studienparametern und Störungsparametern ein. Ausserdem stützen wir uns bei der Schätzung der Varianz der Studienparameter auf eine geeignete Linearisierung der Prospektionsmethoden.

Auf der Grundlage der Parametrisierung können die Prospektionsmethoden als Parameterfunktionen verallgemeinert werden. Wir ordnen jedem Parametersatz die gemessenen Werte an den Receivern zu. Prospektionsmethoden befassen sich mit der Frage, wie die Parameter auf der Grundlage der Messergebnisse bestimmt werden können. Uns geht es jedoch um eine noch allgemeinere Frage. Wir wollen im Voraus bestimmen, wie genau die Parameterschätzungen auf der Grundlage der Messungen sein können.

Um diese Frage zu beantworten, müssen wir das Problem zunächst angemessen linearisieren. Anschliessend werden alle Unsicherheiten des Modells bestimmt und im letzten Schritt wird das so spezifizierte Problem mit der Methode der verallgemeinerten kleinsten Quadrate analysiert. Wir beschreiben die Schritte anhand einer mathematischen Formulierung.

5.1 Schritt 1: Linearisierung der Prospektionsmethoden.

Bezeichnen wir die Studienparameter $\mathbf{p} \in \mathbb{R}^s$ und Störgrössen $\mathbf{d} \in \mathbb{R}^t$. Weiter drücken wir die Receiverwerte $\mathbf{r}^* \in \mathbb{R}^n$ als die Abbildung von $\mathbf{d}$, und $\mathbf{p}$ aus,

$$\mathbf{r}^* = \mathbf{r}^*(\mathbf{d}, \mathbf{p}).$$

Wir gehen weiter davon aus, dass diese Beziehung linearisiert werden kann und $\mathbf{p}$ als Lösung des inversen problems in Abhängigkeit von $\mathbf{r}^*$ und $\mathbf{d}$ ausgedrückt werden kann. Wir stellen die lineare Abhängigkeit der Receiverwerte von

den Studienparametern und der Störgrössen fest.

$$\mathbf{r}^* = \mathbb{A} \cdot \mathbf{d} + \mathbb{B} \cdot \mathbf{p},$$

wo $\mathbb{A}$ und $\mathbb{B}$ die Koeffizientenmatrizen sind. Sie werden mit dem automatischen Simulationswerkzeug auf der Grundlage statistischer Methoden ermittelt.

5.2 Schritt 2: Quantifizierung der Modellunsicherheit.

Die Unsicherheiten im Modell sind nicht nur auf diese Störgrößen zurückzuführen, sondern auch auf die Ungenauigkeit der Messgeräte. Bezeichnen wir den Messfehler mit $\mathbf{c} \in \mathbb{R}^n$. Wir wollen die Abweichung der Messdaten quantifizieren, die sowohl durch die Abweichung der Störparameter als auch durch den Messfehler verursacht wird.

Im vorherigen Schritt haben wir Messfehler ignoriert und idialisierte Receiverwerte mit einem Stern markiert. Definieren wir nun Receiverwerte $\mathbf{r} \in \mathbb{R}^n$ auch unter Berücksichtigung von Messfehlern.

$$\mathbf{r} = \mathbb{A} \cdot \mathbf{d} + \mathbb{B} \cdot \mathbf{p} + \mathbf{c},$$

Die Vektoren $\mathbf{c}$, $\mathbf{d}$, $\mathbf{p}$, und $\mathbf{r}$, stellen Zufallsvektoren mit einem Normalverteilungsmodell dar:

$$\begin{aligned}
\mathbf{c} &\sim (\mathbf{0}, \mathbf{\Sigma_c}), \qquad \mathbf{\Sigma_c} = \mathrm{Cov}(\mathbf{c}), \\
\mathbf{d} &\sim (\mathbf{0}, \mathbf{\Sigma_d}), \qquad \mathbf{\Sigma_d} = \mathrm{Cov}(\mathbf{d}), \\
\mathbf{p} &\sim (\mu_\mathbf{p}, \mathbf{\Sigma_p}), \quad \mathbf{\Sigma_p} = \mathrm{Cov}(\mathbf{p}), \\
\mathbf{r} &\sim (\mu_\mathbf{r}, \mathbf{\Sigma_r}), \qquad \mathbf{\Sigma_r} = \mathrm{Cov}(\mathbf{r}).
\end{aligned}$$

Es ist zu beachten, dass die Störgrössen $\mathbf{d}$ so gewählt werden müssen, dass sie einen Mittelwert von 0 haben. Unter Verwendung der hier angegebenen Notation kann das Problem wie folgt formuliert werden. Wir wollen die bestmögliche Schätzung der Genauigkeit des Parameters $\mathbf{p}$ erhalten, die sich durch ein Optimierungsproblem auf der Grundlage der Daten $\mathbf{r}$ erzielen lässt.

Wir definieren den totalen Receiverfehler $\mathbf{e}$

$$\mathbf{e} = \mathbb{A} \cdot \mathbf{d} + \mathbf{c}.$$

Unter der Annahme, dass $\mathbf{c}$ und $\mathbf{d}$ unkorreliert sind, können wir die Receiverfehlerkovarianz durch die Messwertkovarianz und den Beitrag der Störgrössen ausdrücken.

$$\mathbf{\Sigma_e} = \mathrm{Cov}(\mathbb{A} \cdot \mathbf{d}) + \mathrm{Cov}(\mathbf{c}) = \mathbb{A} \cdot \mathbf{\Sigma_d} \cdot \mathbb{A}^T + \mathbf{\Sigma_c}.$$

5.3 Schritt 3: Parameter-Schätzer.

Im letzten Schritt bestimmen wir, wie genau wir die Studienparameter schätzen können. Zu diesem Zweck verwenden wir die Methode der verallgemeinerten kleinsten Quadrate (GLS). Schreiben wir das Ausgangsproblem um in der Form

$$\mathbf{r} = \mathbb{B} \cdot \mathbf{p} + \mathbf{e}$$

$\mathbf{r} \in \mathbb{R}^n$, Empfänger-Sensorwerte, Antwortwerte, $\mathbf{p} \in \mathbb{R}^s$, Studienparameter, Vektor der Unbekannten, $\mathbb{B} \in \mathbb{R}^{n \times s}$, durch die Linearisierung erhaltene Modellmatrix, voller Rang, $\mathbf{e} \in \mathbb{R}^n$, Fehler mit einer bekannten regelmässigen Kovarianzmatrix $\mathbf{\Sigma_e}$.

GLS schätzt $\mathbf{p}$ durch Minimierung der verallgemeinerten quadratischen Länge des Restvektors

$$\mathbf{p}_{est} = \mathrm{argmin}_\mathbf{x} \left(\mathbf{r} - \mathbb{B}\mathbf{x}\right)^T \mathbf{\Sigma_e}^{-1} \left(\mathbf{r} - \mathbb{B}\mathbf{x}\right).$$

Wir sind jedoch an den Eigenschaften dieser Schätzung interessiert. GLS ist der beste lineare unbiased Schätzer.

$$E\left[\mathbf{p}_{est}\middle|\mathbb{B}\right] = \mathbf{p}$$

Kovarianz des GLS-Schätzers

$$\begin{aligned}\mathbf{\Sigma}_{\text{est}} &= \left(\mathbb{B}^T {\mathbf{\Sigma}_{\mathbf{e}}}^{-1} \mathbb{B}\right)^{-1}, \\ \mathbf{p}_{\text{est}} &\sim \mathcal{N}\left(\mathbf{p}, \left(\mathbb{B}^T {\mathbf{\Sigma}_{\mathbf{e}}}^{-1} \mathbb{B}\right)^{-1}\right).\end{aligned}$$

Die Kovarianz $\mathbf{\Sigma}_{\text{est}}$ ist das gesuchte Ergebnis. In der Regel wird dann die Quadratwurzel aus den Elementen auf der Diagonale gebildet. Es handelt sich also um die Standardabweichung der geschätzten Parameter. Somit lautet unsere Behauptung, dass die gewählte Prospektivmethode keine bessere Schätzung der untersuchten Parameter liefern kann. Wir möchten hinzufügen, dass diese Behauptung auf der Genauigkeit der Linearisierung basiert. In der Praxis hängt die Linearisierung von der Wahl des Ausgangsmodells ab. Darüber hinaus ist auf die Bedeutung einer komplexen Parametrisierung hinzuweisen. Wenn wir das Problem zu stark vereinfachen und nicht alle relevanten Störgrössen einbeziehen, wird das Ergebnis eine zu optimistische Schätzung sein.

Wir illustrieren die Parametrisierung und Schätzung anhand eines synthetischen Modells, das auf dem Gebiet im Aargau basiert. An diesem Beispiel zeigen wir die Anwendung an einer gravimetrischen und seismischen Studie. Wir untersuchen die Möglichkeiten der Platzierung von Receivern, die Analyse des Bohrlochentwurfs oder die Kombination von Prospektionsmethoden der Gravimetrie und Seismik. Wir müssen die Parameter auswählen, die für die Prospektionsmethoden von Bedeutung sind. In diesem Gebiet sind die Lage und Tiefe des Grabens von Interesse, ob die Veränderung der Grabengeometrie gravimetrisch bzw. seismisch nachgewiesen werden kann. Die Daten stammen aus der geologischen Datenbank von swisstopo und wurden durch eine Modellierung des Untergrundes aus 4 Schichten vereinfacht: Muschelkalk, Effinger, Grabenfüllung und Grundgebirge. Jede dieser Schichten hat unterschiedliche physikalische Eigenschaften. Für die Gravimetrie ist die Dichte von Interesse, für die Seismik wird es die Wellengeschwindigkeit sein. Parameter, die die Geometrie des Grabens charakterisieren:

- Tiefe des Grabens - Versatz durch Verwerfungen,
- Verwerfung A, Lage,
- Verwerfung A, Neigungswinkel,
- Verwerfung B, Lage,
- Verwerfung B, Neigungswinkel.

Wie bereits erwähnt, stellen wir die Frage, wie genau wir diese Parameter schätzen können. Bei dieser Schätzung müssen die Störgrössen berücksichtigt werden, die die Messwerte beeinflussen. Das sind

- die Dichte der 4 Schichten,
- der Versatz der Schichten.

Wenn nicht anders angegeben, wird eine Standardabweichung der Dichte von $0.05 g.cm^{-3}$ und im Falle des Grundgebirges von $0.1 g.cm^{-3}$ angenommen. Die Standardabweichung der Schichtpositionen beträgt 100 m. Schliesslich beträgt die Standardabweichung der Messungen 0.1 mGal. Die Diskretisierung des Untergrunds erfolgte auf einem quadratischen Gebiet von 17 x 8 x 6 km, die durch 170 x 5 x 120 Blockelemente unterteilt wurde

Die Tabellen 1-4 enthalten eine Zusammenfassung der Ergebnisse auf der Grundlage synthetischer gravimetrischer und - in der letzten Tabelle - seismischer Untersuchungen. In jeder Tabelle sind die Standardabweichungen (STD) $\sigma_p = \sqrt{\text{diag}(\mathbf{\Sigma}_{\text{est}})}$ der untersuchten Parameter angegeben. Wie bereits erwähnt, handelt es sich hierbei um die erwarteten Abweichungen, die in dem besten Fall geschätzt werden können. Die erste Tabelle zeigt die Ergebnisse der gravimetrischen Studie, wenn wir 120 Empfänger gleichmässig in drei Reihen auf der Oberfläche verteilen. Die erste Spalte ist eine Schätzung der Unsicherheiten bei Vernachlässigung des Einflusses der Störgrössen. In der zweiten Spalte wird ihr Einfluss berücksichtigt. In Tabelle 2 sehen wir, ob sich eine ähnlich genaue Bestimmung der Parameter mit weniger Messgeräten erreichen lässt. Wir haben den einheitlichen Abstand zwischen den Receivern allmählich vergrössert. Resultat: Noch mit 60 Receivern erhalten wir eine ähnliche Schätzung wie mit 120 Empfängern. Mit weniger Messgeräten nicht mehr. Im dritten Fall vergleichen wir zwei Studien, wobei die Unsicherheit der Störgrössen

in der zweiten Studie geringer ist. In der Praxis würden wir zum Beispiel mit Slimhole-Bohrungen weitere Informationen gewinnen. Darum geben wir in der Studie 2 Standardabweichung der Dichte von $0.01g.cm^{-3}$ und im Falle des Grundgebirges von $0.02g.cm^{-3}$ ein. Die Standardabweichung des Schichtenversatzes beträgt 50 m. Die Schätzung der Studienparameter verbesserte sich fast um den Faktor 2. Diese Informationen können als Basisdaten für die Entscheidungsfindung bei der Bestimmung der Bohrkosten und des Wertes von Informationen dienen. Die letzte Tabelle ist ein Vergleich der Gravimetrie und der Seismik. In der letzten Spalte sehen wir den möglichen Beitrag der Gravimetrie zur Verbesserung der Parameterschätzung aus der Seismik bei einer Fusionsabschätzung. Um den möglichen Einfluss zu zeigen, wählen wir für diesen Fall eine eher ungenaue seismische Messung mit nur 9 Empfängern. Die Schätzung der Untersuchungsparameter durch Seismik ist sehr präzise. Damit verbessert die Fusion der Methoden die Schätzung der Studienparameter erheblich. Diese Beispiele sind eher illustrativ. Wir gehen davon aus, dass wir in der Praxis auf Fälle stossen werden, in denen selbst bei einem einfacheren Modell der Nutzen von Prospektionsmethoden zu gering sein könnte.

Tab. 5.1: Vergleich der Auswirkungen von Störgrössen auf die Schätzung der Abweichung der untersuchten Parameter.

STD σ_p	#1 nur Messdaten-Abweichungen	#2 zusammen mit Störgrössen
Dip A [°]	0.830	5.108
Dip B [°]	1.153	3.388
Versatz [m]	16.70	450.91
Lage A [m]	35.04	233.61
Lage B [m]	48.71	147.25

Tab. 5.2: Vergleich für unterschiedliche Anzahl von gravimetrischen Receivern.

# Receiver / DX [m]	# 120 / 385 m	# 60 / 770 m	# 30 / 1540 m	# 15 / 3080 m
Dip A [°]	5.108	6.565	8.311	22.278
Dip B [°]	3.388	4.462	8.351	13.701
Versatz [m]	450.91	577.18	699.15	1518.15
Lage A [m]	233.61	304.62	381.21	1021.94
Lage B [m]	147.25	196.31	374.38	661.06

Tab. 5.3: Vergleich zweier Studien, wobei im zweiten Fall geringere Unsicherheiten bei den Störgrössen bestehen.

STD σ_p	Studie #1	Studie #2
Dip A [°]	5.108	3.333
Dip B [°]	3.388	2.671
Versatz [m]	450.91	255.18
Lage A [m]	233.61	155.68
Lage B [m]	147.25	123.36

Tab. 5.4: Vergleich der Unsicherheiten von gravimetrischen Studien, seismischen Studien und gekoppelten Studien.

STD σ_p	Gravimetrie	Seismik	Fusion
Dip A [°]	5.108	4.718	0.161
Dip B [°]	3.388	2.603	0.088
Versatz [m]	450.91	168.80	5.439
Lage A [m]	233.61	354.43	5.174
Lage B [m]	147.25	82.26	1.823

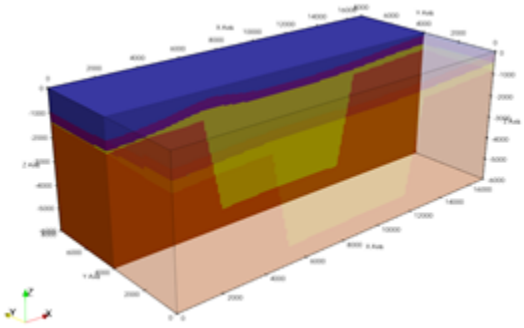

Abb. 5.1: Eine Ansicht des Untersuchungsgebiets.

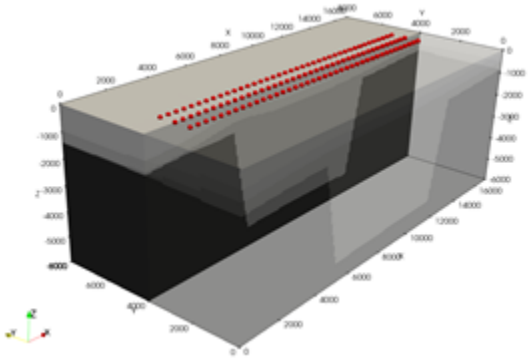

Abb. 5.2: Beispiel für die Platzierung von 120 Receivern.

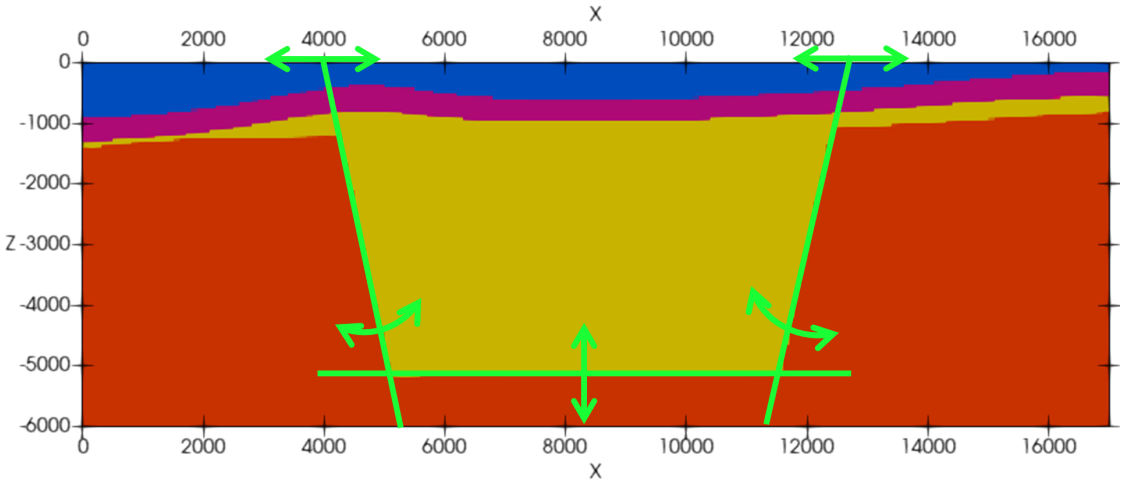

Abb. 5.3: Schnittfläche mit den Geometrieparametern des Grabens.

KAPITEL 6

REX - Reliable Prospection and Exploration Tool

Dieses Tool verfügt weder über eine grafische Oberfläche noch über ein Installationsprogramm wie im Falle von RAT. Wir demonstrieren die Verwendung der Bibliothek direkt in Python mit Hilfe von Skripten, die zur Durchführung einzelner Simulationen oder sogar komplexer parametrischer Studien verwendet werden können.

6.1 Installation

Um die Bibliothek zu verwenden, ist Python Version 3.8 oder höher erforderlich, und Sie müssen die notwendigen Bibliotheken installieren, die sich in der Datei *requirements.txt* befinden. Die Installation kann mit dem Pip-Installer durchgeführt werden.

```
pip install -r requirements.txt
```

6.2 Studienvorbereitung

Im einfachsten Fall, wenn wir nur eine Simulation durchführen wollen, würde das Skript wie folgt aussehen.

```
from prospection import Prospection

OPTIONS = {
    "working dir": "example",
    "model": {
        "generator": "gempy_model_load",
        "options": {
            "site_name": "example name",
            "input_directory_path": "model_path",
            "model_name": "model",
            "extent": [0., 10000., 0., 10000., -6000., 1000.],
            "resolution": [100, 100, 70],
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
            "quantities": [
                {
                    "name": "rho",
                    "values": [
                        0.0,
                        2.1,
                        2.2,
                        2.3,
                        3.3
                    ]
                },
            ]
        }
    },
    "simulation": {
        "problem": "gravity",
        "options": {
            "mesh resolution": [100, 100, 70],
            "mesh padding": 0.0,
            "basement density": 3.3,
            "observation points": {
                "extent": [1000., 9000., 1000., 9000.],
                "resolution": [18, 18],
                "distance to top": 1001.0
            }
        }
    },
    "evaluation": [
        {
            "method": "plot gravity",
            "options": {}
        },
        {
            "method": "visualize density",
            "options": {}
        },
        {
            "method": "visualize density slice",
            "options": {}
        }
    ]
}

case = Prospection(OPTIONS)
case.run()
```

Indem wir den Fall Optionen ändern, können wir mit Python for-Schleifen eine Reihe von Berechnungen für den Fall erstellen.

6.3 Evaluation

In den Kapiteln, die den prospektiven Methoden gewidmet sind, haben wir auch die Bewertung beschrieben. Die Ergebnisse werden in der Regel im *VTK*- oder *CSV*-Format erzeugt und können mit Hilfe von Skripten zur weiteren Auswertung verwendet werden. Zur Weiterverarbeitung des Ergebnisses verwenden wir in der Regel die Auswertungsmethode *parameter*.

```
{
    "method": "parameter",
    "options": {
        "parameter": [
            f"rho_upper_filling {rho_upper_filling:.3f}",
            f"offset_upper_filling {offset_upper_filling:.3f}",
            f"fault2_dip_change {fault2_angle:.3f}",
        ]
    }
}
```

KAPITEL 7

Beispiele

Im vorangegangenen Text haben wir anhand einzelner Beispiele gezeigt, wie man die *rex*-Bibliothek verwendet. Die Grundidee besteht darin, eine *json*-Datei oder ein entsprechendes Python-Wörterbuch mit allen Einstellungen zu erstellen, um das Modell zu generieren, die Berechnung durchzuführen und auszuwerten. Mit dieser Methode können die Einstellungen parametrisiert werden. In diesem Abschnitt zeigen wir anhand einiger Python-Skripte, wie man eine Reihe von Berechnungen durchführt und sie mit Hilfe der in Abschnitt 5 beschriebene Sensitivitätsanalyse verarbeitet.

7.1 Modellbildung und Berechnungsreihen

Dieses Beispiel zeigt, wie wir eine Reihe von Berechnungen erstellen. Wir führen eine Variation der Parameter des verwendeten Ausgangsmodells durch.

Im Header der Datei importieren wir das Objekt *Prospektion* aus der Bibliothek *rex*. Als nächstes werden einige Standard-Python-Bibliotheken verwendet. Wir importieren *pathlib* zur Arbeit mit Dateien und *itertools* zur Erzeugung von Variationen von Parametern.

```
"""
Example: Synthetic model
========================
"""
import pathlib
from itertools import product

from prospection import Prospection
```

Im folgenden Schritt definieren wir das Default-Wörterbuch mit Einstellungsdaten.

```
OPTIONS = {
    "working dir": "example/synthetic_model",
    "model": {
        "generator": "gempy_model_load",
        "options": {
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
            'site_name': "synthetic_model",
            'input_directory_path':
                'example/model/base_model',
            'model_name': "synthetic_model",
            'extent': [0., 28000., 0., 14000., -6500., 1000.],
            'resolution': [100, 50, 60],
            'quantities': [
                {
                    'name': "rho",
                    'values': [
                        0.0,
                        0.0,
                        0.0,
                        0.0,
                        0.0,
                        2.1,
                        2.2,
                        2.3,
                        2.4,
                        2.5,
                        2.6,
                        3.3
                    ]
                },
            ]
        }
    },
    "simulation": {
        "problem": "gravity",
        "options": {
            'mesh resolution': [100, 50, 60],
            'mesh padding': 0.0,
            'basement density': 2.9,
            'observation points': {
                "extent": [1000., 27000., 1000., 13000.],
                'resolution': [27, 13],
                'distance to top': 1001.0
            }
        }
    },
    'evaluation': [
        {
            'method': 'plot gravity',
            'options': {}
        },
        {
            'method': 'visualize density',
            'options': {}
        },
        {
            'method': 'visualize density slice',
            'options': {}
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        }
    ]
}
```

Dann kommt die eigentliche Steuerungsfunktion. Die Eingabe sind die Standardeinstellungen, das Zielverzeichnis und ein Satz von Parametern.

```
def run(case_options, results_path, parameter_set):
    i = 0
    for (
        rho_pink,
        rho_orange,
        rho_unconformity,
        rho_upper_filling,
        rho_lower_filling,
        offset_upper_filling,
        offset_lower_filling,
        fault2_dip,
        fault2_azimuth,
        fault5_dip,
        fault5_azimuth,
        fault6_dip,
        fault6_azimuth,
    ) in parameter_set:
        i += 1
        working_dir = (
            f"{results_path}/case/"
            f"sample_{i:05d}"
        )

        if pathlib.Path(working_dir).exists():
            continue

        case_options['working dir'] = working_dir

        model_options = case_options['model']['options']
        model_options['quantities'] = [
            {
                'name': "rho",
                'values': [
                    0.0,
                    0.0,
                    0.0,
                    0.0,
                    0.0,
                    2.4,
                    rho_pink,
                    rho_orange,
                    rho_unconformity,
                    rho_upper_filling,
                    rho_lower_filling,
                    2.67
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
            ]
        },
    ]
    model_options['modifications'] = [
        {
            'layer': 'Lower-filling',
            'translation': [0.0, 0.0, offset_upper_filling],
        },
        {
            'layer': 'Upper-filling',
            'translation': [0.0, 0.0, offset_lower_filling],
        },
        {
            'layer': 'Fault2',
            'rotation': {
                'origin': [8816.5713501, 1260., 937.5],
                'axis':   [1.0, 0.0, 0.0],
                'angle':  fault2_dip
            }
        },
        {
            'layer': 'Fault2',
            'rotation': {
                'origin': [8816.5713501, 1260., 937.5],
                'axis':   [0.0, 0.0, 1.0],
                'angle':  fault2_azimuth
            }
        },
        {
            'layer': 'Fault5',
            'rotation': {
                'origin': [17323.23495, 4600, -848.5713],
                'axis':   [1.0, 0.0, 0.0],
                'angle':  fault5_dip
            }
        },
        {
            'layer': 'Fault5',
            'rotation': {
                'origin': [17323.23495, 4600, -848.5713],
                'axis':   [0.0, 0.0, 1.0],
                'angle':  fault5_azimuth
            }
        },
        {
            'layer': 'Fault6',
            'rotation': {
                'origin': [20109.50946, 4600, -1019.9999],
                'axis':   [1.0, 0.0, 0.0],
                'angle':  fault6_dip
            }
        },
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
            {
                'layer': 'Fault6',
                'rotation': {
                    'origin': [20109.50946, 4600, -1019.9999],
                    'axis':   [0.0, 0.0, 1.0],
                    'angle':  fault6_azimuth
                }
            },
        ]
```

In dem letzten Teil der *run*-Funktion definieren wir die Auswertungsmethoden. Dabei ist wichtig die *parameter*-Methode, weil sie die verwendete Parameter für die weitere Analyse speichert.

```
case_options['evaluation'] = [
    {
        'method': 'plot gravity',
        'options': {}
    },
    {
        'method': 'visualize density',
        'options': {}
    },
    {
        'method': 'visualize density slice',
        'options': {}
    },
    {
        'method': "parameter",
        'options': {
            'parameter': [
                f"rho_pink {rho_pink:.3f}",
                f"rho_orange {rho_orange:.3f}",
                f"rho_unconformity {rho_unconformity:.3f}",
                f"rho_upper_filling {rho_upper_filling:.3f}",
                f"rho_lower_filling {rho_lower_filling:.3f}",
                f"offset_upper_filling {offset_upper_filling:.3f}",
                f"offset_lower_filling {offset_lower_filling:.3f}",
                f"fault2_dip_change {fault2_dip:.3f}",
                f"fault2_azimuth_change {fault2_azimuth:.3f}",
                f"fault5_dip_change {fault5_dip:.3f}",
                f"fault5_azimuth_change {fault2_azimuth:.3f}",
                f"fault6_dip_change {fault6_dip:.3f}",
                f"fault6_azimuth_change {fault2_azimuth:.3f}",
            ]
        }
    }
]
```

Im Schluss der *run*-Funktion werden einzelne Fälle ausgeführt.

```
case = Prospection(case_options)
case.run()
```

Die Definition von dem Parameter-Satz wird in dem Hauptabschnitt ausgeführt und mit den Ausgangseinstellungen und dem Arbeitsverzeichnis in die *run*-Funktion weitergegeben. Zuerst wird das Ausgangsmodell berechnet, dann Variationen der Störungsparameter und schliesslich Variationen der Studienparameter.

```
if __name__ == '__main__':
    initial_parameter = product(
        [2.55],
        [2.55],
        [2.60],
        [2.37],
        [2.57],
        [0.0],
        [0.0],
        [0.0],
        [0.0],
        [0.0],
        [0.0],
        [0.0],
        [0.0],
    )

    disturbance_parameter = product(
        [2.53, 2.57],
        [2.53, 2.57],
        [2.58, 2.62],
        [2.32, 2.42],
        [2.52, 2.62],
        [0.0],
        [0.0],
        [0.0],
        [0.0],
        [0.0],
        [0.0],
        [0.0],
        [0.0],
    )

    study_parameter = product(
        [2.55],
        [2.55],
        [2.60],
        [2.37],
        [2.57],
        [-100., 100.],
        [-100., 100.],
        [-5.0, 5.0],
        [-5.0, 5.0],
        [-5.0, 5.0],
        [-5.0, 5.0],
        [-5.0, 5.0],
        [-5.0, 5.0],
    )
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
parent_path = pathlib.Path("example/synthetic_model")

i_path = parent_path / "initial"
run(OPTIONS, i_path, initial_parameter)

d_path = parent_path / "disturbance"
run(OPTIONS, d_path, disturbance_parameter)

s_path = parent_path / "study"
run(OPTIONS, s_path, study_parameter)
```

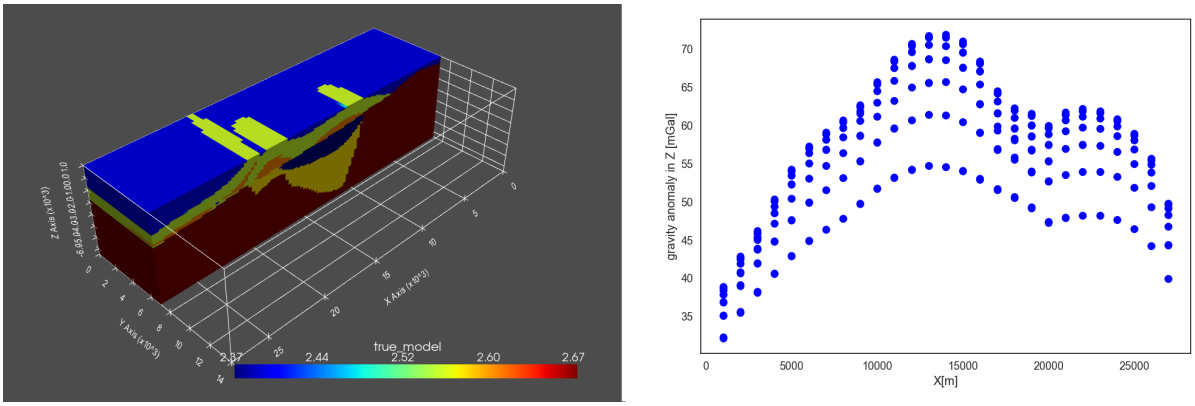

Abb. 7.1: Ausgangsmodells: Links die Schichten visualisiert bei der Dichte und rechts berechnete gravimetrische Anomalie.

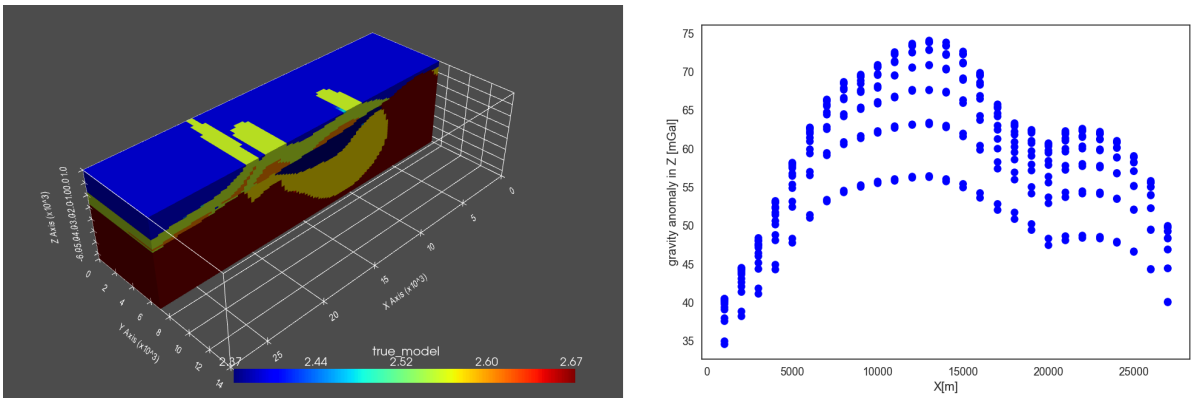

Abb. 7.2: Modifiziertes Modell 1: Links die Schichten visualisiert bei der Dichte und rechts berechnete gravimetrische Anomalie.

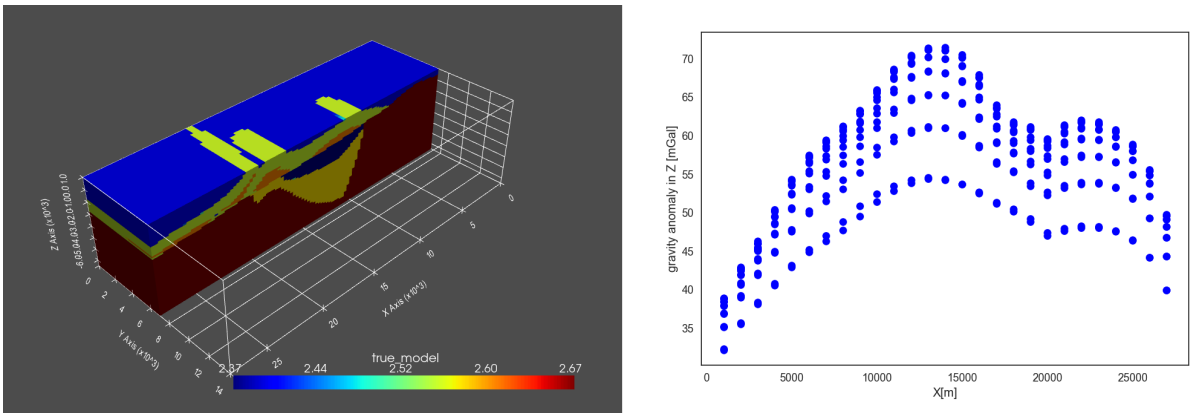

Abb. 7.3: Modifiziertes Modell 2: Links die Schichten visualisiert bei der Dichte und rechts berechnete gravimetrische Anomalie.

7.2 Sensitivitätsanalyse

Wir haben eine Reihe von Berechnungen durchgeführt und wenden nun die im Abschnitt 5 beschriebene Sensitivitätsanalyse an.

In dem Skript werden wir *pandas* verwenden, um die Ergebnisse für die Datenanalyse zu verarbeiten, *pyvista*, um die Ergebnisse im VTK-Format zu verarbeiten, und wiederum *numpy* und *pathlib*. Die Sensitivitätsanalyse-Verarbeitung und die Störungsparameter sind vorerst als zwei Utilities implementiert, die wir ebenfalls importieren.

```
import pathlib
import pyvista as pv
import numpy as np
import pandas as pd

from prospection.utils.sensitivity import sensitivity
from prospection.utils.sensitivity import disturbance
```

Zum Abrufen von Parametern und Daten aus Prospektionsmethoden benötigen wir mehrere Hilfsfunktionen.

```
def read_parameter(file_path):
    result = {}
    with file_path.open('r') as f:
        for line in f.readlines():
            if line:
                key, value = line.strip().split(' ')
                result[key] = float(value)

    return result


def get_density_model(working_dir):
    vtk_data = pv.read(working_dir / 'forward' / 'density_model.vtr')
    return vtk_data.cell_arrays['true_model']

def get_receiver_type():
    return 'gravity'
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
def get_receiver_data(working_dir):
    vtk_filename = 'gravity_observation_points.vtk'
    vtk_data = pv.read(working_dir / 'forward' / vtk_filename)
    return vtk_data.point_arrays['gravity anomaly']
```

Die folgende Funktion geht die einzelnen berechneten Fälle durch, holt die notwendigen Daten und speichert sie in einer Tabelle im pandas.Dataframe zur weiteren Verarbeitung. Die Markierung der Spalten mit p, d und r unterscheidet Studienparameter, Störungsparameter und Ergebnisse.

```
def evaluate_results(
    parent_path,
    study_parameters,
    disturbance_parameters,
    pickle_name="results.pkl"
):
    results = []
    path = pathlib.Path(parent_path)
    return_name = get_receiver_type()

    for case_path in sorted(path.glob('case/*')):
        parameter_file = case_path / "evaluation/parameter"
        table = read_parameter(parameter_file)

        case_return = get_receiver_data(case_path)
        case_density_model = get_density_model(case_path)

        evaluations = {}

        for parameter in study_parameters:
            evaluations["p_" + parameter] = table[parameter]

        for parameter in d_parameters:
            evaluations["d_" + parameter] = table[parameter]

        for num, value in enumerate(case_return, start=1):
            evaluations["r_" + str(num)] = value

        results.append(evaluations)

    df = pd.DataFrame(results)

    file_path = path / pickle_name
    df.to_pickle(file_path)
    return df
```

Die Bewertungsfunktion wird anschliessend vom Hauptteil des Skripts aus aufgerufen. Zunächst führen wir die Störungsparameteranalyse und dann die eigentliche Sensitivitätsanalyse für die definierte Receiver-Abweichung durch.

```
if __name__ == '__main__':
    parent_path = pathlib.Path(__file__).resolve().parent
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    pickle_name = f'results_gravity_disturbances.pkl'

    study_parameters = []
    disturbances = ['rho_muschelkalk', 'rho_effinger', 'rho_graben', 'rho_basement',
↪'offset_muschelkalk', 'offset_effinger', 'offset_graben']

    df_d = evaluate_results(
        parent_path / 'disturbance',
        study_parameters,
        disturbances,
        pickle_name
    )

    receiver_disturbance = disturbance(df_d)

    pickle_name = f'results_gravity_study.pkl'
    study_parameters = ['dipA', 'dipB', 'depth_graben', 'fault_1_x', 'fault_2_x' ]
    disturbances = []

    df_s = evaluate_results(
        parent_path / 'study',
        study_parameters,
        disturbances,
        pickle_name,
        divider
    )

    receiver_deviation = 0.1
    parameter_sensitivity = sensitivity(df_s, receiver_disturbance, receiver_deviation)

    print(parameter_sensitivity)
```

Als Ergebnis erhalten wir eine Tabelle der Unsicherheiten, wie in Abschnitt 5 beschrieben. In Tab. 7.1 ist ein Beispiel für einen Vergleich für zwei verschiedene Empfängerabweichungen.

Tab. 7.1: Vergleich der Unsicherheiten von gravimetrischen Studien.

STD σ_p	STD Messung 0.05 mGal	STD Messung 0.1 mGal
Offset Upper-Filling [m]	59.37	113.42
Offset Lower-Filling [m]	55.58	108.31
Dip Fault 2 [°]	2.718	5.205
Dip Fault 5 [°]	7.511	14.118
Dip Fault 6 [°]	3.561	6.848

Literaturverzeichnis

[1] Pierrick Altwegg. *Gravimetry for geothermal exploration: methodology, computer programs and two case studies in the Swiss Molasse basin*. PhD thesis, Université de Neuchâtel, 2015.

[2] Mathias Bauer, Willi Freeden, Hans Jacobi, and Thomas Neu. *Handbuch Tiefe Geothermie*. Springer, 2014.

[3] Rowan Cockett, Seogi Kang, Lindsey J Heagy, Adam Pidlisecky, and Douglas W Oldenburg. Simpeg: an open source framework for simulation and gradient based parameter estimation in geophysical applications. *Computers & Geosciences*, 85:142–154, 2015.

[4] Willi Freeden and Helga Nutz. Mathematik als Schlüsseltechnologie zum Verständnis des Systems „Tiefe Geothermie “. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 117(1):45–84, 2015.

[5] Luca Guglielmetti, Lorenzo Perozzi, David Dupuy, Francois Martin, Valentin Métraux, Michel Meyer, Goran Mijic, Andrea Moscariello, Carole Nawratil de Bono, and Pier V. Radogna. High resolution gravity data to characterize density variations and reduce uncertainty in geothermal reservoirs in the geneva basin (gb). In *Proceedings World Geothermal Congress 2020*. 2020.

[6] Philip Klingler. *Charakterisierung des geothermischen Reservoirs Riehen: 3D Struktur und Tracer-Test*. CHYN-Zentrum für Hydrogeologie, Universität Neuenburg, 2010.

[7] Jan Niederau, Anozie Ebigbo, Gabriele Marquart, Juliane Arnold, and Christoph Clauser. On the impact of spatially heterogenous permeability on free convection in the perth basin, australia. *Geothermics*, 66:119–133, 2017.

[8] Miguel de la Varga, Alexander Schaaf, and Florian Wellmann. Gempy 1.0: open-source stochastic geological modeling and inversion. *Geoscientific Model Development*, 12(1):1–32, 2019.

[9] Richard Webster and Margaret A Oliver. *Geostatistics for environmental scientists*. John Wiley & Sons, 2007.