

Universität Leipzig
Fakultät für Mathematik und Informatik
Mathematisches Institut

Eine erweiterte Theis-Lösung:
Berechnung und Implementierung einer Lösung der transienten
Grundwassergleichung unter Berücksichtigung von Heterogenität im
Coarse-Graining-Modell

Diplomarbeit

Leipzig, Juli 2015

vorgelegt von:
Sebastian Müller
Diplom Mathematik

Betreuender Hochschullehrer:
Prof. Dr. Rainer Schumann
Fakultät für Mathematik und Informatik
Mathematisches Institut

Externe Betreuerin:
Dr. Alraune Zech
Helmholz-Zentrum für Umweltforschung - UFZ
Department Hydrosystemmodellierung

Zusammenfassung

Die vorliegende Arbeit behandelt die Modellierung von Fließprozessen in Grundwasserleitern. Grundlage dafür ist die Grundwassergleichung, welche diese Prozesse mathematisch beschreibt. Die wichtigste hydraulische Eigenschaft von Untergründen ist hierbei die hydraulische Leitfähigkeit, welche die Fließgeschwindigkeit des Grundwassers angibt. Da man die Verteilung der Leitfähigkeit in einem betrachteten Wasserleiter aber durch das Fehlen von Informationen nicht vollständig bestimmen kann, ist man auf vereinfachende Modelle und Versuchsszenarien angewiesen. In der Vergangenheit haben sich zur Untersuchung von Böden sogenannte Pumptests etabliert, wobei ein Brunnen gebohrt wird, welcher den Grundwasserleiter vollständig durchteuft und an dem mit konstanter Rate Wasser aus dem Boden gepumpt wird. Parallel beobachtet man an einem oder mehreren Referenzbrunnen die sich verändernde hydraulische Druckhöhe. Aus diesen Daten möchte man Informationen über den betrachteten Boden gewinnen. Dazu braucht es gewisse Modellfunktionen, welche für fest definierte Standardsituationen das Grundwasserverhalten beschreiben. Eine Möglichkeit, Böden zu klassifizieren ist es, die heterogene Struktur der Leitfähigkeit durch log-normal verteilte Zufallsgrößen zu modellieren. Dabei beschränkt man sich auf den Mittelwert μ , die Varianz σ^2 und die Korrelationslänge ℓ dieser Verteilungen. Das hier zugrunde liegende Coarse-Graining-Modell generiert zu diesen Parametern eine effektive Leitfähigkeitsverteilung, welche nur vom radialen Abstand zum Pumpbrunnen abhängt. Damit wird die Grundwassergleichung zu einer radialsymmetrischen parabolischen Differentialgleichung.

Die resultierende Aufgabe war es, diese Differentialgleichung zu lösen und die entstehende Lösung zu implementieren. Dazu wird zuerst die Laplacetransformation entwickelt, um mit Hilfe des Differentiationssatzes die Zeitableitung zu eliminieren. Daraus resultiert eine gewöhnliche Differentialgleichung im Laplaceraum. Diese Differentialgleichung wird dann in Brunnennähe durch einen verallgemeinerten Potenzreihenansatz gelöst, welcher durch die Frobenius-Methode gegeben ist. Da diese Lösung nur für kleine Radien zulässig ist, werden die entstehenden Basislösungen durch numerische Integration mittels des Rosenbrock-Verfahrens fortgesetzt und die Differentialgleichung im Fernfeld durch eine asymptotische Entwicklung der Basisfunktionen gelöst. Durch diese dreiteilige Entwicklung der Lösung ist es dann sehr einfach möglich alle gegebenen Rand- und Anfangswerte einzupflegen. Letztlich wird diese Lösung aus dem Laplaceraum mit Hilfe des Stehfest-Algorithmus numerisch rücktransformiert.

Es werden zuerst die hydrologischen Grundlagen für Pumpversuche eingeführt, um den physikalischen Rahmen abzustecken. Es folgt eine kurze Einführung in die Geostatistik und eine Erläuterung des Coarse-Graining-Modells. Mit diesen Grundlagen wird dann die Aufgabe mathematisch exakt formuliert. Danach werden theoretische Vorbetrachtungen gemacht und die jeweils nötigen mathematischen Werkzeuge zur Lösung in den verschiedenen Abschnitten bereitgestellt. Dazu gehören die Laplacetransformation und der Stehfest-Algorithmus zur numerischen Rücktransformation, die Frobenius-Methode, das Rosenbrock-Verfahren, Operationen mit Potenzreihen, die Theis-Lösung für homogene Wasserleiter und damit verbunden die modifizierten Besselfunktionen, welche die Basisfunktionen der homogenen Grundwassergleichung im Laplaceraum darstellen. Nach diesen Vorbetrachtungen werden die Lösungsmethoden auf die konkrete Differentialgleichung angewendet. Dabei werden zum einen die Potenzreihen der Koeffizientenfunktionen berechnet und damit eine Bildungsvorschrift für die Potenzreihen der Basisfunktionen gegeben. Zum anderen werden die Basisfunktionen im Fernfeld berechnet und es wird dann analysiert, wie diese Abschnitte zusammenzufügen sind. Im letzten Teil der Arbeit wird dieses Vorgehen dann analysiert und ausgewertet.

Inhaltsverzeichnis

1. Einführung	1
1.1. Grundlagen	1
1.2. Formulierung der Aufgabe	6
1.3. Vorgehen	9
2. Theoretische Grundlagen	11
2.1. Laplacetransformation	11
2.2. Frobenius-Methode	14
2.3. Theis-Lösung	19
2.4. Potenzreihen	27
2.5. Rosenbrock-Verfahren	34
3. Umsetzung	37
3.1. Anwendung der Frobenius-Methode	37
3.2. Fortsetzung durch numerische Integration	43
3.3. Lösung im Fernfeld	44
3.4. Einarbeitung der Randwerte	45
3.5. Die Wahl der Lösungsabschnitte	48
3.6. Implementierung der erweiterten Theis-Lösung	51
4. Auswertung	57
4.1. Fehleranalyse	57
4.2. Plots und Vergleich mit der Theis-Lösung	65
4.3. Fazit und Ausblick	68
A. Anhang	71
A.1. Algorithmen	71
A.2. Visualisierung der Fehleranalyse	90
A.3. Einfluss der Parameter auf die Lösung	96
Symbolverzeichnis	103
Abbildungsverzeichnis	105
Liste der Programmcodes	107
Literatur	109

1. Einführung

1.1. Grundlagen

1.1.1. Pumpversuche

Zur Untersuchung von Grundwasserfließprozessen in verschiedenen Untergründen haben sich Pumpversuche etabliert, wobei an einem Pumpbrunnen mit konstanter Rate Grundwasser abgepumpt wird und an Referenzbrunnen die sich verändernde Grundwasserspiegelhöhe gemessen wird (siehe Abbildung 1). Aus diesen Daten möchte man Rückschlüsse auf die Beschaffenheit des Untergrundes, insbesondere die hydraulische Leitfähigkeit K ziehen. Die Leitfähigkeit K bestimmt die Fließgeschwindigkeit des Grundwassers und kann dabei über mehrere Größenordnungen schwanken (vgl. [Gelhar, 1993]). Dazu benötigt man geeignete Modelle, welche diesem Untersuchungsszenario gerecht werden. Eine Möglichkeit dazu wurde in [Schneider und Attinger, 2008] mit der Coarse-Graining Methode gegeben, welche ich im folgenden kurz vorstellen will. Ich folge dabei der Abhandlung in [Zech, 2013, S. 21ff].

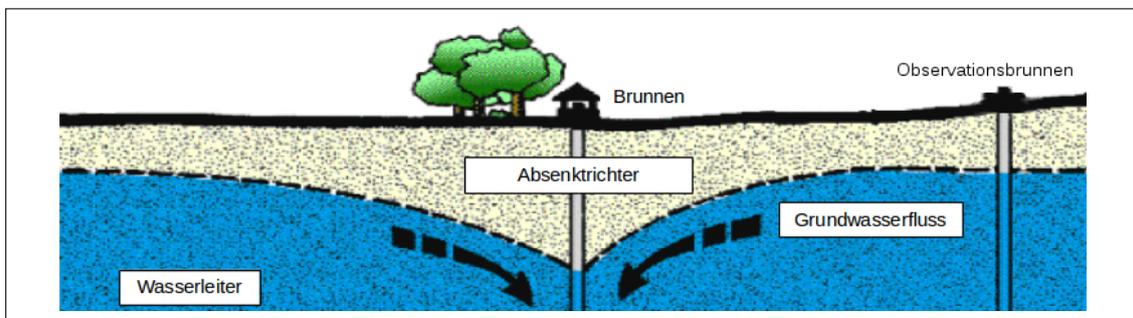


Abbildung 1: Schema eines Pumpversuchs.

1.1.2. Die Grundwassergleichung und das Darcy-Gesetz

Physikalisch wird das Verhalten des Grundwassers durch die *Kontinuitätsgleichung* beschrieben, welche aus der Energiebilanz des Grundwassers hervorgeht. Die sogenannte *Grundwassergleichung* sieht wie folgt aus (vgl. [Zech, 2013, S. 21]):

$$S \frac{\partial h}{\partial t}(\underline{x}, t) + \nabla \cdot \underline{q}(\underline{x}, t) = Q(\underline{x}, t) \quad (1.1.1)$$

Dabei ist S der *spezifische Speicherkoeffizient*, welcher angibt wie viel Wasser der Wasserleiter speichern oder abgeben kann, h die *hydraulische Druckhöhe* oder auch *Standrohrspiegelhöhe* genannt, \underline{q} der *Darcysche Geschwindigkeitsvektor*, welcher den Fluss in einem Punkt beschreibt und Q der *Quellterm*.

Mit dem *Darcy-Gesetz*, benannt nach dem französischen Ingenieur Henry Darcy, kann man den Fluss mit der Standrohrspiegelhöhe umschreiben. Das Gesetz besagt:

$$\underline{q}(\underline{x}, t) = -K_f(\underline{x}) \cdot \nabla h(\underline{x}, t) \quad (1.1.2)$$

Dabei ist K_f der *Durchlässigkeitsbeiwert* des Wasserleiters. Kombiniert man (1.1.1) mit (1.1.2) und ersetzt den Quellterm durch eine noch zu formulierende Randbedingung, so erhält man die sogenannte *saturierte Grundwassergleichung* (vgl. [Zech, 2013, S. 36] und [Häfner et al., 1992, S. 12]):

$$\nabla \cdot (K_f(\underline{x}) \cdot \nabla h(\underline{x}, t)) = S \cdot \frac{\partial h}{\partial t}(\underline{x}, t) \quad (1.1.3)$$

Diese parabolische partielle Differentialgleichung soll der Ausgangspunkt aller folgenden Betrachtungen sein.

1.1.3. Geostatistik und die Beschreibung der Leitfähigkeit

Da die Verteilung der Leitfähigkeit im Boden durch das Fehlen von Informationen nicht vollständig bestimmt werden kann, versucht man diese Unsicherheit durch Wahrscheinlichkeitsverteilungen zu handhaben. In der Geostatistik hat sich die Modellierung der Leitfähigkeitsverteilung $K(\underline{x})$ als log-normal verteilte räumliche Zufallsgröße etabliert. Das heißt, dass $Y(\underline{x}) = \ln K(\underline{x})$ eine normal verteilte Zufallsgröße mit gaußscher Dichtefunktion φ_Y ist:

$$\varphi_Y(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (1.1.4)$$

Daraus ergibt sich für die Verteilungsdichtefunktion φ_K von K :

$$\varphi_K(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\ln x - \mu}{2\sigma^2}\right) \quad (1.1.5)$$

Betrachten wir die verschiedenen Momente von K . Verschiedene Mittelwerte lassen sich für $K(\underline{x})$ bestimmen: das *arithmetische Mittel* K_A , das *geometrische Mittel* K_G und das *harmonische Mittel* K_H , welche im folgenden eine wichtige Rolle spielen. Gemäß des Zusammenhangs zwischen log-normal und normal verteilten Zufallsgrößen ergibt sich:

$$\begin{aligned} K_G &= \exp(\mu) \\ K_A &= K_G \cdot \exp\left(\frac{\sigma^2}{2}\right) \\ K_H &= K_G \cdot \exp\left(-\frac{\sigma^2}{2}\right) \end{aligned} \quad (1.1.6)$$

Die *Varianz* von K in Abhängigkeit der Parameter σ^2 und μ ist gegeben mit:

$$\text{Var}(K) = \left(\exp(\sigma^2) - 1\right) \exp\left(2\mu + \frac{\sigma^2}{2}\right) \quad (1.1.7)$$

Die räumliche Korrelation der Leitfähigkeitswerte wird beschrieben durch die *Autokovarianz*:

$$\text{CV}_K(\underline{s}) = \text{Cov}(K(\underline{x} + \underline{s}), K(\underline{x})) \quad (1.1.8)$$

Im folgenden beschränke ich mich, in Anlehnung an die Arbeit von [Zech, 2013], auf ein gaußartiges Modell:

$$\text{CV}(\underline{x}) = \sigma^2 \cdot \exp\left(-\frac{x^2}{\ell_1^2} - \frac{y^2}{\ell_2^2} - \frac{z^2}{\ell_3^2}\right) \quad (1.1.9)$$

Dabei nimmt man folgende Relationen der Korrelationslängen ℓ_i an:

$$\begin{aligned} \text{in horizontaler } x, y\text{-Richtung: } \ell_1 = \ell_2 &= \ell \\ \text{in vertikaler } z\text{-Richtung: } \ell_3 = \ell_z &= e\ell \end{aligned} \quad (1.1.10)$$

Man nennt ℓ die *Korrelationslänge*, welche angibt, über welche Distanz die Leitfähigkeit in etwa gleich bleibt und $e \in [0, 1]$ die *Anisotropierate*, welche das Verhältnis von vertikaler und horizontaler Korrelationslänge $\frac{\ell_z}{\ell}$ beschreibt.

Aus den Größen μ , σ^2 , ℓ und e kann man synthetische Böden generieren, welche dieser log-normal verteilten Leitfähigkeit entsprechen. In Abbildung 2 sind solche generierten Böden dargestellt und man erkennt gut die Bedeutung von unterschiedlichen Korrelationslängen und der Anisotropie.

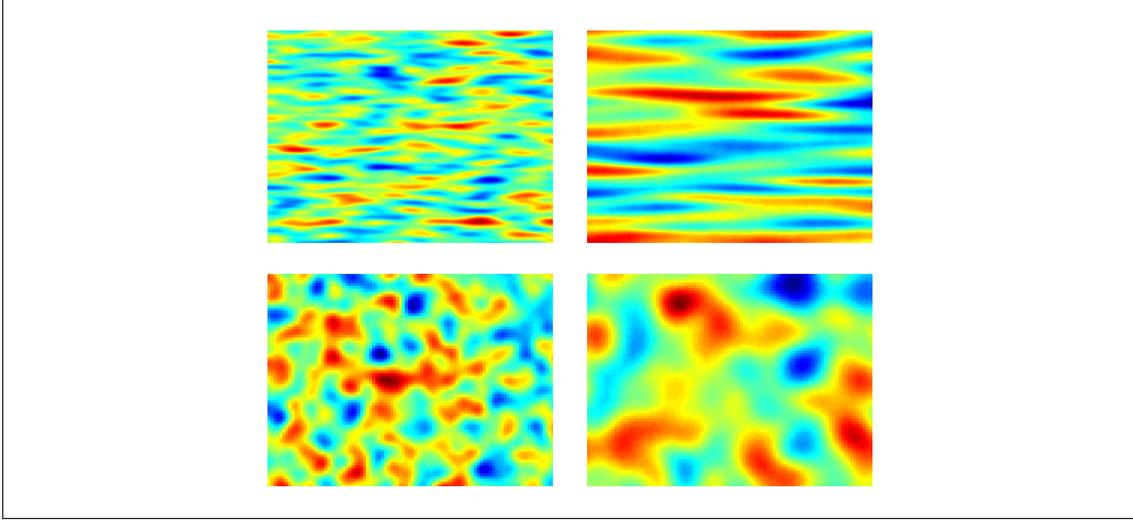


Abbildung 2: Log-normal verteilte Zufallsfelder $K(x)$ mit Gaußscher Korrelationsstruktur für festen Mittelwert μ und feste Varianz σ^2 . Dabei wurde $e = \frac{1}{4}$ (oben) und $e = 1$ (unten), sowie $\ell = 5$ (links) und $\ell = 10$ (rechts) gewählt. Die oberen Bilder lassen sich als vertikale Querschnitte und die unteren als horizontale Querschnitt durch ein stochastisches poröses Medium interpretieren. Die Bilder wurden mit einem Generator erstellt, der nach den Betrachtungen in [Heße et al., 2014] implementiert wurde.

Im Anwendungsbereich unterscheidet man zwei Modelle, welche durch unterschiedliche Skalen-Ordnungen entstehen. Zum einen das natürliche $3D$ -Modell (kleinskalig), welches sich aus der obigen Betrachtung ergibt und zum anderen ein $2D$ -Modell (regionalskalig), wobei die Wasserleiterstärke gegenüber der flächenmäßigen Ausdehnung des Wasserleiters vernachlässigbar wird. In diesem Fall spricht man nicht mehr von der Leitfähigkeit $K(x, y, z)$, sondern von der Durchlässigkeit $T(x, y)$. Diese ergibt sich durch das vertikale Aufintegrieren der Leitfähigkeit (vgl. [Zech, 2013, S. 30]):

$$T(x, y) = \int_{-L}^0 K(x, y, z) dz \quad (1.1.11)$$

Auch die Durchlässigkeit kann man wieder als log-normal verteilte Zufallsgröße zu den Parametern σ^2 und μ modellieren. Es ergeben sich auch hier folgende Formeln für das *geometrische Mittel* T_G und das *harmonische Mittel* T_H :

$$\begin{aligned} T_G &= \exp(\mu) \\ T_H &= T_G \cdot \exp\left(-\frac{\sigma^2}{2}\right) \end{aligned} \quad (1.1.12)$$

1.1.4. Das Coarse-Graining-Modell für Brunnenfluss

Das Coarse-Graining-Modell gibt eine effektive Beschreibung der Leitfähigkeit, die bei einem Pumpstest auftritt. Einfach zusammengefasst berücksichtigt das Modell, dass die Heterogenität in Brunnennähe einen wesentlich höheren Einfluss auf den Grundwasserfluss bei einem Pumpversuch hat, als heterogene Strukturen im Fernfeld.

Das Coarse-Graining-Modell generiert zu den Parametern μ , σ^2 , ℓ und e eine effektive Leitfähigkeitsverteilung $K^{CG}(r)$ respektive Durchlässigkeitsverteilung $T^{CG}(r)$ in einem radialsymmetrischen Modell.

In 2D ist die Coarse-Graining-Durchlässigkeit wie folgt gegeben (siehe Abbildung 3 (links)):

$$T_H^{\text{CG}}(r) = T_G \cdot \exp\left(\frac{-\frac{\sigma^2}{2}}{1 + \left(\frac{\zeta}{\ell} \cdot r\right)^2}\right) \quad (1.1.13)$$

Man erhält als Nahfeldwert $T_{\text{well}} = T_H$ für $r \rightarrow 0$ und als Fernfeldwert $T_{\text{efu}} = T_G$ für $r \rightarrow \infty$, wobei T_G wie in (1.1.12) definiert ist. Die Größe ζ ist dabei ein dimensionsloser Proportionalitätsfaktor. Das heißt, die Coarse-Graining-Durchlässigkeit $T_H^{\text{CG}}(r)$ schafft einen stetigen Übergang von einem effektiven Durchlässigkeitswert am Brunnen zu einem effektiven Durchlässigkeitswert im Fernfeld.

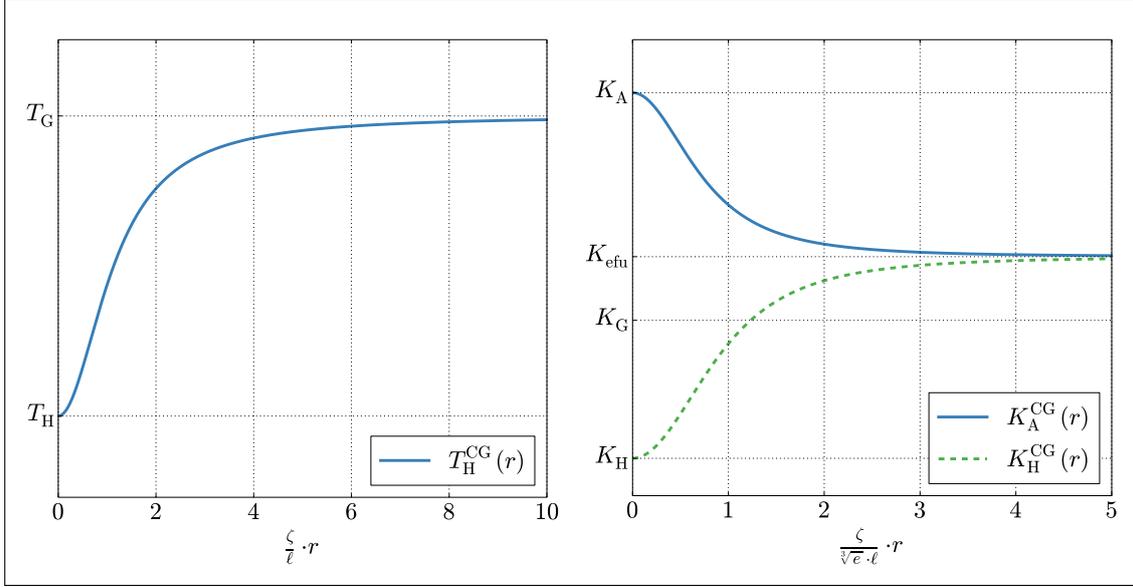


Abbildung 3: Die Coarse-Graining-Durchlässigkeit $T_H^{\text{CG}}(r)$ (links) und -Leitfähigkeit $K^{\text{CG}}(r)$ (rechts)

In 3D unterscheidet man zwischen zwei verschiedenen Randbedingungen am Brunnen, nämlich $K_{\text{well}} = K_A$ oder $K_{\text{well}} = K_H$. Die Wahl von K_A nenne ich im folgenden *BCA-Fall* und die Wahl von K_H *BCH-Fall*. Der Fernfeldwert berechnet sich dagegen einheitlich mit:

$$K_{\text{efu}} = K_G \cdot \exp\left(\sigma^2 \left(\frac{1}{2} - \gamma(e)\right)\right) \quad (1.1.14)$$

Dabei ist $\gamma(e)$ die sogenannte *Anisotropiefunktion* (vgl. [Zech, 2013, S. 92]):

$$\gamma(e) = \frac{e}{2(1-e^2)} \left(\frac{1}{\sqrt{1-e^2}} \arctan \sqrt{\frac{1}{e^2} - 1} - e \right) \quad (1.1.15)$$

Diese Funktion hat einen Wertebereich von $\left[0, \frac{1}{3}\right]$ mit $\gamma(0) = 0$ und $\gamma(1) = \frac{1}{3}$ wie in Abbildung 4 zu sehen. Für eine einheitliche Beschreibung führt man folgende Abkürzung ein:

$$\chi = \ln \frac{K_{\text{well}}}{K_{\text{efu}}} \quad (1.1.16)$$

Man erhält für die beiden Nahfeldwerte also $\chi_A = \sigma^2 \gamma(e)$ und $\chi_H = \sigma^2 (\gamma(e) - 1)$. Damit sieht die Coarse-Graining-Leitfähigkeit wie folgt aus (siehe Abbildung 3 (rechts)):

$$K^{\text{CG}} = K_{\text{efu}} \cdot \exp\left(\chi \cdot \left(1 + \left(\frac{\zeta}{\sqrt{e^2} \cdot \ell} \cdot r\right)^2\right)^{-\frac{3}{2}}\right) \quad (1.1.17)$$

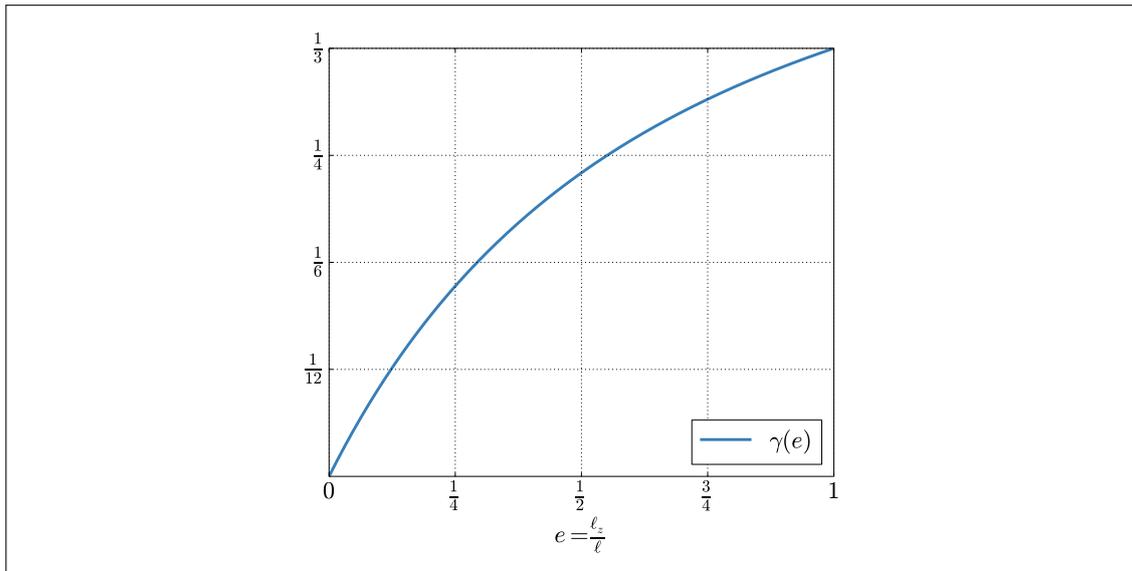


Abbildung 4: Die Anisotropiefunktion $\gamma(e)$, welche den Einfluss der Anisotropierate wiedergibt.

In beiden Modellen vereinfacht sich die Grundwassergleichung also auf ein radial-symmetrisches Problem, welches sich wesentlich einfacher bearbeiten lässt.

1.2. Formulierung der Aufgabe

Meine Aufgabe während meiner Arbeit unter der Aufsicht von Frau Dr. Zech am Umweltforschungszentrum in Leipzig bestand darin, die saturierte transiente Grundwassergleichung (1.1.3), unter Verwendung der Coarse-Graining Lösung für die Leitfähigkeit, zu lösen und zu implementieren. Dabei ist $K(\underline{x})$ eine der beiden oben vorgestellten Funktionen K^{CG} (1.1.17) bzw. T_{H}^{CG} (1.1.13). In beiden Modellen handelt es sich um eine Differentialgleichung mit zwei Raumkoordinaten. Im 2D-Modell ist dies klar und im 3D-Modell entsteht diese Vereinfachung durch die Betrachtung des vertikalen Mittels der Standrohrspiegelhöhe (vgl. [Zech, 2013, S. 127]):

$$h(x, y, t) = \frac{1}{L} \int_{-L}^0 \bar{h}(x, y, z, t) dz \quad (1.2.1)$$

Da die Gleichung radial-symmetrisch ist, kann man annehmen, dass h über z konstant ist. Dies nennt man in der Literatur die *Dupuit-Forchheimer Annahme*. Die Annahme besagt (vgl. [Delleur, 1999, S. 2-13ff]):

$$\frac{\partial}{\partial z} \bar{h}(x, y, z, t) = 0 \quad (1.2.2)$$

Somit vereinfacht sich das vertikale Mittel in Gleichung (1.2.1) zu:

$$h(x, y, t) = \bar{h}(x, y, z_0, t) \quad (1.2.3)$$

Die Differentialgleichung vereinfacht sich dann im 3D-Fall also wirklich auf zwei Raumkoordinaten, da auch $\frac{\partial}{\partial z} K(\underline{x}) = 0$ gilt. Mit $K(x, y) = K(x, y, z_0)$ ergibt sich:

$$\begin{aligned} \nabla \left(K(\underline{x}) \cdot \nabla \bar{h}(\underline{x}, t) \right) &= K_x \bar{h}_x + K \bar{h}_{xx} + K_y \bar{h}_y + K \bar{h}_{yy} + \underbrace{K_z \bar{h}_z + K \bar{h}_{zz}}_{=0} \\ &= \nabla (K(x, y) \cdot \nabla h(x, y, t)) \end{aligned} \quad (1.2.4)$$

Die Form der Differentialgleichung ändert sich als nicht.

1.2.1. Die Anfangs- und Randwerte

Mit den Anfangs- und Randwerten gibt man die Rahmenbedingungen des Pumpstests an. Die Anfangsbedingung beschreibt die Standrohrspiegelhöhen zu einem gewissen Anfangszeitpunkt, den man ohne Einschränkungen auf $t = 0$ setzt. Im gegebenen Fall geht man von einer homogenen Anfangsbedingung aus, das heißt:

$$\forall_{\underline{x}} h(\underline{x}, 0) = h_a \quad (1.2.5)$$

Es soll also zum Zeitpunkt $t = 0$ ein konstanter Wasserstand h_a im Wasserleiter vorliegen, sodass im betrachteten Zeitraum eine gespannte Grundwasserströmung vorliegt (vgl. [Häfner et al., 1992, S. 12]). Mit der Substitution auf die *Spiegelhöhendifferenz* (vgl. [Häfner et al., 1992, S. 159]):

$$u(\underline{x}, t) = h(\underline{x}, t) - h_a \quad (1.2.6)$$

kann man dies aber auf die einheitliche Standard-Anfangsbedingung

$$\forall_{\underline{x}} u(\underline{x}, 0) = 0 \quad (1.2.7)$$

festlegen und u erfüllt offensichtlich noch dieselbe Differentialgleichung wie h .

Zur Festlegung der Randwerte benötigt man noch einige räumliche Angaben. Man geht von einem zylinderförmigen Wasserleiter mit Radius $r_\infty \in (0, \infty]$ und einer vertikalen Stärke von L aus. Des Weiteren nimmt man einen ebenfalls zylindrischen Pumpbrunnen um $(x, y) = 0$ mit Radius $r_{\text{well}} \in [0, r_\infty)$ an, welcher den Wasserleiter vollständig durchteuft. Für $r_{\text{well}} = 0$ ist der idealisierte Fall einer *Liniensenke* gemeint, welcher häufig einfache Lösungen der Grundwassergleichung ermöglicht. Am äußeren Rand $\Gamma_\infty = \{\underline{x} \in \mathbb{R}^2 \mid |\underline{x}| = r_\infty\}$ des Wasserleiters geht man von einem unendlich großen Grundwasservorrat aus, was heißt, dass die Standrohrspiegelhöhe über die ganze Zeit konstant ist:

$$\forall t \geq 0 \forall \underline{x} \in \Gamma_\infty h(\underline{x}, t) = 0 \quad (1.2.8)$$

Damit fehlt nur noch die Randbedingung am Pumpbrunnen. Man geht dort von einer konstanten Pumprate Q_w aus und beschreibt die Randbedingung durch den Fluss durch den Pumpbrunnen (vgl. [Zech, 2013, S. 36]). Man erhält in beiden Modellen für die Mantelfläche des Brunnens:

$$\Gamma_{\text{well}}^{2D} = \{\underline{x} \in \mathbb{R}^2 \mid |\underline{x}| = r_{\text{well}}\} \quad (1.2.9)$$

$$\Gamma_{\text{well}}^{3D} = \Gamma_{\text{well}}^{2D} \times [-L, 0] \quad (1.2.10)$$

Somit erhält man:

$$\begin{aligned} Q_w &= \oint_{\Gamma_{\text{well}}^{3D}} \underline{q} d\sigma \\ &= -L \cdot \oint_{\Gamma_{\text{well}}^{2D}} K(\nabla h) d\sigma \end{aligned} \quad (1.2.11)$$

Mit $L = 1$ im $2D$ -Fall. Im Falle $r_{\text{well}} = 0$ setzt man die Randbedingung durch einen Grenzübergang in Gleichung (1.2.11) für $r_{\text{well}} \rightarrow 0_+$ (vgl. [Häfner et al., 1992, S. 242]).

1.2.2. Transformation in Polarkoordinaten

Da alle Randbedingungen radial-symmetrisch formuliert wurden und die Durchlässigkeit respektive Leitfähigkeit durch die Coarse-Graining Methode ebenfalls nur vom Abstand zum Brunnen abhängen, ergibt sich eine ebenfalls radial-symmetrische Standrohrspiegelhöhe um den Brunnen. Daher transformiere ich die Gleichung in Polarkoordinaten:

$$\Phi(r, \varphi) = \begin{pmatrix} r \cos \varphi \\ r \sin \varphi \end{pmatrix} \quad (1.2.12)$$

Mit $r > 0$ und $\varphi \in [0, 2\pi)$. Für die Polarkoordinaten ergibt sich folgende Basis:

$$\underline{e}_r = \begin{pmatrix} \cos \varphi \\ \sin \varphi \end{pmatrix} \quad (1.2.13)$$

$$\underline{e}_\varphi = \begin{pmatrix} -\sin \varphi \\ \cos \varphi \end{pmatrix} \quad (1.2.14)$$

Und folgende Funktionaldeterminante:

$$|\det \nabla \Phi| = r \quad (1.2.15)$$

Sei $A(x, y) = A_r(r, \varphi) \cdot \underline{e}_r + A_\varphi(r, \varphi) \cdot \underline{e}_\varphi$ ein radial-symmetrisches Vektorfeld und $f(x, y)$ eine radial-symmetrische Funktion. Es gilt dann:

$$\begin{aligned} \nabla \cdot A(x, y) &= \frac{1}{r} \frac{\partial}{\partial r} (r \cdot A_r) + \underbrace{\frac{1}{r} \frac{\partial A_\varphi}{\partial \varphi}}_{=0} \\ &= \frac{1}{r} \frac{\partial}{\partial r} (r \cdot A_r) \end{aligned} \quad (1.2.16)$$

$$\begin{aligned} \nabla f(x, y) &= \frac{\partial f}{\partial r} \underline{e}_r + \underbrace{\frac{1}{r} \frac{\partial f}{\partial \varphi} \underline{e}_\varphi}_{=0} \\ &= \frac{\partial f}{\partial r} \underline{e}_r \end{aligned} \quad (1.2.17)$$

Damit ergibt sich für den räumliche Anteil der Differentialgleichung:

$$\nabla \cdot (K(\|\underline{x}\|) \cdot \nabla h(\|\underline{x}\|, t)) = \frac{1}{r} \frac{\partial}{\partial r} \left(r \cdot K(r) \cdot \frac{\partial h}{\partial r}(r, t) \right) \quad (1.2.18)$$

Und somit erhält man folgende Differentialgleichung mit nunmehr nur noch 2 Veränderlichen:

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \cdot K(r) \cdot \frac{\partial h}{\partial r}(r, t) \right) = S \frac{\partial h}{\partial t}(r, t) \quad (1.2.19)$$

Wobei $K(r)$ eine der beiden Funktionen $T_H^{\text{CG}}(r)$ bzw. $K^{\text{CG}}(r)$ ist.

Jetzt gilt es noch die Rand- und Anfangsbedingungen zu transformieren. Als Anfangsbedingung ergibt sich analog:

$$\forall_{r>0} h(r, 0) = 0 \quad (1.2.20)$$

Die Randbedingung im Fernfeld ist durch die radiale Form des Wasserleiters ebenfalls einfach zu formulieren:

$$\forall_{t \geq 0} h(r_\infty, t) = 0 \quad (1.2.21)$$

Falls $r_\infty = \infty$ gilt, betrachtet man den Grenzübergang. Allgemein formuliere ich es wie folgt:

$$\forall_{t \geq 0} \lim_{r \rightarrow r_\infty} h(r, t) = 0 \quad (1.2.22)$$

Für die Brunnenrandbedingung ergibt sich:

$$\begin{aligned} \oint_{\Gamma_{\text{well}}^{2D}} K(\nabla h) \, d\sigma &= \oint_{\Gamma_{\text{well}}^{2D}} \langle K(\nabla h), \underline{n} \rangle \, d\sigma \\ &= \int_0^{2\pi} K(r_{\text{well}}) \cdot \frac{\partial h}{\partial r}(r_{\text{well}}, t) \cdot r_{\text{well}} \cdot \left\langle \begin{pmatrix} \cos \varphi \\ \sin \varphi \end{pmatrix}, \begin{pmatrix} \cos \varphi \\ \sin \varphi \end{pmatrix} \right\rangle \, d\varphi \\ &= 2\pi \cdot K(r_{\text{well}}) \cdot r_{\text{well}} \cdot \frac{\partial h}{\partial r}(r_{\text{well}}, t) \end{aligned} \quad (1.2.23)$$

Im Fall $r_{\text{well}} = 0$ betrachtet man wieder den Grenzübergang. Somit ergibt sich in 2D:

$$\forall_{t>0} \lim_{r \rightarrow r_{\text{well}}} r \cdot \frac{\partial h}{\partial r}(r, t) = -\frac{Q_w}{2\pi T_H^{\text{CG}}(r_{\text{well}})} \quad (1.2.24)$$

Und in 3D:

$$\forall_{t>0} \lim_{r \rightarrow r_{\text{well}}} r \cdot \frac{\partial h}{\partial r}(r, t) = -\frac{Q_w}{2\pi L K^{\text{CG}}(r_{\text{well}})} \quad (1.2.25)$$

1.3. Vorgehen

Zur Lösung der Grundwassergleichung (1.2.19) bin ich weitestgehend den Empfehlungen in [Häfner et al., 1992] gefolgt. Die Schritte sind wie folgt:

- Ausnutzen der Radialsymmetrie, welche durch das Coarse-Graining-Modell entstanden ist, zur Reduktion der Gleichung auf eine Raum-Variable
- Transformation der Gleichung vom Zeit-Raum in den Laplace-Raum zur Eliminierung der Zeitableitung
- Lösen der entstandenen gewöhnlichen Differentialgleichung im Laplace-Raum mit Hilfe der Frobenius-Methode in Umgebung von $r = 0$
- Fortsetzung der entstandenen Basislösungen durch numerische Integration bis zu einem Maximalradius r_{\max}
- Approximation der Fernfeldlösung für $r \geq r_{\max}$ durch die Lösung für homogene Wasserleiter, da die Coarse-Graining-Leitfähigkeit bzw. -Durchlässigkeit gegen feste Werte im Fernfeld konvergieren
- Zusammenfügen der Lösungen mit der Bedingung der stetigen Differenzierbarkeit
- Einarbeitung der Randwerte im Nah- und Fernfeld
- Numerische Rücktransformation der Lösung in den Zeit-Raum mit Hilfe des Stehfest-Algorithmus

Dieser Ablauf wird in Abbildung 5 in einem Diagramm veranschaulicht. Das detaillierte Vorgehen wird in den folgenden Kapiteln erklärt. Dabei entwickle ich zuerst die theoretischen Grundlagen, beschreibe danach die Entwicklung der Lösung und werte dies aus.

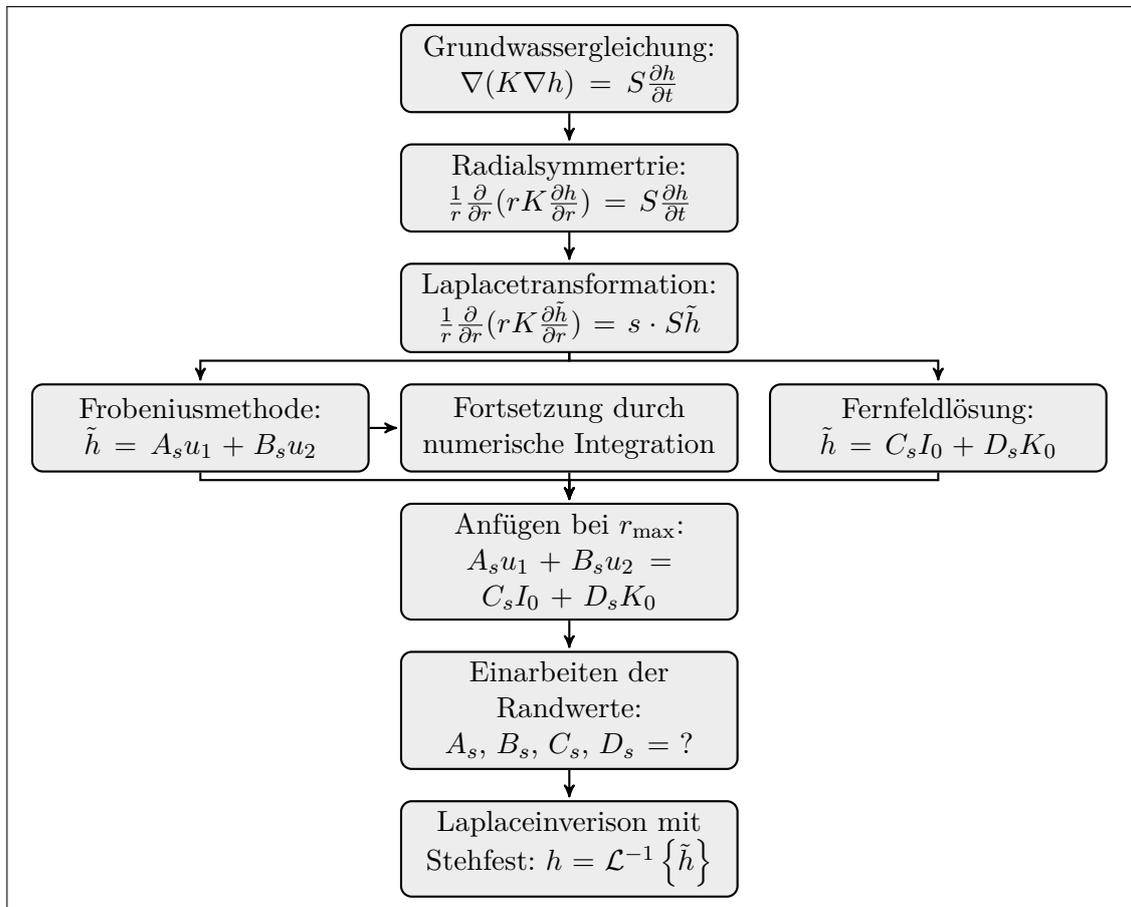


Abbildung 5: Abfolge der Lösungsschritte

2. Theoretische Grundlagen

2.1. Laplacetransformation

Neben der Fouriertransformation ist die Laplacetransformation wohl eine der wichtigsten und meistgenutzten Integraltransformationen (vgl. [Weber und Ulrich, 2007]). Das wichtigste Werkzeug, welches die Laplacetransformation zur Behandlung von Differentialgleichungen bereithält, ist der Differentiationssatz. Durch diesen werden Ableitungen im Zeit-Raum in algebraische Operationen im Laplaceraum umgewandelt. Daher eignet sie sich bei der Behandlung von Transport- und Diffusionsgleichungen ideal, um die auftretende Zeitableitung zu eliminieren. Ich gebe hier eine Definition der Laplacetransformation, führe Begriffe und Bezeichnungen ein und stelle eine Übersicht an Eigenschaften zusammen. Auf die Beweise werde ich hier dabei verzichten und verweise dazu nur auf die Literatur [Timmann, 2010] und [Weber und Ulrich, 2007].

2.1.1. Die Laplacetransformation

2.1.1.1. Definition (Laplacetransformation) Sei $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ eine in jedem Intervall $[0, T]$ absolut integrierbare Funktion und von höchstens exponentiellem Wachstum:

$$\exists_{M,r,t_0 \in \mathbb{R}} \forall_{t \geq t_0} |f(t)| \leq M \cdot e^{rt} \quad (2.1.1)$$

Das uneigentliche Riemannintegral

$$\tilde{f}(s) := \int_0^{\infty} e^{-st} f(t) dt \quad (2.1.2)$$

konvergiere für ein $s_0 \in \mathbb{R}$. Man nennt $\tilde{f} : [s_0, \infty) \rightarrow \mathbb{R}$ die *Laplacetransformierte* zu f .

2.1.1.2. Eigenschaften, Bemerkungen, Konventionen und Rechtfertigungen

1. Man schreibt auch

$$\tilde{f} = \mathcal{L}\{f\} \quad (2.1.3)$$

um direkt auf die Laplacetransformation zu verweisen.

2. Wenn das Integral (2.1.2) für $s_0 \in \mathbb{R}$ konvergiert, so auch für alle $s > s_0$. Wenn (2.1.2) für alle $s < s_k$ divergiert und für alle $s > s_k$ konvergiert, so nennt man s_k die *Konvergenzabszisse* des Integrals (2.1.2). In s_k selbst muss keine Konvergenz vorliegen. Die Laplacetransformierte existiert dann auch für $s \in \mathbb{C}$ mit $\Re(s) > s_k$.
3. Für $s > r$ aus Bedingung (2.1.1) konvergiert das Integral (2.1.2) absolut. Falls (2.1.2) für $s > s_a$ absolut konvergiert und für $s < s_a$ absolut divergiert, so nennt man s_a die *Abszisse absoluter Konvergenz*. In s_a selbst muss ebenfalls keine Konvergenz vorliegen. Die Laplacetransformierte existiert für $s \in \mathbb{C}$ mit $\Re(s) > s_a$ und es liegt auch dort absolute Konvergenz vor.
4. Konvergiert (2.1.2) für ein s_0 absolut, so konvergiert es in $[s_0, \infty)$ gleichmäßig.
5. Die Laplacetransformierte \tilde{f} ist für $s > s_a$ beliebig oft differenzierbar und es gilt

$$\lim_{s \rightarrow \infty} \tilde{f}(s) = 0 \quad (2.1.4)$$

2.1.1.3. Rechenregeln

1. *Linearität*: Seien $f, g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ zwei Funktionen welche den Bedingungen in 2.1.1.1 genügen und $\lambda \in \mathbb{R}$ beliebig. Es gilt:

$$\mathcal{L}\{f + \lambda \cdot g\} = \mathcal{L}\{f\} + \lambda \cdot \mathcal{L}\{g\} \quad (2.1.5)$$

2. *Ähnlichkeit*: Sei $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ eine Funktion welche den Bedingungen in 2.1.1.1 genügen und $\lambda > 0$ beliebig. Dann ist auch $f(\lambda t)$ Laplacetransformierbar und im gemeinsamen Transformationsbereich gilt:

$$\mathcal{L}\{f(\lambda \cdot [\cdot])\} = \frac{1}{\lambda} \cdot \tilde{f}\left(\frac{s}{\lambda}\right) \quad (2.1.6)$$

3. *Verschiebung*: Sei $f : \mathbb{R} \rightarrow \mathbb{R}$ eine Funktion, wobei $f|_{[0, \infty)}$ den Bedingungen in 2.1.1.1 genüge und $f|_{(-\infty, 0)} \equiv 0$ gelte. Dann gilt für $\lambda \in \mathbb{R}$:

$$\mathcal{L}\{f([\cdot] + \lambda)\} = e^{\lambda s} \cdot \left(\tilde{f}(s) - \int_0^{\lambda} e^{-st} f(t) dt \right) \quad (2.1.7)$$

4. *Dämpfung*: Sei $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ eine Funktion, welche den Bedingungen in 2.1.1.1 genüge und $\lambda > 0$ beliebig. Dann ist auch $e^{-\lambda t} f(t)$ Laplacetransformierbar und es gilt:

$$\mathcal{L}\{e^{-\lambda [\cdot]} \cdot f\} = \tilde{f}(s + \lambda) \quad (2.1.8)$$

5. *Differentiation nach Parameter*: Sei $f : \mathbb{R}_{\geq 0} \times \mathbb{R} \rightarrow \mathbb{R}$ eine Funktion, wobei $f([\cdot], r)$ für alle $r \in \mathbb{R}$ den Bedingungen in 2.1.1.1 genüge und $f(t, [\cdot])$ für alle $t \geq 0$ stetig differenzierbar ist und $\frac{\partial}{\partial r} f([\cdot], r)$ ebenfalls für alle $r \in \mathbb{R}$ Laplacetransformierbar ist. Dann gilt:

$$\mathcal{L}\left\{\frac{\partial}{\partial r} f([\cdot], r)\right\} = \frac{\partial}{\partial r} \tilde{f}(s, r) \quad (2.1.9)$$

6. *Grenzwerte*: Sei $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ differenzierbar und f' genüge den Bedingungen in 2.1.1.1 und es gelte $s_a = 0$. Falls $\lim_{t \rightarrow \infty} f(t)$ existiert¹, so gilt:

$$\lim_{t \rightarrow \infty} f(t) = \lim_{s \rightarrow 0_+} s \cdot \tilde{f}(s) \quad (2.1.10)$$

$$\lim_{t \rightarrow 0_+} f(t) = \lim_{s \rightarrow \infty} s \cdot \tilde{f}(s) \quad (2.1.11)$$

7. *Differentiation im Laplacebereich*: Sei f Laplacetransformierbar. Dann gilt:

$$\frac{\partial}{\partial s} \tilde{f}(s) = -\mathcal{L}\{[\cdot] \cdot f\} \quad (2.1.12)$$

8. *Differentiationssatz*: Sei $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ differenzierbar und f' genüge den Bedingungen in 2.1.1.1. Dann ist auch f Laplaceintegrierbar und der Grenzwert

$$\lim_{t \rightarrow 0_+} f(t) = f(0_+) \quad (2.1.13)$$

existiert. Falls das Laplaceintegral (2.1.2) von f' für ein s konvergiert, so konvergiert auch das Laplaceintegral für f in diesem s und es gilt:

$$\mathcal{L}\{f'\} = s \cdot \tilde{f}(s) - f(0_+) \quad (2.1.14)$$

¹Dieser könnte auch die Werte $\pm\infty$ annehmen.

2.1.2. Die Laplacerücktransformation

2.1.2.1. Die Umkehrformel Sei $f(t)$ eine Laplacetransformierbare Funktion wie in Definition 2.1.1.1 und s_a die zugehörige Abszisse absoluter Konvergenz. Sei $c > s_a$ beliebig. Dann gilt folgende Umkehrformel für die Laplacetransformation (vgl. [Häfner et al., 1992, S. 78]):

$$\begin{aligned} f(t) &= \mathcal{L}^{-1} \{ \tilde{f} \} \\ &= \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} \tilde{f}(s) e^{st} ds \end{aligned} \quad (2.1.15)$$

Dass dies stimmt, kann man mittels der Fouriertransformation beweisen, welche ich hier nicht entwickeln möchte. Die Gleichheit in (2.1.15) stimmt allerdings, der Natur von Integralgleichungen nach, nur fast überall im Sinne des Riemannintegrals. Für wenigstens stetige Funktionen erhält man also Eindeutigkeit. Die Umkehrformel ist zur Implementation allerdings weniger geeignet, da man die Funktion im Laplacebereich an echt komplexen Werten berechnen und dazu ein Integral mit unendlich langem Integrationsweg approximieren müsste.

Es gibt mehrere Möglichkeiten, eine Funktion numerisch zurück zu transformieren. Zum Beispiel über den Residuensatz, mit dem man das Integral (2.1.15) alternativ auswerten könnte, oder die Entwicklung der Funktion \tilde{f} in elementare Funktionen, welche sich einfacher rücktransformieren lassen. Da ich die gesuchte Lösungsfunktion der Grundwassergleichung allerdings im Laplacebereich in eine Potenzreihe bezüglich r entwickle und s lediglich als Parameter betrachte, habe ich mich für den Stehfest Algorithmus entschieden.

2.1.2.2. Stehfest Algorithmus Um die Lösung der Differentialgleichung aus dem Laplacebereich zurück zu transformieren verwende ich, auf die Empfehlung in [Cheng et al., 1994] hin, den Stehfest Algorithmus (vgl. [Stehfest, 1970]), welcher auf den deutschen Mathematiker Harald Stehfest zurückgeht.

Für eine Funktion $\tilde{f}(s)$ im Laplace-Raum erfolgt die Berechnung der Inversen $f(t)$ dabei wie folgt:

$$f_N(t) = \frac{\ln 2}{t} \cdot \sum_{n=1}^N c_n \cdot \tilde{f}\left(\frac{n \cdot \ln 2}{t}\right) \quad (2.1.16)$$

$$c_n = (-1)^{n+\frac{N}{2}} \cdot \sum_{k=\lfloor \frac{n+1}{2} \rfloor}^{\min\{n, \frac{N}{2}\}} \frac{k^{\frac{N}{2}+1} \cdot \binom{2k}{k}}{\left(\frac{N}{2} - k\right)! \cdot (n-k)! \cdot (2k-n)!} \quad (2.1.17)$$

Dabei ist N ein gerade natürliche Zahl. Diese Reihe konvergiert im Allgemeinen nicht für $N \rightarrow \infty$, da es sich um einen probabilistischen Ansatz handelt. Aber für Anwendungszwecke empfiehlt sich eine Wahl im folgenden Bereich:

$$N = 6, 8, 10, 12, 14, 16, 18, 20 \quad (2.1.18)$$

Nach den Analysen in [Cheng et al., 1994] ist dieser Algorithmus für Funktionen, wie sie in diesem Kontext auftreten, recht effektiv, da auch die Brunnenfunktion $W(t) = E_1\left(\frac{1}{t}\right)$ getestet wurde (vgl. Abschnitt 2.3). Somit kann man für den homogenen Fall numerische und analytische Inversion einfach miteinander vergleichen. Da die gesuchte Lösung der Grundwassergleichung im Allgemeinen dem Verlauf der Brunnenfunktion folgt, soll dies als Argument für die Verwendung dieses Algorithmus herhalten.

2.2. Frobenius-Methode

Nach der Transformation der partiellen Differentialgleichung (1.2.19) in den Laplaceraum (vgl. Abschnitt 3.1) besteht die Notwendigkeit eine gewöhnliche Differentialgleichung zweiter Ordnung mit nicht konstanten Koeffizienten zu lösen:

$$r^2 \cdot g''(r) + r \cdot \alpha(r) g'(r) + \beta(r) g(r) = 0 \quad (2.2.1)$$

Da in vorliegendem Fall $\frac{\alpha(r)}{r}$ eine Singularität in $r = 0$ hat, funktioniert ein normaler Potenzreihenansatz nicht und man kann auch keinen numerischen Löser dafür verwenden. Nach einer Idee von Ferdinand G. Frobenius kann man in solch einem Falle eine *verallgemeinerte Potenzreihe* als Ansatz für eine Fundamentalsystem nehmen. In diesem Abschnitt folge ich dem Vorgehen in [Teschl, 2012, S. 134ff.].

2.2.1. Berechnung eines Fundamentalsystems

Eine verallgemeinerte Potenzreihe im Sinne der Frobenius-Methode ist wie folgt gegeben:

$$g(r) = r^\nu \sum_{k=0}^{\infty} g^{(k)} \cdot r^k \quad (2.2.2)$$

$$\Rightarrow g'(r) = r^\nu \sum_{k=0}^{\infty} (k + \nu) \cdot g^{(k)} \cdot r^{k-1} \quad (2.2.3)$$

$$\Rightarrow g''(r) = r^\nu \sum_{k=0}^{\infty} (k + \nu - 1)(k + \nu) \cdot g^{(k)} \cdot r^{k-2} \quad (2.2.4)$$

Dabei ist $\nu \in \mathbb{C}$ eine beliebige komplexe Zahl, welche noch zu determinieren ist. Seien $\alpha(r)$ und $\beta(r)$ in Potenzreihen um $r = 0$ entwickelbar:

$$\alpha(r) = \sum_{k=0}^{\infty} \alpha^{(k)} \cdot r^k \quad \beta(r) = \sum_{k=0}^{\infty} \beta^{(k)} \cdot r^k \quad (2.2.5)$$

Für die einzelnen Summanden der Differentialgleichung (2.2.1) ergibt sich damit:

$$r^2 \cdot g(r) = r^\nu \sum_{k=0}^{\infty} (k + \nu - 1) \cdot (k + \nu) \cdot g^{(k)} \cdot r^k \quad (2.2.6)$$

$$r \cdot \alpha(r) \cdot g'(r) = r^\nu \sum_{k=0}^{\infty} \left(\sum_{j=0}^k \alpha^{(k-j)} (j + \nu) g^{(j)} \right) \cdot r^k \quad (2.2.7)$$

$$\beta(r) \cdot g'(r) = r^\nu \sum_{k=0}^{\infty} \left(\sum_{j=0}^k \beta^{(k-j)} g^{(j)} \right) \cdot r^k \quad (2.2.8)$$

Setzt man (2.2.6), (2.2.7) und (2.2.8) in die Differentialgleichung (2.2.1) ein, so erhält man:

$$0 = r^\nu \sum_{k=0}^{\infty} \left((k + \nu - 1)(k + \nu) \cdot g_s^{(k)} \right) \quad (2.2.9)$$

$$\begin{aligned} & + \left(\sum_{j=0}^k \alpha^{(k-j)} (j + \nu) g^{(j)} \right) + \left(\sum_{j=0}^k \beta^{(k-j)} g^{(j)} \right) \Big) \cdot r^k \\ & = r^\nu \sum_{k=0}^{\infty} \left(\left((k + \nu) \left((k + \nu - 1) + \alpha^{(0)} \right) + \beta^{(0)} \right) g^{(k)} \right. \\ & \quad \left. + \sum_{j=0}^{k-1} \left(\alpha^{(k-j)} (j + \nu) + \beta^{(k-j)} \right) g^{(j)} \right) r^k \end{aligned} \quad (2.2.10)$$

Den Term vor $g^{(k)}$ in (2.2.10) nennt man *indiziales Polynom*, welches ich mit $I(\nu)$ bezeichne:

$$I(\nu) := \nu(\nu - 1 + \alpha^{(0)}) + \beta^{(0)} \quad (2.2.11)$$

Aus Gleichung (2.2.10) kann man eine Rekursionsformel für die Koeffizienten $g_s^{(k)}$ konstruieren, da nach dem Vergleichssatz für Potenzreihen alle Reihenkoeffizienten in (2.2.10) gleich 0 sein müssen:

$$\forall_{k \in \mathbb{N}} 0 = I(k + \nu) \cdot g^{(k)} + \sum_{j=0}^{k-1} \left(\alpha^{(k-j)}(j + \nu) + \beta^{(k-j)} \right) g^{(j)} \quad (2.2.12)$$

Umstellen nach $g^{(k)}$ ergibt folgende Rekursionsgleichung:

$$g^{(k)} = -\frac{1}{I(k + \nu)} \cdot \sum_{j=0}^{k-1} \left(\alpha^{(k-j)}(j + \nu) + \beta^{(k-j)} \right) g^{(j)} \quad (2.2.13)$$

wobei $g_s^{(0)}$ vorerst frei gewählt werden kann. Die erhaltene verallgemeinerte Potenzreihe $g_s(r)$ löst die Gleichung:

$$r^2 \cdot g''(r) + r \cdot \alpha(r) g'(r) + \beta(r) g(r) = I(\nu) \cdot g^{(0)} \cdot r^\nu \quad (2.2.14)$$

Man sucht also die Nullstellen des indizialen Polynoms² um die ursprüngliche Differentialgleichung zu lösen. Diese *charakteristischen Exponenten* sind wie folgt gegeben:

$$\nu_{1,2} = \frac{1 - \alpha^{(0)}}{2} \pm \sqrt{\frac{(1 - \alpha^{(0)})^2}{4} - \beta^{(0)}} \quad (2.2.15)$$

Damit vereinfacht sich das indiziale Polynom zu $I(\nu) = (\nu - \nu_1) \cdot (\nu - \nu_2)$ und ich setze im Folgenden $g^{(0)} = 1$ als Anfangsbedingung für die Basislösungen. Falls³ $\nu_1 - \nu_2 \notin \mathbb{N}_0$, so ergeben sich formal zwei linear unabhängige Basislösungen:

$$u_1(r) = r^{\nu_1} \cdot \sum_{k=0}^{\infty} g_1^{(k)} \cdot r^k \quad (2.2.16)$$

$$g_1^{(k)} = -\frac{1}{k(k + \nu_1 - \nu_2)} \cdot \sum_{j=0}^{k-1} \left(\alpha^{(k-j)}(j + \nu_1) + \beta^{(k-j)} \right) g_1^{(j)} \quad (2.2.17)$$

$$u_2(r) = r^{\nu_2} \cdot \sum_{k=0}^{\infty} g_2^{(k)} \cdot r^k \quad (2.2.18)$$

$$g_2^{(k)} = -\frac{1}{k(k + \nu_2 - \nu_1)} \cdot \sum_{j=0}^{k-1} \left(\alpha^{(k-j)}(j + \nu_2) + \beta^{(k-j)} \right) g_2^{(j)} \quad (2.2.19)$$

Falls jedoch gilt

$$m := \nu_1 - \nu_2 \in \mathbb{N} \quad (2.2.20)$$

so erhält man in Gleichung (2.2.19) für $k = m$ eine Division durch 0. Die $g_2^{(k)}$ sind für $k = 1 \dots (m-1)$ durch die Gleichung (2.2.19) dennoch eindeutig bestimmt. Falls für $k = m$ aber gilt:

$$0 = \sum_{j=0}^{m-1} \left(\alpha^{(m-j)}(j + \nu_2) + \beta^{(m-j)} \right) g_2^{(j)} \quad (2.2.21)$$

²Man könnte auch $g^{(0)} = 0$ setzen, womit man allerdings die Nulllösung generiert.

³Ich nehme ohne Einschränkung der Allgemeinheit $\Re(\nu_1) \geq \Re(\nu_2)$ an.

so kann man $g_2^{(m)}$ beliebig wählen. Dies bedeutet nur, dass man zu u_2 ein beliebiges Vielfaches von u_1 hinzuaddieren kann und dennoch zwei linear unabhängige Basislösungen erhält. Die restlichen $g_2^{(k)}$ sind dann wieder eindeutig durch (2.2.19) definiert. Falls (2.2.21) nicht erfüllt ist, so kann es in diesem Falle keine zweite Lösung der Form (2.2.18) geben. Im Falle $\nu_1 = \nu_2$ ist dies ebenso der Fall, da dann u_1 und u_2 identisch wären. Um dann dennoch eine zweite Basislösung zu erhalten, nutzt man eine Art *Variation der Konstanten* in folgender Weise:

$$u_2(r) = \underbrace{r^{\nu_2} \cdot \sum_{k=0}^{\infty} g_2^{(k)} \cdot r^k}_{:=h(r)} + c \cdot \ln(r) \cdot u_1(r) \quad (2.2.22)$$

wobei $c \in \mathbb{C}$ noch zu determinieren ist. Setzt man (2.2.22) in die Differentialgleichung (2.2.1) ein, so erhält man:

$$\begin{aligned} & r^2 \cdot \left(h''(r) + c \cdot \left(\ln(r) \cdot u_1''(r) + 2 \cdot \frac{u_1'(r)}{r} - \frac{u_1(r)}{r^2} \right) \right) \\ & + r \cdot \alpha(r) \cdot \left(h'(r) + c \cdot \left(\ln(r) \cdot u_1'(r) + \frac{u_1(r)}{r} \right) \right) \\ & + \beta(r) \cdot (h(r) + c \cdot \ln(r) \cdot u_1(r)) = 0 \end{aligned} \quad (2.2.23)$$

$$\begin{aligned} \Leftrightarrow & \quad r^2 \cdot h''(r) + r \cdot \alpha(r) \cdot h'(r) + \beta(r) \cdot h(r) \\ & + c \cdot (2 \cdot r \cdot u_1'(r) + (\alpha(r) - 1) \cdot u_1(r)) \\ & + c \cdot \ln(r) \cdot \underbrace{(r^2 \cdot u_1''(r) + r \cdot \alpha(r) \cdot u_1'(r) + \beta(r) \cdot u_1(r))}_{=0} = 0 \end{aligned} \quad (2.2.24)$$

Der letzte Term in (2.2.24) verschwindet, da u_1 ein Lösung der Differentialgleichung (2.2.1) ist. Man erhält für h somit folgende Differentialgleichung:

$$r^2 \cdot h''(r) + r \cdot \alpha(r) h'(r) + \beta(r) h(r) = -c (2r \cdot u_1'(r) + (\alpha(r) - 1) \cdot u_1(r)) \quad (2.2.25)$$

Entwickelt man beide Seiten von (2.2.25) in Potenzreihen, so erhält man analog zu (2.2.10) und unter Beachtung, dass $\nu_1 = \nu_2 + m$:

$$\text{LHS} = r^{\nu_2} \sum_{k=0}^{\infty} \left(I(k + \nu_2) g_2^{(k)} + \sum_{j=0}^{k-1} \left(\alpha^{(k-j)} (j + \nu_2) + \beta^{(k-j)} \right) g_2^{(j)} \right) r^k \quad (2.2.26)$$

$$\begin{aligned} \text{RHS} &= -c \left(\sum_{k=0}^{\infty} 2(k + \nu_1) \cdot g_1^{(k)} \cdot r^{k+\nu_1} \right. \\ & \quad \left. + \sum_{k=0}^{\infty} \left(\sum_{j=0}^k \alpha^{(k-j)} g_1^{(j)} \right) \cdot r^{k+\nu_1} - \sum_{k=0}^{\infty} g_1^{(k)} \cdot r^{k+\nu_1} \right) \\ &= -cr^{\nu_2} \sum_{k=m}^{\infty} \left((2(k + \nu_2) + \alpha^{(0)} - 1) g_1^{(k-m)} + \sum_{j=0}^{k-m-1} \alpha^{(k-m-j)} g_1^{(j)} \right) r^k \end{aligned} \quad (2.2.27)$$

Da ν_1 und ν_2 Nullstellen von $I(\nu)$ sind, erhält man aus (2.2.11):

$$\begin{aligned} \alpha^{(0)} - 1 &= -\nu_1 - \nu_2 \\ &= -m - 2\nu_2 \end{aligned}$$

Setzt man dies in (2.2.27) ein, so erhält man für die rechte Seite von (2.2.25):

$$\text{RHS} = -cr^{\nu_2} \sum_{k=m}^{\infty} \left((2k - m) g_1^{(k-m)} + \sum_{j=0}^{k-m-1} \alpha^{(k-m-j)} g_1^{(j)} \right) r^k \quad (2.2.28)$$

Nun muss man (2.2.26) und (2.2.28) wieder gliedweise vergleichen um eine Rekursionsformel für die $g_2^{(k)}$ zu erhalten. Ich setze wieder $g_2^{(0)} = 1$ als Anfangsbedingung für diese Basislösung. Für $k < m$ ergibt sich genau wie in Gleichung (2.2.19):

$$g_2^{(k)} = -\frac{1}{k(k-m)} \cdot \sum_{j=0}^{k-1} \left(\alpha^{(k-j)}(j + \nu_2) + \beta^{(k-j)} \right) g_2^{(j)} \quad (2.2.29)$$

Für $k = m$ erhält man:

$$\sum_{j=0}^{m-1} \left(\alpha^{(m-j)}(j + \nu_2) + \beta^{(m-j)} \right) g_2^{(j)} = -cm \quad (2.2.30)$$

Damit ist c für $m \neq 0$ eindeutig bestimmt durch:

$$c = \frac{1}{m} \cdot \sum_{j=0}^{m-1} \left(\alpha^{(m-j)}(j + \nu_2) + \beta^{(m-j)} \right) g_2^{(j)} \quad (2.2.31)$$

Falls $m = 0$, also $\nu_1 = \nu_2$ gilt, so kann man c beliebig aus $\mathbb{C} \setminus \{0\}$ wählen, da dann auch u_1 und h verallgemeinerte Potenzreihen vom selben Typ sind. $g_2^{(m)}$ kann im Fall $m \neq 0$ wieder beliebig gewählt werden und für $m = 0$ entspricht dies nur einem beliebigen Anfangswert, der auch als 0 gewählt werden könnte. Für $k > m$ erhält man schließlich:

$$g_2^{(k)} = -\frac{1}{k(k-m)} \cdot \left(\sum_{j=0}^{k-1} \left(\alpha^{(k-j)}(j + \nu_2) + \beta^{(k-j)} \right) g_2^{(j)} + c \left((2k-m)g_1^{(k-m)} + \sum_{j=0}^{k-m-1} \alpha^{(k-m-j)} g_1^{(j)} \right) \right) \quad (2.2.32)$$

Die angestellte Berechnung des obigen Fundamentalsystems für die Differentialgleichung (2.2.1) war bisher rein formal. Es ist noch zu zeigen, dass die entstehenden Potenzreihen wirklich einen positiven Konvergenzradius haben. Dass dies der Fall ist, wurde von Lazarus I. Fuchs im Jahre 1866 bewiesen (siehe [Fuchs, 1866]). Ich trage alle Resultat in folgendem Satz zusammen:

2.2.2. Satz von Fuchs

Seien die Koeffizientenfunktionen $\alpha(r)$ und $\beta(r)$ der Differentialgleichung (2.2.1) in einer Umgebung von 0 holomorph mit den Konvergenzradien R_α und R_β . Seien ν_1 und ν_2 die komplexen Nullstellen des indizialen Polynoms (2.2.11) wobei $\Re(\nu_1) \geq \Re(\nu_2)$ gelte. Es ergeben sich zwei Fälle:

1. Falls $\nu_1 - \nu_2 \notin \mathbb{N}_0$ gilt, so gibt es ein Fundamentalsystem der Form:

$$u_i(r) = r^{\nu_i} g_i(r) \quad (i = 1, 2) \quad (2.2.33)$$

Wobei die g_1 und g_2 holomorphe Funktionen um 0 sind mit $g_{1,2}(0) = 1$.

2. Falls $\nu_1 - \nu_2 \in \mathbb{N}_0$ gilt, so gibt es ein Fundamentalsystem der Form:

$$u_1(r) = r^{\nu_1} g_1(r) \quad (2.2.34)$$

$$u_2(r) = r^{\nu_2} g_2(r) + c \cdot \ln(r) \cdot g_1(r) \quad (2.2.35)$$

Wobei die g_1 und g_2 wieder holomorphe Funktionen um 0 sind mit $g_{1,2}(0) = 1$. Die Konstante $c \in \mathbb{C}$ ist dabei für $\nu_1 \neq \nu_2$ durch Gleichung (2.2.31) bestimmt oder kann im Fall (2.2.21) gleich 0 gewählt werden. Für $\nu_1 = \nu_2$ kann c beliebig aus $\mathbb{C} \setminus \{0\}$ gewählt werden.

In beiden Fällen haben die g_i einen positiven Konvergenzradius R mit der Eigenschaft:

$$R \geq \min \{R_\alpha, R_\beta\} \quad (2.2.36)$$

Beweis: Die Funktion u_1 und u_2 sind ihrer Konstruktion nach offensichtlich unabhängig. Es ist also nur noch zu zeigen, dass der Konvergenzradius von g_1 wirklich positiv ist. Da ebenfalls nach Konstruktion g_2 in Gleichung (2.2.35) aus g_1 hervorgeht, sind die Konvergenzradien auch in diesem Falle identisch. Sei $R \in (0, \min \{R_\alpha, R_\beta\})$ beliebig. Seien die $g_1^{(k)}$ wie in Gleichung (2.2.17) definiert. Ich zeige:

$$\forall_{k \geq k_0} \left| g_1^{(k)} \right| \cdot R^k \leq \varepsilon \quad (2.2.37)$$

Für ein $\varepsilon > 0$. Damit ist dann der Beweis erbracht. Ich definiere:

$$A := \sum_{k=0}^{\infty} \left| \alpha^{(k)} \right| R^k$$

$$B := \sum_{k=0}^{\infty} \left| \beta^{(k)} \right| R^k$$

Diese Grenzwerte existieren, da $R < \min \{R_\alpha, R_\beta\}$ gewählt wurde. Dann gibt es ein $k_0 \in \mathbb{N}$ sodass für alle $k > k_0$:

$$\frac{(|\nu_1| + k) A + B}{(\Re(\nu_1 - \nu_2) + k) k} \leq 1$$

Somit wähle man ε wie folgt:

$$\varepsilon := \max_{k=0 \dots k_0} \left| g_1^{(k)} \right| \cdot R^k$$

Jetzt gilt die Behauptung schon für $k \leq k_0$. Für $k \geq k_0$ schließen wir per Induktion. Falls die Behauptung (2.2.37) für $k-1$ gilt, dann folgt für k aus (2.2.17):

$$\begin{aligned} \left| g_1^{(k)} \right| \cdot R^k &= \left| \frac{1}{k(k + \nu_1 - \nu_2)} \cdot \sum_{j=0}^{k-1} \left(\alpha^{(k-j)} (j + \nu_1) + \beta^{(k-j)} \right) g_1^{(j)} \right| R^k \\ &\leq \frac{1}{k(k + \Re(\nu_1 - \nu_2))} \cdot \sum_{j=0}^{k-1} \left(\left| \alpha^{(k-j)} \right| (j + |\nu_1|) + \left| \beta^{(k-j)} \right| \right) \frac{\varepsilon}{R^j} R^k \\ &\leq \frac{1}{k(k + \Re(\nu_1 - \nu_2))} \cdot \sum_{j=0}^{k-1} \left(\left| \alpha^{(k-j)} \right| (k + |\nu_1|) + \left| \beta^{(k-j)} \right| \right) R^{k-j} \cdot \varepsilon \\ &\leq \frac{(|\nu_1| + k) A + B}{k(k + \Re(\nu_1 - \nu_2))} \varepsilon \\ &\leq \varepsilon \end{aligned}$$

Damit ist der Konvergenzradius R_{g_1} von g_1 mindestens R und da $R < \min \{R_\alpha, R_\beta\}$ beliebig war, folgt:

$$R_{g_1} \geq \min \{R_\alpha, R_\beta\}$$

Somit ist alles gezeigt. □

Wir haben jetzt ein Werkzeug, um Lösungen der Differentialgleichung (2.2.1) in einer Umgebung von 0 zu berechnen. Es wird sich zeigen, dass diese Basislösungen ideal geeignet sind, um später die Randbedingung am Pumpbrunnen in die Lösung der Grundwasser-gleichung einzuarbeiten.

Ich möchte als erste Anwendung der Frobenius-Methode die Theis-Lösung für homogene Wasserleiter berechnen und dabei auch leichte Verallgemeinerungen dieser Lösung geben. Im Laplaceraum entsteht in diesem Falle gerade die modifizierte Besselsche Differentialgleichung 0-ter Ordnung. Da ich später auf diese Lösung zurückgreife, lohnt es sich, die Basislösungen dieser Differentialgleichung näher zu betrachten.

2.3. Theis-Lösung

Die Theis-Lösung (siehe [Theis, 1935]), benannt nach dem Hydrogeologen Charles Vernon Theis, ist eine Lösung der Grundwassergleichung für homogene Wasserleiter. Es handelt es sich um die radialsymmetrische Lösung der partiellen Differentialgleichung (1.2.19) mit konstanter Leitfähigkeit K :

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \cdot \frac{\partial h}{\partial r} (r, t) \right) = \frac{S}{K} \frac{\partial h}{\partial t} (r, t) \quad (2.3.1)$$

$$\lim_{r \rightarrow 0^+} r \cdot \frac{\partial h}{\partial r} (r, t) = -\frac{Q_w}{2\pi K L} \quad (2.3.2)$$

$$\lim_{r \rightarrow \infty} h(r, t) = 0 \quad (2.3.3)$$

$$h(r, t = 0) = 0 \quad (2.3.4)$$

Ich möchte die Lösung dieser Differentialgleichung auf zwei Arten geben. Zum einen durch eine geeignete Variablentransformation, der sogenannten Boltzmann-Transformation, die es erlaubt die Lösung direkt zu berechnen. Zum anderen über den Weg der Laplacetransformation und der Frobenius-Methode.

2.3.1. Die Boltzmann-Transformation

Ich folge hier den Betrachtungen in [Häfner et al., 1992, S. 315ff]. Die nach dem österreichischen Physiker Ludwig E. Boltzmann benannte Transformation ist im Grunde eine einfache Variablentransformation für partielle Differentialgleichungen zweier Veränderliche, wobei eine Substitution von dem Verhältnis $\frac{r}{\sqrt{t}}$ abhängt. Ich wähle für den hier behandelten Fall folgende Transformation:

$$\begin{aligned} \varphi : (r, t) &\mapsto (\eta, \xi) \\ \eta &= \frac{r^2 S}{4K \cdot t} \\ \xi &= t \end{aligned} \quad (2.3.5)$$

Als Jacobimatrix dieser Transformation erhält man:

$$\frac{\partial(\eta, \xi)}{\partial(r, t)} = \begin{pmatrix} \frac{rS}{2K \cdot t} & -\frac{r^2 S}{4K \cdot t^2} \\ 0 & 1 \end{pmatrix} \quad (2.3.6)$$

Somit ist die Funktionaldeterminante gegeben mit:

$$\left| \frac{\partial(\eta, \xi)}{\partial(r, t)} \right| = \frac{rS}{2K \cdot t} \quad (2.3.7)$$

und diese ist für $r, t > 0$ immer positiv und die Transformation ist bijektiv für $\varphi : \mathbb{R}_+^2 \rightarrow \mathbb{R}_+^2$, womit es sich um eine zulässige Variablentransformation handelt. Die Rücktransformation ist gegeben mit:

$$\begin{aligned} \psi : (\eta, \xi) &\mapsto (r, t) \\ r &= \sqrt{\eta \xi} \cdot 2\sqrt{\frac{K}{S}} \\ t &= \xi \end{aligned} \quad (2.3.8)$$

Damit erhalten wir eine neue Funktion \tilde{h} :

$$\tilde{h}(\eta, \xi) = h(r(\eta, \xi), t(\eta, \xi)) \quad (2.3.9)$$

Die Differentialoperatoren transformieren sich wie folgt:

$$\begin{aligned}\frac{\partial}{\partial r} &= \frac{\partial \eta}{\partial r} \frac{\partial}{\partial \eta} + \frac{\partial \xi}{\partial r} \frac{\partial}{\partial \xi} \\ &= \frac{rS}{2K \cdot t} \frac{\partial}{\partial \eta}\end{aligned}\quad (2.3.10)$$

$$\begin{aligned}\frac{\partial}{\partial t} &= \frac{\partial \eta}{\partial t} \frac{\partial}{\partial \eta} + \frac{\partial \xi}{\partial t} \frac{\partial}{\partial \xi} \\ &= -\frac{r^2 S}{4K \cdot t^2} \frac{\partial}{\partial \eta} + \frac{\partial}{\partial \xi}\end{aligned}\quad (2.3.11)$$

Die Besonderheit dieser Transformation ist, dass die Differentialgleichung und die Randbedingungen unabhängig von ξ beschrieben werden können und man somit die Annahme

$$\frac{\partial}{\partial \xi} \tilde{h}(\eta, \xi) = 0 \quad (2.3.12)$$

machen kann, womit sich die partielle Differentialgleichung (2.3.1) auf eine gewöhnliche Differentialgleichung in η reduziert. Für die Anfangsbedingung am Brunnen (2.3.2) ergibt sich:

$$\begin{aligned}r \cdot \frac{\partial h}{\partial r}(r, t) &= r \cdot \frac{rS}{2K \cdot t} \frac{\partial \tilde{h}}{\partial \eta}(\eta, \xi) \\ &= 2\eta \frac{\partial \tilde{h}}{\partial \eta}(\eta, \xi)\end{aligned}\quad (2.3.13)$$

Da $\eta(r, t) \xrightarrow[r \rightarrow 0]{} 0$ gilt, ist diese Randbedingung äquivalent zu:

$$\lim_{\eta \rightarrow 0^+} \eta \frac{\partial \tilde{h}}{\partial \eta}(\eta, \xi) = -\frac{Q_w}{4\pi KL} \quad (2.3.14)$$

Weiter gilt:

$$\begin{aligned}\forall_{t>0} \eta(r, t) &\xrightarrow[r \rightarrow \infty]{} \infty \\ \forall_{r>0} \eta(r, t) &\xrightarrow[t \rightarrow 0]{} \infty\end{aligned}\quad (2.3.15)$$

Somit kann man die Randbedingung (2.3.3) und die Anfangsbedingung (2.3.4) in einer Randbedingung in Abhängigkeit von η beschreiben:

$$\lim_{\eta \rightarrow \infty} \tilde{h}(\eta, \xi) = 0 \quad (2.3.16)$$

Die Annahme (2.3.12) ist somit gerechtfertigt und die Ableitungsoperatoren vereinfachen sich zu:

$$\frac{\partial}{\partial r} = \frac{2}{r} \eta \frac{\partial}{\partial \eta} \quad (2.3.17)$$

$$\frac{\partial}{\partial t} = -\frac{1}{t} \eta \frac{\partial}{\partial \eta} \quad (2.3.18)$$

Die Differentialgleichung (2.3.1) transformiert sich wie folgt:

$$\begin{aligned}\frac{1}{r} \frac{\partial}{\partial r} \left(r \cdot \frac{\partial h}{\partial r}(r, t) \right) &= \frac{S}{K} \frac{\partial h}{\partial t}(r, t) \\ \Leftrightarrow \frac{S}{2K \cdot t} \frac{\partial}{\partial \eta} \left(2\eta \frac{\partial \tilde{h}}{\partial \eta}(\eta) \right) &= -\frac{S}{K} \cdot \frac{1}{t} \eta \frac{\partial \tilde{h}}{\partial \eta}(\eta) \\ \Leftrightarrow \frac{\partial \tilde{h}}{\partial \eta}(\eta) + \eta \frac{\partial^2 \tilde{h}}{\partial \eta^2}(\eta) &= -\eta \frac{\partial \tilde{h}}{\partial \eta}(\eta)\end{aligned}\quad (2.3.19)$$

Damit erhält man schließlich folgende gewöhnliche Differentialgleichung erste Ordnung für $g(\eta) = \frac{\partial \tilde{h}}{\partial \eta}(\eta)$:

$$g'(\eta) = -\left(1 + \frac{1}{\eta}\right)g(\eta) \quad (2.3.20)$$

Deren Lösung ist wiederum gegeben durch:

$$\begin{aligned} g(\eta) &= \exp\left(-\int \left(1 + \frac{1}{\eta}\right) d\eta + C\right) \\ &= C_1 \cdot \frac{e^{-\eta}}{\eta} \end{aligned} \quad (2.3.21)$$

Die Konstante C_1 kann man durch Einsetzen der Randbedingung aus (2.3.14) determinieren:

$$C_1 = -\frac{Q_w}{4\pi KL} \quad (2.3.22)$$

Integration von (2.3.21) ergibt:

$$\tilde{h}(\eta) = -\frac{Q_w}{4\pi KL} \int_1^{\eta} \frac{e^{-\tau}}{\tau} d\tau + C_2 \quad (2.3.23)$$

Einsetzen der zweiten Randbedingung (2.3.16) liefert:

$$C_2 = \frac{Q_w}{4\pi KL} \int_1^{\infty} \frac{e^{-\tau}}{\tau} d\tau \quad (2.3.24)$$

Und somit erhält man insgesamt:

$$h(r, t) = \frac{Q_w}{4\pi KL} \int_{\eta}^{\infty} \frac{e^{-\tau}}{\tau} d\tau \quad \eta = \frac{r^2 S}{4Kt} \quad (2.3.25)$$

Dabei ist zu bemerken, dass dies, wie in [Abramowitz et al., 1972, S. 228] nachzulesen, gerade die Integralexponentialfunktion E_1 ist

$$E_1(\eta) = \int_{\eta}^{\infty} \frac{e^{-\tau}}{\tau} d\tau \quad (2.3.26)$$

Diese Funktion wird wegen ihrer Lösungseigenschaft der Grundwassergleichung auch *Brunnenfunktion* genannt (vgl. [Häfner et al., 1992, S. 584]).

2.3.2. Lösung mittels Laplace-Transformation

Ich transformiere die Differentialgleichung (2.3.1) bezüglich t in den Laplaceraum und erhalten unter Verwendung des Differentiationssatzes (2.1.9) und der Anfangsbedingung (2.3.4) mit $g_s(r) := \mathcal{L}\{h(r, \cdot)\}(s)$:

$$\begin{aligned} \frac{1}{r} \frac{\partial}{\partial r} \left(r \cdot \frac{\partial g_s}{\partial r}(r) \right) &= \frac{S}{K} \cdot s \cdot g_s(r) \\ \Leftrightarrow g_s''(r) + \frac{1}{r} g_s'(r) &= \frac{S}{K} \cdot s \cdot g_s(r) \\ \Leftrightarrow r^2 \cdot g_s''(r) + r \cdot g_s'(r) - r^2 \frac{s \cdot S}{K} \cdot g_s(r) &= 0 \end{aligned} \quad (2.3.27)$$

Die Differentialgleichung (2.3.27) hat somit schon genau die benötigte Form (2.2.1) für die Frobenius-Methode. Ich substituiere allerdings noch r in folgender Art und Weise:

$$\tau := T_s \cdot r \quad T_s = \sqrt{\frac{s \cdot S}{K}} \quad (2.3.28)$$

$$\tilde{g}(\tau) = g\left(\frac{\tau}{T_s}\right) \quad (2.3.29)$$

Damit erhält man:

$$g'(r) = \tilde{g}'(\tau) \cdot T_s \quad (2.3.30)$$

$$g''(r) = \tilde{g}''(\tau) \cdot T_s^2 \quad (2.3.31)$$

Setzt man dies in (2.3.27) ein, erhält man:

$$\tau^2 \tilde{g}''(\tau) + \tau \tilde{g}'(\tau) - \tau^2 \tilde{g}(\tau) = 0 \quad (2.3.32)$$

Und dies ist gerade die modifizierte Besselsche Differentialgleichung 0-ter Ordnung. Allgemein ist die modifizierte Besselsche Differentialgleichung der Ordnung $\lambda \in \mathbb{C}$ gegeben mit (vgl. [Abramowitz et al., 1972, S. 374]):

$$\tau^2 \tilde{g}''(\tau) + \tau \tilde{g}'(\tau) - (\tau^2 + \lambda^2) \tilde{g}(\tau) = 0 \quad (2.3.33)$$

Diese möchte ich für $\lambda = 0$ mit der Frobenius-Methode lösen. Für die Koeffizientenfunktionen erhält man:

$$\alpha(\tau) = 1 \quad (2.3.34)$$

$$\beta(\tau) = -\tau^2 \quad (2.3.35)$$

Somit ergibt sich als indiziales Polynom nach Gleichung (2.2.11):

$$I(\nu) = \nu^2 \quad (2.3.36)$$

Damit sind die charakteristische Exponenten identisch und gleich 0:

$$\nu_{1,2} = 0 \quad (2.3.37)$$

Nach dem Satz von Fuchs gibt es also ein Fundamentalsystem der Form:

$$u_1(\tau) = \sum_{k=0}^{\infty} u_1^{(k)} \cdot \tau^k \quad (2.3.38)$$

$$u_2(\tau) = \sum_{k=0}^{\infty} u_2^{(k)} \cdot \tau^k + c \cdot \ln(\tau) \cdot u_1(\tau) \quad (2.3.39)$$

Ich berechne zuerst u_1 . Mit dem Anfangswert $u_1^{(0)} = 1$ ergibt sich aus Formel (2.2.17):

$$\begin{aligned} u_1^{(k)} &= -\frac{1}{k^2} \cdot \sum_{j=0}^{k-1} (\alpha^{(k-j)} \cdot j + \beta^{(k-j)}) u_1^{(j)} \\ &= \frac{1}{k^2} \cdot u_1^{(k-2)} \end{aligned} \quad (2.3.40)$$

Somit sind alle $u_1^{(2j+1)} = 0$ und für die geraden gilt:

$$u_1^{(2k)} = \frac{1}{4^k (k!)^2} \quad (2.3.41)$$

Damit erhält man folgende Potenzreihe:

$$u_1(\tau) = \sum_{k=0}^{\infty} \frac{\tau^{2k}}{4^k (k!)^2} \quad (2.3.42)$$

Dies entspricht gerade der Potenzreihe der ersten modifizierten Besselfunktion $I_0(\tau)$ (vgl. [Abramowitz et al., 1972, S. 375]). Für die zweite Basislösung erhält man mit der Wahl $c = -1$ und $u_2^{(0)} = 0$ aus Gleichung (2.2.32):

$$\begin{aligned} u_2^{(k)} &= -\frac{1}{k^2} \left(-u_2^{(k-2)} - \left(2ku_1^{(k)} + \sum_{j=0}^{k-1} \alpha^{(k-j)} u_1^{(j)} \right) \right) \\ &= \frac{1}{k^2} \cdot u_2^{(k-2)} + \frac{2}{k} u_1^{(k)} \end{aligned} \quad (2.3.43)$$

Somit sind auch hier wieder alle $u_2^{(2j+1)} = 0$ und es gilt für die geraden Koeffizienten:

$$u_2^{(2k)} = \frac{1}{4k^2} u_2^{(2(k-1))} + \frac{1}{4^k (k!)^2} \quad (2.3.44)$$

Zur Abkürzung definiere ich für die Partialsumme bis k der harmonischen Reihe:

$$H_k = \sum_{j=1}^k \frac{1}{j} \quad (2.3.45)$$

Und damit erhält man für $k \in \mathbb{N}_0$:

$$u_2^{(2k)} = \frac{H_k}{4^k (k!)^2} \quad (2.3.46)$$

Die zweite Basislösung ist somit gegeben durch:

$$u_2(\tau) = \sum_{k=1}^{\infty} \frac{H_k}{4^k (k!)^2} \tau^{2k} - \ln \tau \cdot u_1(\tau) \quad (2.3.47)$$

Ganz ähnlich sieht auch die modifizierte Besselfunktion K_0 aus, welche in der Literatur (vgl. [Abramowitz et al., 1972, S. 375]) häufig als zweite Basislösung angegeben wird. Sie unterscheidet sich von unserer Lösung u_2 nur durch einen weiteren linearen Anteil⁴ von I_0 :

$$K_0(\tau) = \sum_{k=1}^{\infty} \frac{H_k}{4^k (k!)^2} \tau^{2k} - \left(\ln \frac{\tau}{2} + \gamma \right) \cdot I_0(\tau) \quad (2.3.48)$$

Wobei γ die *Euler-Mascheroni-Konstante* ist:

$$\gamma = \lim_{k \rightarrow \infty} (H_k - \ln(k)) \quad (2.3.49)$$

Somit ist die Lösung im Laplaceraum durch eine Linearkombination von I_0 und K_0 wie folgt gegeben:

$$g_s(r) = A_s \cdot I_0(T_s r) + B_s \cdot K_0(T_s r) \quad (2.3.50)$$

⁴Man beachte, dass $\ln \frac{\tau}{2} = \ln \tau - \ln 2$.

2.3.2.1. Die Standard-Randbedingungen Jetzt gilt es noch die Randbedingungen (2.3.2) und (2.3.3) in die Lösung einarbeiten. Der Anfangswert (2.3.4) wurde durch den Differentiationssatz schon eingepflegt. Mit der Laplacetransformation erhält man die beiden Randwerte:

$$\lim_{r \rightarrow \infty} g_s(r) = 0 \quad (2.3.51)$$

$$\lim_{r \rightarrow 0} r \cdot g'_s(r) = -\frac{Q_w}{2\pi KL} \cdot \frac{1}{s} \quad (2.3.52)$$

Nach [Abramowitz et al., 1972, S. 377f] haben die beiden Basisfunktion für große Werte τ folgendes asymptotisches Verhalten:

$$I_0(\tau) = \frac{e^\tau}{\sqrt{2\pi\tau}} + \mathcal{O}\left(\frac{1}{\tau}\right) \quad (2.3.53)$$

$$K_0(\tau) = \sqrt{\frac{\pi}{2\tau}} e^{-\tau} + \mathcal{O}\left(\frac{1}{\tau}\right) \quad (2.3.54)$$

Somit ergibt die Randbedingung (2.3.51):

$$A_s = 0 \quad (2.3.55)$$

Für die Ableitung von K_0 gilt nach Formel (2.3.48) für kleine Werte von τ :

$$K'_0(\tau) = -\frac{1}{\tau} + \mathcal{O}(\tau) \quad (2.3.56)$$

Somit ergibt sich durch einsetzen der Randbedingung (2.3.52):

$$\begin{aligned} \frac{Q_w}{2\pi KL} \cdot \frac{1}{s} &= -B_s \cdot \lim_{r \rightarrow 0^+} T_s r \cdot K'_0(T_s r) \\ &= B_s \end{aligned} \quad (2.3.57)$$

Also ist die Lösung im Laplaceraum gegeben durch:

$$g_s(r) = \frac{Q_w}{2\pi KL} \cdot \frac{1}{s} \cdot K_0\left(\sqrt{\frac{S}{K}} \cdot \sqrt{s} \cdot r\right) \quad (2.3.58)$$

Diese Funktion ist jetzt in den Zeit-Bereich zurück zu transformieren. Da wir die Lösung im Zeit-Bereich schon mit Gleichung (2.3.25) berechnet haben, erhalten wir folgendes Lemma:

Laplacetransformation von $E_1\left(\frac{1}{t}\right)$: Für die Laplacetransformierte der Funktion

$$E_1\left(\frac{1}{t}\right) = \int_1^\infty \frac{e^{-\frac{\tau}{t}}}{\tau} d\tau \quad (2.3.59)$$

gilt:

$$\mathcal{L}\left\{E_1\left(\frac{1}{t}\right)\right\}(s) = \frac{2}{s} \cdot K_0(2\sqrt{s}) \quad (2.3.60)$$

Beweis: Man setze in Gleichung (2.3.25) sowie in Gleichung (2.3.58):

$$\begin{aligned} \frac{Q_w}{4\pi KL} &= 1 \\ r &= 2\sqrt{\frac{K}{S}} \end{aligned}$$

Die Aussage folgt aus der Eindeutigkeit der Lösung der Differentialgleichung. \square

2.3.2.2. Verallgemeinerte Randbedingungen Die Randbedingungen der Theis-Lösung sind zum einen ein unendlich dünner Brunnen, also eine Linienquelle und zum anderen ein unendlich weit ausgedehnter Wasserleiter. Möchte man die Theis-Lösung aber mit numerischen Simulation vergleichen, d.h. Pumptestdaten auf diskreten Gittern, so ist man auf endliche Randwerte angewiesen. Die Herleitung der Brunnenfunktion in Abschnitt 2.3.1 zeigt, dass man auf die speziellen Randannahmen (2.3.2) und (2.3.3) angewiesen ist. Möchte man allerdings einen Brunnen mit positivem Radius $r_{\text{well}} > 0$ respektive einen endlich weit ausgedehnten Wasserleiter mit Radius $r_{\infty} < \infty$ modellieren, also

$$\frac{\partial h}{\partial r}(r_{\text{well}}, t) = -\frac{Q_w}{2\pi KL \cdot r_{\text{well}}} \quad (2.3.61)$$

$$h(r_{\infty}, t) = 0 \quad (2.3.62)$$

als Randbedingung einarbeiten, so scheitert man an dem Versuch, dies mit der Boltzmann-Transformation zu behandeln. Dem entgegen ist die Lösung (2.3.50) im Laplaceraum dafür bestens geeignet. Mit der Laplacetransformation sind diese Randbedingungen gegeben mit:

$$g'_s(r_{\text{well}}) = -\frac{Q_w}{2\pi KL \cdot r_{\text{well}}} \cdot \frac{1}{s} \quad (2.3.63)$$

$$g_s(r_{\infty}) = 0 \quad (2.3.64)$$

Es ergeben sich somit drei Fälle:

1. $r_{\text{well}} > 0$ und $r_{\infty} = \infty$:

Aus der Randbedingung (2.3.64) folgt genau wie im Falle der Theis-Lösung, dass

$$A_s = 0 \quad (2.3.65)$$

Somit hat die Lösung nur noch die Form

$$g_s(r) = B_s \cdot K_0(T_s r) \quad (2.3.66)$$

Für die Ableitungen von I_0 und K_0 möchte ich nach [Abramowitz et al., 1972, S. 376] folgende Relation verwenden:

$$I'_0(\tau) = I_1(\tau) \quad (2.3.67)$$

$$K'_0(\tau) = -K_1(\tau) \quad (2.3.68)$$

Dabei sind I_1 und K_1 die Basislösungen der modifizierten Besselschen Differentialgleichung zur Ordnung 1. Somit erhält man für die Randbedingung (2.3.63):

$$\begin{aligned} \frac{Q_w}{2\pi KL \cdot r_{\text{well}}} \cdot \frac{1}{s} &= B_s \cdot T_s \cdot K_1(T_s r_{\text{well}}) \\ \Rightarrow B_s &= \frac{Q_w}{2\pi KL \cdot T_s r_{\text{well}} \cdot K_1(T_s r_{\text{well}})} \cdot \frac{1}{s} \end{aligned} \quad (2.3.69)$$

Der Grenzübergang $r_{\text{well}} \rightarrow 0$ ist hier auch gerechtfertigt, da wie in Abschnitt 2.3.2.1 gezeigt, folgende Beziehung gilt:

$$\lim_{r_{\text{well}} \rightarrow 0} T_s r_{\text{well}} \cdot K_1(T_s r_{\text{well}}) = 1$$

Somit ist die Lösung in diesem Falle gegeben mit:

$$g_s(r) = \frac{Q_w}{2\pi KL \cdot T_s r_{\text{well}} \cdot K_1(T_s r_{\text{well}})} \cdot \frac{1}{s} \cdot K_0(T_s r) \quad (2.3.70)$$

2. $r_{\text{well}} = 0$ und $r_{\infty} < \infty$:

Für die Randbedingung am Brunnen gilt:

$$\begin{aligned} \frac{Q_w}{2\pi KL} \cdot \frac{1}{s} &= -A_s \cdot \underbrace{\lim_{r \rightarrow 0+} T_s r \cdot I_1(T_s r)}_{=0} + B_s \cdot \underbrace{\lim_{r \rightarrow 0+} T_s r \cdot K_1(T_s r)}_{=1} \\ &= B_s \end{aligned} \quad (2.3.71)$$

Und für die zweite Randbedingung folgt:

$$\begin{aligned} 0 &= A_s \cdot I_0(T_s r_{\infty}) + B_s \cdot K_0(T_s r_{\infty}) \\ \Rightarrow A_s &= -B_s \cdot \frac{K_0(T_s r_{\infty})}{I_0(T_s r_{\infty})} \end{aligned} \quad (2.3.72)$$

Somit ist die Lösung in diesem Falle gegeben mit:

$$g_s(r) = \frac{Q_w}{2\pi KL} \cdot \frac{1}{s} \cdot \left(K_0(T_s r) - \frac{K_0(T_s r_{\infty})}{I_0(T_s r_{\infty})} \cdot I_0(T_s r) \right) \quad (2.3.73)$$

3. $r_{\text{well}} > 0$ und $r_{\infty} < \infty$:

Für die Koeffizienten A_s und B_s erhält man folgendes lineares Gleichungssystem:

$$\begin{aligned} 0 &= A_s \cdot I_0(T_s r_{\infty}) + B_s \cdot K_0(T_s r_{\infty}) \\ \frac{Q_w \cdot \frac{1}{s}}{2\pi KL \cdot T_s r_{\text{well}}} &= -A_s \cdot I_1'(T_s r_{\text{well}}) + B_s \cdot K_1'(T_s r_{\text{well}}) \end{aligned} \quad (2.3.74)$$

Damit folgt mit der Cramerschen Regel:

$$A_s = \frac{Q_w \cdot \frac{1}{s}}{2\pi KL \cdot T_s r_{\text{well}}} \cdot \frac{-K_0(T_s r_{\infty})}{I_1(T_s r_{\text{well}}) K_0(T_s r_{\infty}) + K_1(T_s r_{\text{well}}) I_0(T_s r_{\infty})} \quad (2.3.75)$$

$$B_s = \frac{Q_w \cdot \frac{1}{s}}{2\pi KL \cdot T_s r_{\text{well}}} \cdot \frac{I_0(T_s r_{\infty})}{I_1(T_s r_{\text{well}}) K_0(T_s r_{\infty}) + K_1(T_s r_{\text{well}}) I_0(T_s r_{\infty})} \quad (2.3.76)$$

Somit ist die Lösung in diesem Falle gegeben mit:

$$g_s(r) = \frac{Q_w \cdot \frac{1}{s}}{2\pi KL \cdot T_s r_{\text{well}}} \cdot \frac{I_0(T_s r_{\infty}) K_0(T_s r) - K_0(T_s r_{\infty}) I_0(T_s r)}{I_1(T_s r_{\text{well}}) K_0(T_s r_{\infty}) + K_1(T_s r_{\text{well}}) I_0(T_s r_{\infty})} \quad (2.3.77)$$

Da diese Funktionen nicht bzw. schwierig symbolisch zurücktransformiert werden können, kann man sie zur numerischen Auswertung mit dem Stehfest-Algorithmus zurücktransformieren. Für den ersten Fall wird im Buch von Häfner [Häfner et al., 1992, S. 598f] folgende symbolische Funktion im Zeit-Raum angegeben:

$$\mathcal{S}(a, b) := \frac{4}{\pi} \int_0^{\infty} \frac{(J_1(\tau) Y_0(b\tau) - Y_1(\tau) J_0(b\tau))}{\tau^2 (J_1^2(\tau) + Y_1^2(\tau))} (1 - e^{-a\tau^2}) d\tau \quad (2.3.78)$$

Die Funktionen J_0 , J_1 , Y_0 und Y_1 sind dabei weitere Lösungen der Besselschen Differentialgleichung (vgl. [Abramowitz et al., 1972, S. 358ff]). Damit lautet die Laplacerücktransformierte von (2.3.70):

$$\mathcal{L}^{-1} \left\{ \frac{Q_w}{2\pi KL} \cdot \frac{K_0(T_s r)}{s \cdot T_s r_{\text{well}} \cdot K_1(T_s r_{\text{well}})} \right\} (t) = \frac{Q_w}{4\pi KL} \cdot \mathcal{S} \left(\frac{K \cdot t}{S \cdot r_{\text{well}}^2}, \frac{r}{r_{\text{well}}} \right) \quad (2.3.79)$$

Zur numerischen Auswertung ist diese Funktion allerdings weniger geeignet.

2.4. Potenzreihen

Zur Anwendung der Frobenius-Methode müssen die entstehenden Koeffizientenfunktionen als Potenzreihen vorliegen. Um diese zu entwickeln, trage ich einige Eigenschaften von Potenzreihen zusammen.

2.4.1. Wichtige Potenzreihen

Folgende Potenzreihen werden im folgenden häufig verwendet. Zum einen die *Exponentialreihe*:

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} \quad (2.4.1)$$

Mit einem Konvergenzradius von ∞ , sowie die Binomische Reihe als Verallgemeinerung des binomischen Satzes. Diese ist für $\lambda \in \mathbb{C}$ gegeben mit:

$$(1+x)^\lambda = \sum_{k=0}^{\infty} \binom{\lambda}{k} x^k \quad (2.4.2)$$

Dabei ist $\binom{\lambda}{k}$ der verallgemeinerte Binomialkoeffizient:

$$\binom{\lambda}{k} = \prod_{j=1}^k \frac{(\lambda - j + 1)}{j} \quad (2.4.3)$$

Der Konvergenzradius ist dabei 1 falls $\lambda \notin \mathbb{N}$. Falls $\lambda \in \mathbb{N}$, so wird (2.4.2) zu einem Polynom, da dann (2.4.3) für $k > \lambda$ konstant 0 ist.

2.4.2. Die Cauchyschen Ungleichungen

Sei $a \in \mathbb{C}$ und $\rho > 0$. Sei weiter f eine auf $U_\rho(a)$ holomorphe Funktion mit:

$$\forall_{z \in U_\rho(a)} |f(z)| \leq M \quad (2.4.4)$$

für ein $M \in \mathbb{R}$. Dann gilt:

$$\forall_{k \in \mathbb{N}} \left| f^{(k)}(a) \right| \leq \frac{k!M}{\rho^k} \quad (2.4.5)$$

Ein Beweis dafür findet sich zum Beispiel in [Rudin, 1999, S. 255].

Bestimmung des Konvergenzradius:

Mit den Cauchyschen Ungleichungen kann man sehr einfach den Konvergenzradius der Potenzreihe einer holomorphen Funktion bestimmen, wenn man das Holomorphie-Gebiet dieser Funktion genau kennt. Sei dazu $f : \mathbb{C} \rightarrow \mathbb{C}$ eine Funktion und $\Omega_f \subset \mathbb{C}$ deren Holomorphie-Gebiet. Sei $z_0 \in \Omega_f$ beliebig und $\rho > 0$ so, dass $\overline{U}_\rho(z_0) \subset \Omega_f$ ist. Da f in Ω_f holomorph ist, ist sie um z_0 in eine Potenzreihe entwickelbar:

$$f(z) = \sum_{k=0}^{\infty} \frac{f^{(k)}(z_0)}{k!} \cdot (z - z_0)^k \quad (2.4.6)$$

Da f stetig ist und $\overline{U}_\rho(z_0)$ kompakt, existiert das Maximum:

$$M := \max_{z \in \overline{U}_\rho(z_0)} |f(z)| \quad (2.4.7)$$

Es gilt nach den Cauchyschen Ungleichungen Folgendes:

$$\forall_{k \in \mathbb{N}} \frac{|f^{(k)}(z_0)|}{k!} \cdot \rho^k \leq M \quad (2.4.8)$$

Somit konvergiert die Reihe (2.4.6) für $|z - z_0| < \rho$, da für $|z - z_0| = \rho \cdot \varepsilon$ mit $\varepsilon \in [0, 1)$ gilt:

$$\begin{aligned} \sum_{k=0}^{\infty} \left| \frac{f^{(k)}(z_0)}{k!} \cdot (z - z_0)^k \right| &= \sum_{k=0}^{\infty} \left| \frac{f^{(k)}(z_0)}{k!} \right| \rho^k \varepsilon^k \\ &\leq M \cdot \frac{1}{1 - \varepsilon} < \infty \end{aligned} \quad (2.4.9)$$

Nun kann man die Umgebung $U_\rho(z_0)$ noch größer wählen und erhält dasselbe Resultat. Daraus ergibt sich durch Grenzübergang für den Konvergenzradius ρ von f um z_0 :

$$\rho = \sup \{ r > 0 \mid U_r(z_0) \subset \Omega_f \} \quad (2.4.10)$$

Beispiel: Man betrachte die binomische Reihe (2.4.2) mit $\lambda \notin \mathbb{N}$. Da z^λ für negative z nicht definiert ist, ist der Definitionsbereich für

$$(1 + z)^\lambda = \exp(\lambda \ln(1 + z))$$

gegeben durch $\mathbb{C} \setminus (-\infty, -1)$. Entwickelt man die Funktion um $z_0 = 0$, so erhält man nach obiger Betrachtung den Konvergenzradius 1, als Abstand von 0 zu $(-\infty, -1)$.

Man erspart sich also das Abschätzen der Reihenkoeffizienten, was für kompliziertere Funktionen die Bestimmung des Konvergenzradius erst möglich macht.

2.4.3. Die Regel von Faà di Bruno

Die Regel von Faà di Bruno, benannt nach dem italienischen Mathematiker Francesco Faà di Bruno, ist in der Differentialrechnung eine Verallgemeinerung der Kettenregel auf Ableitungen höherer Ordnung. Um sie zu formulieren, möchte ich vorweg einige Notationen definieren. Sei dazu $\mathbf{k} \in \mathbb{N}_0^n$ ein Multiindex. Dann gelte:

$$\mathbf{k}! := \prod_{j=1}^n k_j! \quad (2.4.11)$$

$$|\mathbf{k}| := \sum_{j=1}^n k_j \quad (2.4.12)$$

$$\binom{|\mathbf{k}|}{\mathbf{k}} := \frac{|\mathbf{k}|!}{\mathbf{k}!} \quad (2.4.13)$$

$$T_n := \left\{ \mathbf{k} \in \mathbb{N}_0^n \mid \sum_{j=1}^n j \cdot k_j = n \right\} \quad (2.4.14)$$

$$P(n) := |T_n| \quad (2.4.15)$$

Dabei nennt man T_n die *Menge der Partitionen von n* und $P(n)$ die *Partitionsfunktion*. Des Weiteren ist $\binom{|\mathbf{k}|}{\mathbf{k}}$ der *Multinomialkoeffizient* zu \mathbf{k} mit dessen Hilfe man den Multinomialssatz formulieren kann. Seien dazu $n, m \in \mathbb{N}$ und $x_1, \dots, x_m \in \mathbb{C}$. Dann gilt:

$$\left(\sum_{j=1}^m x_j \right)^n = \sum_{|\mathbf{k}|=n} \binom{|\mathbf{k}|}{\mathbf{k}} \cdot \prod_{j=1}^m x_j^{k_j} \quad (2.4.16)$$

Der Beweis erfolgt über eine vollständige Induktion.

Seien $f, g : \mathbb{C} \rightarrow \mathbb{C}$ Funktionen, wobei g in z_0 und f in $g(z_0)$ holomorph⁵ sind. Sei $n \in \mathbb{N}$ beliebig. Dann gilt:

$$(f \circ g)^{(n)}(z_0) = \sum_{\mathbf{k} \in T_n} \frac{n!}{\mathbf{k}!} f^{(|\mathbf{k}|)}(g(z_0)) \cdot \prod_{m=1}^n \left(\frac{g^{(m)}(z_0)}{m!} \right)^{k_m} \quad (2.4.17)$$

Beweis: Sei ohne Beschränkung der Allgemeinheit $z_0 = 0$ und $g(z_0) = 0$. Zuerst zeige ich, dass $(f \circ g)^{(n)}(z_0)$ nur von den Ableitung bis zum Grad n von f und g abhängt. Genauer: Es gibt Polynome $P_{n,k}$, sodass:

$$(f \circ g)^{(n)} = \sum_{k=0}^n (f^{(k)} \circ g) \cdot P_{n,k}(g^{(1)}, \dots, g^{(k)}) \quad (2.4.18)$$

Dies folgt einfach via Induktion. Für $n = 0$ gilt: $P_{0,0} = 1$. Daraus folgt:

$$\begin{aligned} (f \circ g)^{(n+1)} &= \left(\sum_{k=0}^n (f^{(k)} \circ g) \cdot P_{n,k}(g^{(1)}, \dots, g^{(k)}) \right)' \\ &= \sum_{k=0}^n (f^{(k+1)} \circ g) \cdot g' \cdot P_{n,k}(g^{(1)}, \dots, g^{(k)}) \\ &\quad + \sum_{j=1}^k \partial_j P_{n,k}(g^{(1)}, \dots, g^{(k)}) \cdot g^{(j+1)} \\ &= \sum_{k=0}^{n+1} (f^{(k)} \circ g) \cdot P_{n+1,k}(g^{(1)}, \dots, g^{(k)}) \end{aligned} \quad (2.4.19)$$

Somit treten nur $f^{(1)}, \dots, f^{(n)}$ bzw. $g^{(1)}, \dots, g^{(n)}$ auf. Also kann ich mir folgende Hilfsfunktionen definieren:

$$\tilde{f}(z) = \sum_{k=0}^n \underbrace{\frac{f^{(k)}(0)}{k!}}_{=: f_k} z^k \quad (2.4.20)$$

$$\tilde{g}(z) = \sum_{k=1}^n \underbrace{\frac{g^{(k)}(0)}{k!}}_{=: g_k} z^k \quad (2.4.21)$$

Dann folgt wegen obiger Betrachtung:

$$(\tilde{f} \circ \tilde{g})^{(n)}(0) = (f \circ g)^{(n)}(0) \quad (2.4.22)$$

Nun kann ich aber diese Polynome einfach ineinander einsetzen und erhalte mit dem Multinomialatz:

$$\begin{aligned} (\tilde{f} \circ \tilde{g})(z) &= \sum_{m=0}^n f_m \cdot \left(\sum_{j=1}^n g_j \cdot z^j \right)^m \\ &= \sum_{m=0}^n f_m \cdot \sum_{|\mathbf{k}|=m} \binom{|\mathbf{k}|}{\mathbf{k}} \cdot \prod_{j=1}^n (g_j \cdot z^j)^{k_j} \\ &= \sum_{m=0}^n f_m \cdot \sum_{|\mathbf{k}|=m} \binom{|\mathbf{k}|}{\mathbf{k}} \cdot \left(\prod_{j=1}^n g_j^{k_j} \right) \cdot z^{1 \cdot k_1 + \dots + n \cdot k_n} \end{aligned} \quad (2.4.23)$$

⁵Für reelle Funktionen würde reichen, dass g und f in z_0 respektive $g(z_0)$ n -mal differenzierbar sind.

Also ist $(\tilde{f} \circ \tilde{g})(z)$ ein Polynom:

$$(\tilde{f} \circ \tilde{g})(z) = \sum_{k=0}^N p_k z^k \quad (2.4.24)$$

Und für die n -te Ableitung in 0 gilt damit:

$$(\tilde{f} \circ \tilde{g})^{(n)}(0) = n! \cdot p_n \quad (2.4.25)$$

Es sind also in Gleichung (2.4.23) die Koeffizienten vor allen z^n zusammen addieren. Durch Umsortieren erhält man die Bedingung

$$1 \cdot k_1 + \dots + n \cdot k_n = n \quad (2.4.26)$$

Somit ist über alle Partition von n zu summieren. Man erhält, unter Beachtung der Beziehung $f_m \cdot m! = f^{(m)}(g(0))$ und $g_j = \frac{g^{(j)}(0)}{j!}$:

$$(f \circ g)^{(n)}(0) = \sum_{\mathbf{k} \in T_n} \frac{n!}{\mathbf{k}!} \cdot (f^{(|\mathbf{k}|)} \circ g)(0) \cdot \prod_{m=1}^n \left(\frac{g^{(m)}(0)}{m!} \right)^{k_m} \quad (2.4.27)$$

Damit ist alles gezeigt. □

Diese Regel werde ich benutzen, um die Potenzreihe von verketteten analytischen Funktionen aufzustellen. Um dies später zu implementieren, ordne ich die Partitionen von n lexikografisch:

$$\begin{aligned} T_n &= \{ \mathbf{k}_{n,1}, \dots, \mathbf{k}_{n,P(n)} \} \\ \mathbf{k}_{n,1} &= (0, \dots, 0, 1) \\ \mathbf{k}_{n,P(n)} &= (n, 0, \dots, 0) \end{aligned} \quad (2.4.28)$$

2.4.4. Operationen mit Potenzreihen

2.4.4.1. Vektorraumoperationen Seien $f, g : \mathbb{C} \rightarrow \mathbb{C}$ Funktionen, welche in $z_0 \in \mathbb{C}$ holomorph sind, mit folgenden Potenzreihen:

$$f(z) = \sum_{k=0}^{\infty} f_k (z - z_0)^k \quad g(z) = \sum_{k=0}^{\infty} g_k (z - z_0)^k \quad (2.4.29)$$

Die Konvergenzradien seien R_f und R_g . Sei $\lambda \in \mathbb{C}$ beliebig. Dann gilt:

$$f(z) + \lambda g(z) = \sum_{k=0}^{\infty} (f_k + \lambda g_k) (z - z_0)^k \quad (2.4.30)$$

Der Konvergenzradius erfüllt dabei $R \geq \min \{R_f, R_g\}$. Somit bilden die um z_0 entwickelten Potenzreihen einen Vektorraum.

Beweis: Da die Potenzreihen im gemeinsamen Konvergenzbereich gleichmäßig konvergieren, folgt die Aussage sofort. □

2.4.4.2. Multiplikation Seien $f, g : \mathbb{C} \rightarrow \mathbb{C}$ Funktionen, welche in $z_0 \in \mathbb{C}$ holomorph sind, mit folgenden Potenzreihen:

$$f(z) = \sum_{k=0}^{\infty} f_k (z - z_0)^k \quad g(z) = \sum_{k=0}^{\infty} g_k (z - z_0)^k \quad (2.4.31)$$

Die Konvergenzradien seien R_f und R_g . Dann ist $f(z) \cdot g(z)$ ebenfalls holomorph in z_0 und es gilt:

$$(f \cdot g)(z) = \sum_{k=0}^{\infty} \left(\sum_{j=0}^k f_j g_{k-j} \right) (z - z_0)^k \quad (2.4.32)$$

Der Konvergenzradius erfüllt dabei $R \geq \min \{R_f, R_g\}$.

Beweis: Die Funktion $f \cdot g$ ist offensichtlich holomorph in z_0 , da sie nach der Produktregel differenzierbar ist:

$$(f \cdot g)' = f' \cdot g + f \cdot g'$$

Somit ist sie um z_0 in eine Potenzreihe entwickelbar. Nach dem Satz von Taylor gilt:

$$(f \cdot g)(z) = \sum_{k=0}^{\infty} \frac{(f \cdot g)^{(k)}(z_0)}{k!} (z - z_0)^k$$

Mit der Leibnizregel folgt:

$$\begin{aligned} \frac{(f \cdot g)^{(k)}(z_0)}{k!} &= \frac{1}{k!} \cdot \sum_{j=0}^k \binom{k}{j} f^{(j)}(z_0) g^{(k-j)}(z_0) \\ &= \frac{1}{k!} \cdot \sum_{j=0}^k k! \cdot \frac{f^{(j)}(z_0)}{j!} \frac{g^{(k-j)}(z_0)}{(k-j)!} \\ &= \sum_{j=0}^k f_j g_{k-j} \end{aligned}$$

Nach den Betrachtungen in Abschnitt 2.4.2 folgt, dass der Konvergenzradius der entstehenden Potenzreihe größer ist als $\min \{R_f, R_g\}$. \square

2.4.4.3. Ableitung Sei $f : \mathbb{C} \rightarrow \mathbb{C}$ eine Funktion, welche in $z_0 \in \mathbb{C}$ holomorph ist, mit folgender Potenzreihe:

$$f(z) = \sum_{k=0}^{\infty} f_k (z - z_0)^k \quad (2.4.33)$$

Der Konvergenzradius sei R_f . Dann ist $f'(z)$ ebenfalls holomorph in z_0 und es gilt:

$$f'(z) = \sum_{k=0}^{\infty} f_{k+1} \cdot (k+1) (z - z_0)^k \quad (2.4.34)$$

Der Konvergenzradius ist ebenfalls R_f .

Beweis: Die Funktion f ist offensichtlich differenzierbar und da die Reihe gleichmäßig konvergiert, kann man den Ableitungsoperator in die Summe ziehen. Der Konvergenzradius ist nach dem Quotientenkriterium identisch. \square

2.4.4.4. Verkettung Seien $f, g : \mathbb{C} \rightarrow \mathbb{C}$ Funktionen, wobei g in $z_0 \in \mathbb{C}$ und f in $g(z_0)$ holomorph ist, mit folgenden Potenzreihen:

$$f(z) = \sum_{k=0}^{\infty} f_k (z - g(z_0))^k \quad g(z) = \sum_{k=0}^{\infty} g_k (z - z_0)^k \quad (2.4.35)$$

Bezeichne Ω_f das Holomorphie-Gebiet von f und Ω_g das Holomorphie-Gebiet von g . Dann ist $f \circ g$ holomorph in z_0 und es gilt:

$$\begin{aligned} (f \circ g)(z) &= \sum_{k=0}^{\infty} h_k (z - z_0)^k \\ h_0 &= f_0 \\ \forall_{n \geq 1} h_n &= \sum_{\mathbf{k} \in T_n} \binom{|\mathbf{k}|}{\mathbf{k}} \cdot f_{|\mathbf{k}|} \cdot \prod_{m=1}^n g_m^{k_m} \end{aligned} \quad (2.4.36)$$

Der Konvergenzradius erfüllt dabei:

$$R \geq \sup \{r > 0 \mid U_r(z_0) \subset \Omega_g, g(U_r(z_0)) \subset \Omega_f\} \quad (2.4.37)$$

Beweis: Die Funktion $f \circ g$ ist offensichtlich holomorph in z_0 , da sie nach der Kettenregel differenzierbar ist:

$$(f \circ g)'(z_0) = f'(g(z_0)) \cdot g'(z_0) \quad (2.4.38)$$

Somit ist sie um z_0 in eine Potenzreihe entwickelbar. Nach dem Satz von Taylor gilt:

$$(f \circ g)(z) = \sum_{k=0}^{\infty} \frac{(f \circ g)^{(k)}(z_0)}{k!} (z - z_0)^k \quad (2.4.39)$$

Mit der Regel von Faà di Bruno folgt nun für $n \geq 1$:

$$\begin{aligned} \frac{(f \circ g)^{(n)}(z_0)}{n!} &= \sum_{\mathbf{k} \in T_n} \frac{1}{\mathbf{k}!} f^{(|\mathbf{k}|)}(g(z_0)) \cdot \prod_{m=1}^n \left(\frac{g^{(m)}(z_0)}{m!} \right)^{k_m} \\ &= \sum_{\mathbf{k} \in T_n} \frac{|\mathbf{k}|!}{\mathbf{k}!} f_{|\mathbf{k}|} \cdot \prod_{m=1}^n g_m^{k_m} \end{aligned}$$

Durch zweifache Anwendung des Resultats aus Abschnitt 2.4.2 folgt die Abschätzung für den Konvergenzradius. Dass $R > 0$ gilt, folgt aus der Stetigkeit von f und g . \square

Um später die Potenzreihen berechnen zu können, werde ich die Summe in (2.4.36) über die lexikografisch geordneten Partitionen laufen lassen:

$$\forall_{n \geq 1} h_n = \sum_{j=1}^{P(n)} \binom{|\mathbf{k}_{n,j}|}{\mathbf{k}_{n,j}} \cdot f_{|\mathbf{k}_{n,j}|} \cdot \prod_{m=1}^n g_m^{k_{n,j,m}} \quad (2.4.40)$$

$$\forall_{n \geq 1} \forall_{1 \leq j \leq P(n)} \mathbf{k}_{n,j} = (k_{n,j,1}, \dots, k_{n,j,n}) \quad (2.4.41)$$

Siehe dazu Kapitel 3.6.1. Die ersten geordneten Partitionen für $n = 1, \dots, 4$ sehen wie folgt aus:

$$T_1 = \{(1)\} \quad (2.4.42)$$

$$T_2 = \{(0, 1), (2, 0)\}$$

$$T_3 = \{(0, 0, 1), (1, 1, 0), (3, 0, 0)\}$$

$$T_4 = \{(0, 0, 0, 1), (1, 0, 1, 0), (0, 2, 0, 0), (2, 1, 0, 0), (4, 0, 0, 0)\}$$

Damit erhält man eine eindeutige Berechnungsvorschrift für die h_n . Diese Summe wird mit wachsendem n schnell groß, wie zum Beispiel $P(40) = 37338$ zeigt. Nach [Matoušek, 2007, S. 396f] kann man für $P(n)$ folgende Schranken geben:

$$P(n) \leq \exp\left(\pi \cdot \sqrt{\frac{2}{3}n}\right) \quad (2.4.43)$$

$$P(n) \geq \frac{e^{-5}}{n^2} \cdot \exp(2\sqrt{n}) \quad (2.4.44)$$

$$P(n) \sim \frac{\exp\left(\pi \cdot \sqrt{\frac{2}{3}n}\right)}{4\sqrt{3}n} =: E(n) \quad (2.4.45)$$

Als Beispiel ergibt sich für $n = 40$:

$$\begin{aligned} P(40) &\leq 11107317 \\ P(40) &\geq 2 \\ P(40) &\sim 40080 \end{aligned} \tag{2.4.46}$$

Somit sind die Abschätzungen durch die gegebenen Schranken für relativ kleine n sehr grob. Die approximative Abschätzung durch $E(n)$ hingegen ist recht zuverlässig. Zur näherungsweise Berechnung von Funktionen durch ihre Potenzreihe muss man diese ohnehin ab einem gewissen Summanden abbrechen. Ich nutze als Abschneidewert:

$$n_{\max} = 40 \tag{2.4.47}$$

Da alle Fälle von verketteten Funktionen die hier auftreten $f(z) = e^z$ erfüllen, betrachte ich diesen Fall genauer. Sei g holomorph in z_0 . Es gilt:

$$\forall_{n \geq 0} f^{(n)}(g(z_0)) = e^{g(z_0)} \tag{2.4.48}$$

Somit folgt aus Formel (2.4.36):

$$\begin{aligned} \forall_{n \geq 1} h_n &= e^{g(z_0)} \cdot \sum_{\mathbf{k} \in T_n} \prod_{m=1}^n \frac{g_m^{k_m}}{k_m!} \\ &= e^{g(z_0)} \cdot \sum_{j=1}^{P(n)} \prod_{m=1}^n \frac{g_m^{k_{n,j,m}}}{k_{n,j,m}!} \end{aligned} \tag{2.4.49}$$

Da die Exponentialfunktion eine ganze Funktion ist, gilt für den Konvergenzradius:

$$R \geq R_g \tag{2.4.50}$$

2.5. Rosenbrock-Verfahren

Da ich später darauf angewiesen bin, die entstehenden Basisfunktionen numerisch fortzusetzen, stelle ich hier ein Verfahren zur numerischen Integration von Anfangswertproblemen in Form von gewöhnlichen Differentialgleichungssystemen vor. Ich folge dabei dem Kapitel 16.6. der Numerical-Recipes über steife Differentialgleichungssysteme in [Press et al., 1992, S. 729ff].

Das *Rosenbrock-Verfahren* ist eine Verallgemeinerung der Runge-Kutta-Methode, welches durch seine semi-implizite Formulierung gut für *steife* System geeignet ist. Auf eine exakte Definition von *Steifheit* und deren Analyse möchte ich hier allerdings verzichten und verweise daher nur auf das Buch von [Strehmel, 1995].

Man betrachtet ein Differentialgleichungssystem der Form:

$$\underline{y}' = f(\underline{y}) \quad (2.5.1)$$

Falls die rechte Seite noch von (2.5.1) explizit von der Variablen x abhängt, also $\underline{y}' = f(\underline{y}, x)$, kann man dies durch die Substitution

$$\begin{pmatrix} \underline{y} \\ x \end{pmatrix}' = \begin{pmatrix} f(\underline{y}, x) \\ 1 \end{pmatrix} \quad (2.5.2)$$

auf obige Form bringen.

Die allgemeine Vorgehensweise von semi-impliziten Verfahren lässt sich am einfachsten durch die *semi-implizite Euler Methode* beschreiben. Dabei erhält man aus Gleichung (2.5.1) durch implizites Differenzieren mit einer Schrittlänge von h die implizite Vorschrift:

$$\underline{y}_{n+1} = \underline{y}_n + h \cdot f(\underline{y}_{n+1}) \quad (2.5.3)$$

Linearisiert man diese Gleichung durch die erste Näherung von f , so ergibt sich:

$$\underline{y}_{n+1} = \underline{y}_n + h \cdot \left(f(\underline{y}_n) + \frac{\partial f}{\partial \underline{y}} \Big|_{\underline{y}_n} \cdot (\underline{y}_{n+1} - \underline{y}_n) \right) \quad (2.5.4)$$

Und damit erhält man nun für \underline{y}_{n+1} folgende Formel:

$$\underline{y}_{n+1} = \underline{y}_n + h \cdot \left(E - h \cdot \frac{\partial f}{\partial \underline{y}} \Big|_{\underline{y}_n} \right)^{-1} \cdot f(\underline{y}_n) \quad (2.5.5)$$

Dadurch muss in jedem Iterationsschritt die Inverse folgender Matrix berechnet werden:

$$E - h \cdot \frac{\partial f}{\partial \underline{y}} \quad (2.5.6)$$

Das Rosenbrock-Verfahren ist nun solch ein semi-implizites Verfahren höherer Ordnung. Man sucht hier eine Lösung der Form:

$$\underline{y}(x_0 + h) = \underline{y}_0 + \sum_{i=1}^s c_i \cdot \underline{k}_i \quad (2.5.7)$$

wobei die Korrekturterme \underline{k}_i ähnlich wie in Gleichung (2.5.4) durch folgende lineare Gleichungssysteme bestimmt sind:

$$\forall_{i=1 \dots s} (E - \gamma_i h \cdot f') \cdot \underline{k}_i = h \cdot f \left(\underline{y}_0 + \sum_{j=1}^{i-1} \alpha_{ij} \underline{k}_j \right) + h \cdot f' \cdot \sum_{j=1}^{i-1} \gamma_{ij} \underline{k}_j \quad (2.5.8)$$

Die Koeffizienten γ , c_i , α_{ij} und γ_{ij} sind dabei fix gewählt. Es gibt verschieden Parametersätze für diese Koeffizienten mit unterschiedlichen Stabilitätseigenschaften. Die Wahl der Parameter hier ist an die Standard-Parameter aus [Press et al., 1992] angelehnt.

Des Weiteren wurde eine adaptive Schrittweitensteuerung implementiert. Das heißt, dass in jedem Schritt zwei verschiedene Näherung \underline{y} und \hat{y} der exakten Lösung $\underline{y}_{\text{ex}}$ mit unterschiedlichen Parametersätzen berechnet werden, die folgendes erfüllen:

$$\underline{y}_{\text{ex}} = \underline{y} + \mathcal{O}(h^5) \quad (2.5.9)$$

$$\underline{y}_{\text{ex}} = \hat{y} + \mathcal{O}(h^4) \quad (2.5.10)$$

Aus deren Differenz:

$$|\underline{y} - \hat{y}| = \mathcal{O}(h^4) \quad (2.5.11)$$

wird dann abgeschätzt, ob man im aktuellen Schritt die Weite h vergrößern kann oder verkleinern muss. Ähnliche Methoden gibt es auch für das klassische Runge-Kutta-Verfahren.

3. Umsetzung

3.1. Anwendung der Frobenius-Methode

Mit der oben geleisteten Vorarbeit kann ich nun zur Bearbeitung der gegebenen Differentialgleichung übergehen. Ich formuliere als Erstes die Differentialgleichungen der beiden Modelle im Laplaceraum und gehe dann über zur Anwendung der Frobenius-Methode. Da die beiden Fälle sehr ähnlich sind, kann dies gleichzeitig passieren. Einzig die Potenzreihen der Koeffizientenfunktionen sind separat zu entwickeln. Ich betrachte also folgende beide Differentialgleichungen:

1. Im 2D-Modell:

$$\begin{aligned} \frac{1}{r} \frac{\partial}{\partial r} \left(r \cdot \exp \left(\frac{-\frac{1}{2}\sigma^2}{1 + \left(\frac{\zeta \cdot r}{\ell}\right)^2} \right) \cdot \frac{\partial h}{\partial r}(r, t) \right) &= \frac{S}{T_G} \frac{\partial h}{\partial t}(r, t) & (3.1.1) \\ \forall t > 0 \quad \lim_{r \rightarrow r_{\text{well}}} r \cdot \frac{\partial h}{\partial r}(r, t) &= -\frac{Q_w}{2\pi T_H^{\text{CG}}(r_{\text{well}})} \\ \forall t > 0 \quad \lim_{r \rightarrow r_{\infty}} h(r, t) &= 0 \\ h(r, t = 0) &= 0 \end{aligned}$$

Die Differentialgleichung schreibe ich wie folgt um:

$$\frac{\partial^2 h}{\partial r^2} + \frac{1}{r} \left(1 + \frac{\sigma^2 \left(\frac{\zeta \cdot r}{\ell}\right)^2}{\left(1 + \left(\frac{\zeta \cdot r}{\ell}\right)^2\right)^2} \right) \frac{\partial h}{\partial r} - \frac{S}{T_G} \exp \left(\frac{\frac{1}{2}\sigma^2}{1 + \left(\frac{\zeta \cdot r}{\ell}\right)^2} \right) \frac{\partial h}{\partial t} = 0 \quad (3.1.2)$$

Auf Gleichung (3.1.2) wende ich nun die Laplacetransformation bezüglich t an und erhalte mit dem Differentiationssatz (2.1.14) und den Bezeichnungen

$$\begin{aligned} \mathcal{L}_t \{h(r, [\cdot])\}(s) &=: g_s(r) & (3.1.3) \\ \frac{\zeta}{\ell} &=: C \end{aligned}$$

Folgendes:

$$r^2 g_s'' + r \left(1 + \frac{\sigma^2 (Cr)^2}{(1 + (Cr)^2)^2} \right) g_s' - r^2 \frac{s \cdot S}{T_G} \exp \left(\frac{\frac{1}{2}\sigma^2}{1 + (Cr)^2} \right) g_s = 0 \quad (3.1.4)$$

Ich führe folgende Vereinfachungen ein:

$$\begin{aligned} Cr &=: \tau & (3.1.5) \\ g_s(r) &=: \tilde{g}_s(\tau) \end{aligned}$$

und erhalte damit für die Ableitungen:

$$r g_s'(r) = \tau \tilde{g}_s'(\tau) \quad r^2 g_s''(r) = \tau^2 \tilde{g}_s''(\tau) \quad (3.1.6)$$

Insgesamt erhält man folgende gewöhnliche Differentialgleichung:

$$\tau^2 \tilde{g}_s'' + \tau \left(1 + \frac{\sigma^2 \tau^2}{(1 + \tau^2)^2} \right) \tilde{g}_s' - \frac{\tau^2 s \cdot S}{C^2 T_G} \exp \left(\frac{\frac{1}{2}\sigma^2}{1 + \tau^2} \right) \tilde{g}_s = 0 \quad (3.1.7)$$

Mit den Randwerten:

$$\forall s > 0 \quad \lim_{\tau \rightarrow Cr_{\text{well}}} \tau \cdot \tilde{g}_s'(\tau) = -\frac{Q_w}{2\pi T_H^{\text{CG}}(r_{\text{well}})} \cdot \frac{1}{s} \quad (3.1.8)$$

$$\forall s > 0 \quad \lim_{\tau \rightarrow Cr_{\infty}} \tilde{g}_s(\tau) = 0 \quad (3.1.9)$$

2. Im 3D-Modell:

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \cdot \exp \left(\frac{\chi}{\sqrt{1 + \left(\frac{\zeta \cdot r}{\sqrt[3]{e} \cdot \ell} \right)^2}} \right) \cdot \frac{\partial h}{\partial r} (r, t) \right) = \frac{S}{K_{\text{efu}}} \frac{\partial h}{\partial t} (r, t) \quad (3.1.10)$$

$$\forall t > 0 \quad \lim_{r \rightarrow r_{\text{well}}} r \cdot \frac{\partial h}{\partial r} (r, t) = - \frac{Q_w}{2\pi L K^{\text{CG}} (r_{\text{well}})}$$

$$\forall t > 0 \quad \lim_{r \rightarrow r_{\infty}} h (r, t) = 0$$

$$h (r, t = 0) = 0$$

Die Differentialgleichung schreibe ich wie folgt um:

$$\frac{\partial^2 h}{\partial r^2} + \frac{1}{r} \left(1 - \frac{3\chi \cdot \left(\frac{\zeta \cdot r}{\sqrt[3]{e} \cdot \ell} \right)^2}{\sqrt{1 + \left(\frac{\zeta \cdot r}{\sqrt[3]{e} \cdot \ell} \right)^2}} \right) \frac{\partial h}{\partial r} - \frac{S}{K_{\text{efu}}} \exp \left(\frac{-\chi}{\sqrt{1 + \left(\frac{\zeta \cdot r}{\sqrt[3]{e} \cdot \ell} \right)^2}} \right) \frac{\partial h}{\partial t} = 0 \quad (3.1.11)$$

Auf Gleichung 3.1.11 wende ich die Laplacetransformation bezüglich t an und erhalte mit dem Differentiationsatz (2.1.14) und den Bezeichnungen

$$\mathcal{L}_t \{h (r, [\cdot])\} (s) =: g_s (r) \quad (3.1.12)$$

$$\frac{\zeta}{\sqrt[3]{e} \cdot \ell} =: C$$

Folgendes:

$$r^2 g_s'' + r \left(1 - \frac{3\chi \cdot (Cr)^2}{\sqrt{1 + (Cr)^2}} \right) g_s' - r^2 \frac{s \cdot S}{K_{\text{efu}}} \exp \left(\frac{-\chi}{\sqrt{1 + (Cr)^2}} \right) g_s = 0 \quad (3.1.13)$$

Unter Verwendung der Substitution (3.1.5) und der Ableitungen (3.1.6) erhält man für das 3D-Modell folgende gewöhnliche Differentialgleichung:

$$\tau^2 \tilde{g}_s'' + \tau \left(1 - \frac{3\chi \cdot \tau^2}{\sqrt{1 + \tau^2}} \right) \tilde{g}_s' - \frac{\tau^2 s \cdot S}{C^2 K_{\text{efu}}} \exp \left(\frac{-\chi}{\sqrt{1 + \tau^2}} \right) \tilde{g}_s = 0 \quad (3.1.14)$$

Mit den Randwerten:

$$\forall s > 0 \quad \lim_{\tau \rightarrow Cr_{\text{well}}} \tau \cdot \tilde{g}_s' (\tau) = - \frac{Q_w}{2\pi L K^{\text{CG}} (r_{\text{well}})} \cdot \frac{1}{s} \quad (3.1.15)$$

$$\forall s > 0 \quad \lim_{\tau \rightarrow Cr_{\infty}} \tilde{g}_s (\tau) = 0 \quad (3.1.16)$$

3.1.1. Gemeinsames Vorgehen im 2D- und 3D-Modell

Mit der obigen Modellierung des Problems erhält man in beiden Fällen eine Differentialgleichung folgender Form:

$$\tau^2 \tilde{g}_s'' (\tau) + \tau \alpha (\tau) \tilde{g}_s' + s \cdot \beta (\tau) \tilde{g}_s = 0 \quad (3.1.17)$$

$$\forall s > 0 \quad \lim_{\tau \rightarrow Cr_{\text{well}}} \tau \cdot \tilde{g}_s' (\tau) = Q (r_{\text{well}}) \cdot \frac{1}{s}$$

$$\forall s > 0 \quad \lim_{\tau \rightarrow Cr_{\infty}} \tilde{g}_s (\tau) = 0$$

Mit jeweils analytischen Funktionen α und β :

$$\alpha(\tau) = \sum_{k=0}^{\infty} \alpha^{(k)} \cdot \tau^k \quad (3.1.18)$$

$$\beta(\tau) = \sum_{k=0}^{\infty} \beta^{(k)} \cdot \tau^k \quad (3.1.19)$$

Welche in beiden Fällen Folgendes erfüllen:

$$\alpha^{(0)} = 1 \quad (3.1.20)$$

$$s \cdot \beta^{(0)} = 0 \quad (3.1.21)$$

Somit ist auch in beiden Fällen das indiziale Polynom (2.2.11) gegeben mit:

$$I(\nu) = \nu^2 \quad (3.1.22)$$

Es gibt also nur den charakteristischen Exponenten $\nu = 0$ und letztlich nach dem Satz von Fuchs 2.2.2 für beide Modelle ein Fundamentalsystem der Form:

$$u_{1,s}(\tau) = g_{1,s}(\tau) \quad (3.1.23)$$

$$u_{2,s}(\tau) = g_{2,s}(\tau) + \ln(\tau) \cdot g_{1,s}(\tau) \quad (3.1.24)$$

Mit analytischen Funktionen $g_{1,s}$ und $g_{2,s}$:

$$g_{1,s}(\tau) = \sum_{k=0}^{\infty} g_{1,s}^{(k)} \cdot \tau^k \quad (3.1.25)$$

$$g_{2,s}(\tau) = \sum_{k=0}^{\infty} g_{2,s}^{(k)} \cdot \tau^k \quad (3.1.26)$$

Diese werden, in Anlehnung an die Theis-Lösung in Abschnitt 2.3.2 und nach den Berechnungen in Abschnitt 2.2.1, wie folgt entwickelt:

$$g_{1,s}^{(0)} := 1 \quad (3.1.27)$$

$$g_{1,s}^{(k)} := -\frac{1}{k^2} \cdot \sum_{j=0}^{k-1} \left(\alpha^{(k-j)} j + s \cdot \beta^{(k-j)} \right) g_{1,s}^{(j)} \quad (3.1.28)$$

$$g_{2,s}^{(0)} := 0 \quad (3.1.29)$$

$$g_{2,s}^{(k)} = -\frac{1}{k^2} \cdot \left(\sum_{j=0}^{k-1} \left(\alpha^{(k-j)} j + s \cdot \beta^{(k-j)} \right) g_{2,s}^{(j)} + 2k \cdot g_{1,s}^{(k)} + \sum_{j=0}^{k-1} \alpha^{(k-j)} g_{1,s}^{(j)} \right) \quad (3.1.30)$$

Somit braucht man noch die Potenzreihen der Funktionen α und β , um die Basislösungen zu bestimmen.

3.1.2. Entwicklung der Koeffizientenfunktionen

Um die Basisfunktionen für beide Modelle zu berechnen, fehlen die Potenzreihen der Koeffizientenfunktionen. Für diese möchte ich implementierfreundliche Bildungsvorschriften angeben.

3.1.2.1. Im 2D-Modell Hier sind die Koeffizientenfunktionen gegeben mit:

$$\alpha(\tau) = 1 + \frac{\sigma^2 \tau^2}{(1 + \tau^2)^2} \quad (3.1.31)$$

$$s\beta(\tau) = -s \cdot \frac{S}{C^2 T_G} \tau^2 \exp\left(\frac{\frac{1}{2}\sigma^2}{1 + \tau^2}\right) \quad (3.1.32)$$

Ich zerlege diese Funktionen nun in Teilfunktionen und baue sie von innen nach außen auf.

1. Für $\alpha(\tau)$ definiere ich folgende Hilfsfunktion:

$$\alpha_1(\tau) = (1 + \tau^2)^{-2} \quad (3.1.33)$$

$$\Rightarrow \alpha(\tau) = 1 + \sigma^2 \tau^2 \alpha_1(\tau) \quad (3.1.34)$$

Mit der Formel der binomischen Reihe (2.4.2) folgt:

$$\alpha_1(\tau) = \sum_{k=0}^{\infty} \alpha_1^{(k)} \tau^k \quad (3.1.35)$$

$$\alpha_1^{(k)} = \begin{cases} 0 & k = 2n + 1 \quad n \in \mathbb{N}_0 \\ \binom{-2}{n} & k = 2n \quad n \in \mathbb{N}_0 \end{cases}$$

$$\alpha(\tau) = \sum_{k=0}^{\infty} \alpha^{(k)} \tau^k \quad (3.1.36)$$

$$\alpha^{(k)} = \begin{cases} 1 & k = 0 \\ 0 & k = 1 \\ \sigma^2 \alpha_1^{(k-2)} & k \geq 2 \end{cases}$$

Somit sieht α in erster Näherung wie folgt aus:

$$\alpha(\tau) = 1 + \sigma^2 \tau^2 - 2\sigma^2 \tau^4 + 3\sigma^2 \tau^6 + \dots \quad (3.1.37)$$

Der Konvergenzradius ist dabei durch die binomische Reihe mit 1 gegeben.

2. Für $\beta(\tau)$ definiere ich folgende Hilfsfunktionen:

$$\beta_1(\tau) = \frac{1}{2} \sigma^2 (1 + \tau^2)^{-1} \quad (3.1.38)$$

$$\beta_2(\tau) = \exp(\beta_1(\tau)) \quad (3.1.39)$$

$$\Rightarrow \beta(\tau) = -\frac{S}{C^2 T_G} \tau^2 \beta_2(\tau) \quad (3.1.40)$$

Mit der Formel der binomischen Reihe (2.4.2) folgt auch hier:

$$\beta_1(\tau) = \sum_{k=0}^{\infty} \beta_1^{(k)} \tau^k \quad (3.1.41)$$

$$\beta_1^{(k)} = \begin{cases} 0 & k = 2n + 1 \quad n \in \mathbb{N}_0 \\ \frac{1}{2} \sigma^2 \binom{-1}{n} & k = 2n \quad n \in \mathbb{N}_0 \end{cases}$$

Mit der Formel von Faà di Bruno erhält man mit Formel (2.4.36):

$$\beta_2 = \sum_{k=0}^{\infty} \beta_2^{(k)} \tau^k \quad (3.1.42)$$

$$\beta_2^{(0)} = e^{\frac{1}{2}\sigma^2}$$

$$\beta_2^{(n)} = e^{\frac{1}{2}\sigma^2} \cdot \sum_{j=1}^{P(n)} \prod_{m=1}^n \frac{\left(\beta_1^{(m)}\right)^{k_{n,j,m}}}{k_{n,j,m}!} \quad (n \geq 1)$$

Daraus folgt für β insgesamt:

$$\beta(\tau) = \sum_{k=0}^{\infty} \beta^{(k)} \tau^k \quad (3.1.43)$$

$$\beta^{(k)} = \begin{cases} 0 & k = 0, 1 \\ \frac{-S}{C^2 T_G} \beta_2^{(k-2)} & k \geq 2 \end{cases}$$

Somit sieht β in erster Näherung wie folgt aus:

$$\beta(\tau) = -\frac{S}{C^2 T_G} e^{\frac{1}{2}\sigma^2} \cdot \left(\tau^2 - \sigma^2 \tau^4 + \frac{\sigma^2}{2} (\sigma^2 + 3) \tau^6 + \dots \right) \quad (3.1.44)$$

Der Konvergenzradius ist durch die Betrachtungen in Abschnitt 2.4.2 mit 1 gegeben. Der Satz von Fuchs 2.2.2 garantiert für die beiden Fundamentallösungen $u_{1,s}$ und $u_{2,s}$ einen Konvergenzradius von 1 in τ . Nach Rücktransformation auf r erhält man also einen Konvergenzradius von C^{-1} .

3.1.2.2. Im 3D-Modell Hier sind die Koeffizientenfunktionen gegeben mit:

$$\alpha(\tau) = 1 - \frac{3\chi \cdot \tau^2}{\sqrt{1 + \tau^2^5}} \quad (3.1.45)$$

$$s\beta(\tau) = -s \cdot \frac{S}{C^2 K_{\text{efu}}} \tau^2 \exp\left(\frac{-\chi}{\sqrt{1 + \tau^2^3}}\right) \quad (3.1.46)$$

Ich zerlege diese Funktionen nun in Teilfunktionen und baue sie von innen nach außen auf.

1. Für $\alpha(\tau)$ definiere ich folgende Hilfsfunktion:

$$\alpha_1(\tau) = \left(1 + \tau^2\right)^{-\frac{5}{2}} \quad (3.1.47)$$

$$\Rightarrow \alpha(\tau) = 1 - 3\chi\tau^2 \cdot \alpha_1(\tau) \quad (3.1.48)$$

Mit der Formel der binomischen Reihe (2.4.2) folgt:

$$\alpha_1(\tau) = \sum_{k=0}^{\infty} \alpha_1^{(k)} \tau^k \quad (3.1.49)$$

$$\alpha_1^{(k)} = \begin{cases} 0 & k = 2n + 1 \quad n \in \mathbb{N}_0 \\ \binom{-\frac{5}{2}}{n} & k = 2n \quad n \in \mathbb{N}_0 \end{cases}$$

$$\alpha(\tau) = \sum_{k=0}^{\infty} \alpha^{(k)} \tau^k \quad (3.1.50)$$

$$\alpha^{(k)} = \begin{cases} 1 & k = 0 \\ 0 & k = 1 \\ -3\chi\alpha_1^{(k-2)} & k \geq 2 \end{cases}$$

Somit sieht α in erster Näherung wie folgt aus:

$$\alpha(\tau) = 1 - 3\chi\tau^2 + \frac{15}{2}\chi\tau^4 - \frac{105}{8}\chi\tau^6 + \dots \quad (3.1.51)$$

Der Konvergenzradius ist dabei durch die binomische Reihe mit 1 gegeben.

2. Für $\beta(\tau)$ definiere ich folgende Hilfsfunktionen:

$$\beta_1(\tau) = -\chi(1 + \tau^2)^{-\frac{3}{2}} \quad (3.1.52)$$

$$\beta_2(\tau) = \exp(\beta_1(\tau)) \quad (3.1.53)$$

$$\Rightarrow \beta(\tau) = -\frac{S}{C^2 K_{\text{efu}}} \tau^2 \beta_2(\tau) \quad (3.1.54)$$

Mit der Formel der binomischen Reihe (2.4.2) folgt auch hier:

$$\beta_1(\tau) = \sum_{k=0}^{\infty} \beta_1^{(k)} \tau^k \quad (3.1.55)$$

$$\beta_1^{(k)} = \begin{cases} 0 & k = 2n + 1 \quad n \in \mathbb{N}_0 \\ -\chi \binom{-\frac{3}{2}}{n} & k = 2n \quad n \in \mathbb{N}_0 \end{cases}$$

Mit der Formel von Faà di Bruno erhält man mit Formel (2.4.36):

$$\beta_2 = \sum_{k=0}^{\infty} \beta_2^{(k)} \tau^k \quad (3.1.56)$$

$$\beta_2^{(0)} = e^{-\chi}$$

$$\beta_2^{(n)} = e^{-\chi} \cdot \sum_{j=1}^{P(n)} \prod_{m=1}^n \frac{\left(\beta_1^{(m)}\right)^{k_{n,j,m}}}{k_{n,j,m}!} \quad (n \geq 1)$$

Daraus folgt für β insgesamt:

$$\beta(\tau) = \sum_{k=0}^{\infty} \beta^{(k)} \tau^k \quad (3.1.57)$$

$$\beta^{(k)} = \begin{cases} 0 & k = 0, 1 \\ -\frac{S}{C^2 K_{\text{efu}}} \beta_2^{(k-2)} & k \geq 2 \end{cases}$$

Somit sieht β in erster Näherung wie folgt aus:

$$\beta(\tau) = -\frac{S}{C^2 K_{\text{efu}}} e^{-\chi} \cdot \left(\tau^2 + \frac{3\chi}{2} \tau^4 + \frac{3\chi}{8} (3\chi - 5) \tau^6 + \dots \right) \quad (3.1.58)$$

Der Konvergenzradius ist durch die Betrachtungen in Abschnitt 2.4.2 mit 1 gegeben. Der Satz von Fuchs 2.2.2 garantiert für die beiden Fundamentallösungen $u_{1,s}$ und $u_{2,s}$ einen Konvergenzradius von 1 in τ . Nach Rücktransformation auf r erhält man einen Konvergenzradius von C^{-1} .

3.2. Fortsetzung durch numerische Integration

Um die durch Potenzreihen approximierten Basisfunktionen aus Kapitel 3.1 auf größere Intervalle fortzusetzen, formuliere ich die Differentialgleichung (3.1.17) als gewöhnliches Differentialgleichungssystem erster Ordnung. Sei dazu:

$$\begin{aligned} y_1(\tau) &= \tilde{g}_s(\tau) \\ y_2(\tau) &= \tilde{g}'_s(\tau) \end{aligned} \quad (3.2.1)$$

Für $\underline{y} = (y_1, y_2)^T$ erhält man also folgendes Differentialgleichungssystem:

$$\underline{y}'(\tau) = \begin{pmatrix} y_2(\tau) \\ -\frac{\alpha(\tau)}{\tau} y_2(\tau) - s \frac{\beta(\tau)}{\tau^2} y_1(\tau) \end{pmatrix} \quad (3.2.2)$$

Damit ich einen numerischen Integrator für solche Systeme nutzen kann, brauche ich noch die richtigen Anfangswerte. Als Berechnungsgebiet für die numerische Integration wähle ich ein endliches Intervall $I_2 = [r_{\min}, r_{\max}] = [\tau_1, \tau_2]$, welches ich in Kapitel 3.5 noch spezifiziere. Die Anfangswerte ergeben sich durch die obigen Basislösungen:

$$u_{1,s}(\tau_1) = \sum_{k=0}^{\infty} g_{1,s}^{(k)} \cdot \tau_1^k \quad (3.2.3)$$

$$u'_{1,s}(\tau_1) = \sum_{k=0}^{\infty} (k+1) g_{1,s}^{(k+1)} \cdot \tau_1^k$$

$$u_{2,s}(\tau_1) = \sum_{k=0}^{\infty} g_{2,s}^{(k)} \cdot \tau_1^k + \ln(\tau_1) \cdot u_{1,s}(\tau_1) \quad (3.2.4)$$

$$u'_{2,s}(\tau_1) = \sum_{k=0}^{\infty} (k+1) g_{2,s}^{(k+1)} \cdot \tau_1^k + \frac{1}{\tau_1} \cdot u_{1,s}(\tau_1) + \ln(\tau_1) \cdot u'_{1,s}(\tau_1)$$

Da das Differentialgleichungssystem (3.2.2) für $\tau > 0$ keine Singularitäten mehr aufweist, ist die numerische Approximation der Differentialgleichung zulässig. Ich habe mich für das Rosenbrock-Verfahren als Methode für steife Systeme entschieden, da bei meinen Tests ein normales explizites Runge-Kutta-Verfahren numerische Artefakte erzeugt hatte. Dieses Verfahren wurde in Kapitel 2.5 vorgestellt.

3.3. Lösung im Fernfeld

Die aus dem Coarse-Graining-Modell hervorgehenden Leitfähigkeiten K^{CG} respektive Durchlässigkeiten T_{H}^{CG} haben beide die Eigenschaft, dass sie streng monoton sind und gegen einen Fernfeldwert konvergieren:

$$\lim_{r \rightarrow \infty} K^{\text{CG}}(r) = K_{\text{efu}} \quad (3.3.1)$$

$$\lim_{r \rightarrow \infty} T_{\text{H}}^{\text{CG}}(r) = T_{\text{G}} \quad (3.3.2)$$

Somit kann man approximativ annehmen, dass beide ab einem gewissen r_{max} , welches ich in Kapitel 3.5 noch spezifiziere, konstant diesen Wert annehmen. Als Bedingung dafür nehme ich an, dass der relative Fehler zwischen $K^{\text{CG}}(r)$ und K_{efu} bzw. $T_{\text{H}}^{\text{CG}}(r)$ und T_{G} kleiner als ein gewisses $\varepsilon_{\text{error}}$ sein soll. Also:

$$\left| \frac{K^{\text{CG}}(r_{\text{max}}) - K_{\text{efu}}}{K_{\text{efu}}} \right| < \varepsilon_{\text{error}} \quad (3.3.3)$$

$$\left| \frac{T^{\text{CG}}(r_{\text{max}}) - T_{\text{G}}}{T_{\text{G}}} \right| < \varepsilon_{\text{error}} \quad (3.3.4)$$

Durch die Monotonie ist damit für $r > r_{\text{max}}$ die Annahme des konstanten Fernfeldwertes gerechtfertigt. Somit vereinfacht sich die Differentialgleichung im Fernfeld $[r_{\text{max}}, r_{\infty})$ auf die homogene Grundwassergleichung wie in Gleichung (2.3.1):

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \cdot \frac{\partial}{\partial r} h(r, t) \right) = \frac{S}{K} \frac{\partial}{\partial t} h(r, t) \quad (3.3.5)$$

Wobei K entweder K_{efu} oder T_{G} ist. Die Lösung von (3.3.5) ist die Theis-Lösung, wie ausführlich in Kapitel 2.3 diskutiert. Im Laplaceraum erhält man damit wieder die modifizierte Besselsche Differentialgleichung 0-ter Ordnung wie in Gleichung (2.3.27):

$$r^2 \cdot g_s''(r) + r \cdot g_s'(r) - r^2 \frac{s \cdot S}{K} \cdot g_s(r) = 0 \quad (3.3.6)$$

Somit hat man mit der Substitution $T_s = \sqrt{\frac{sS}{K}}$ und $\tau = Cr$ als Lösung wieder folgende Linearkombination für $\tau > Cr_{\text{max}}$:

$$\tilde{g}_s(\tau) = C_s \cdot I_0 \left(T_s \frac{\tau}{C} \right) + D_s \cdot K_0 \left(T_s \frac{\tau}{C} \right) \quad (3.3.7)$$

Somit ist für die Lösung im Laplaceraum nun für alle $r > 0$ ein Fundamentalsystem berechnet. Es müssen nun nur noch die Randwerte eingearbeitet werden.

3.4. Einarbeitung der Randwerte

Ich habe in den beiden Abschnitten $(r_{\text{well}}, r_{\text{max}}]$ und $[r_{\text{max}}, r_{\infty})$ jeweils eine Fundamentalsystem berechnet, welches die gegebene Differentialgleichung löst:

$$\forall \tau \in C \cdot (r_{\text{well}}, r_{\text{max}}] \quad \tilde{g}_s^1(\tau) = A_s u_{1,s}(\tau) + B_s u_{2,s}(\tau) \quad (3.4.1)$$

$$\forall \tau \in C \cdot [r_{\text{max}}, r_{\infty}) \quad \tilde{g}_s^2(\tau) = C_s \cdot I_0\left(T_s \frac{\tau}{C}\right) + D_s \cdot K_0\left(T_s \frac{\tau}{C}\right) \quad (3.4.2)$$

Nun müssen die Koeffizienten so gewählt werden, dass die Randwerte richtig angenommen werden. Als Bedingung in r_{max} verlange ich, dass die Funktion stetig differenzierbar wird, also:

$$\begin{aligned} \tilde{g}_s^1(Cr_{\text{max}}) &= \tilde{g}_s^2(Cr_{\text{max}}) \\ (\tilde{g}_s^1)'(Cr_{\text{max}}) &= (\tilde{g}_s^2)'(Cr_{\text{max}}) \end{aligned} \quad (3.4.3)$$

Explizit heißt das:

$$\begin{aligned} A_s u_{1,s}(Cr_{\text{max}}) + B_s u_{2,s}(Cr_{\text{max}}) &= C_s I_0(T_s r_{\text{max}}) + D_s K_0(T_s r_{\text{max}}) \\ A_s u'_{1,s}(Cr_{\text{max}}) + B_s u'_{2,s}(Cr_{\text{max}}) &= C_s I_1(T_s r_{\text{max}}) \frac{T_s}{C} - D_s K_1(T_s r_{\text{max}}) \frac{T_s}{C} \end{aligned} \quad (3.4.4)$$

Da durch die numerische Integration der Differentialgleichung in $[r_{\text{min}}, r_{\text{max}}]$ automatisch die Ableitung berechnet wird, stellt dies kein Problem dar.

Mit der Lösung \tilde{g}_s^1 kann man nun die Randbedingung am Brunnen einarbeiten:

$$\forall s > 0 \quad \lim_{\tau \rightarrow Cr_{\text{well}}} \tau \cdot (\tilde{g}_s^1)'(\tau) = Q(r_{\text{well}}) \cdot \frac{1}{s} \quad (3.4.5)$$

Ich nehme $r_{\text{well}} < r_{\text{min}}$ an. Somit kann ich diese Randbedingung mit den Lösungen der Frobenius-Methode einarbeiten:

$$u'_{1,s}(\tau) = \sum_{k=0}^{\infty} (k+1) g_{1,s}^{(k+1)} \cdot \tau^k \quad (3.4.6)$$

$$u'_{2,s}(\tau) = \sum_{k=0}^{\infty} (k+1) g_{2,s}^{(k+1)} \cdot \tau^k + \ln \tau \cdot u'_{1,s}(\tau) + \frac{1}{\tau} u_{1,s}(\tau) \quad (3.4.7)$$

Damit ergibt sich:

1. Für $r_{\text{well}} = 0$:

$$B_s = Q(r_{\text{well}} = 0) \cdot \frac{1}{s} \quad (3.4.8)$$

Hier zeigt sich der Nutzen der Frobenius-Methode, da sich dieser Randwert durch das logarithmische Verhalten der zweiten Basislösung $u_{2,s}$ kanonisch einpflegen lässt. Der zweite Koeffizient A_s ist dann noch zu determinieren.

2. Für $r_{\text{well}} > 0$:

$$\begin{aligned} Cr_{\text{well}} \cdot (\tilde{g}_s^1)'(Cr_{\text{well}}) &= Q(r_{\text{well}}) \cdot \frac{1}{s} \\ \Leftrightarrow A_s u'_{1,s}(Cr_{\text{well}}) + B_s u'_{2,s}(Cr_{\text{well}}) &= \frac{Q(r_{\text{well}})}{Cr_{\text{well}}} \cdot \frac{1}{s} \end{aligned}$$

Somit erhält man in diesem Fall eine lineare Beziehung zwischen A_s und B_s :

$$A_s = -\frac{u'_{2,s}(Cr_{\text{well}})}{u'_{1,s}(Cr_{\text{well}})} \cdot B_s + \frac{Q(r_{\text{well}}) \cdot \frac{1}{s}}{u'_{1,s}(Cr_{\text{well}})} \quad (3.4.9)$$

Mit der Lösung $\tilde{g}_s^2(\tau)$ kann die Randbedingung in r_∞ eingearbeitet werden:

$$\forall_{s>0} \lim_{\tau \rightarrow Cr_\infty} \tilde{g}_s^2(\tau) = 0 \quad (3.4.10)$$

Ich nehme auch hier $r_\infty > r_{\max}$ an. Damit ergibt sich:

1. Für $r_\infty = \infty$:

$$C_s = 0 \quad (3.4.11)$$

Da $I_0(x) \xrightarrow{x \rightarrow \infty} \infty$. Der zweite Koeffizient D_s ist dann noch zu determinieren, da $K_0(x) \xrightarrow{x \rightarrow \infty} 0$ (vgl. Abschnitt 2.3.2.1).

2. Für $r_\infty < \infty$:

$$\begin{aligned} \tilde{g}_s^2(r_\infty) &= 0 \\ \Leftrightarrow C_s \cdot I_0(T_s r_\infty) + D_s \cdot K_0(T_s r_\infty) &= 0 \end{aligned}$$

Somit erhält man auch in diesem Fall eine lineare Beziehung zwischen C_s und D_s :

$$C_s = -\frac{K_0(T_s r_\infty)}{I_0(T_s r_\infty)} \cdot D_s \quad (3.4.12)$$

Man erhält nun in allen Fällen ein lineares Gleichungssystem

$$M \cdot X = R \quad (3.4.13)$$

für die Koeffizienten A_s, B_s, C_s und D_s , wobei:

$$X = (A_s, B_s, C_s, D_s)^T \quad (3.4.14)$$

Es entstehen wie bei der Berechnung der Theis-Lösung in Abschnitt 2.3 vier Fälle:

1. $r_{\text{well}} = 0$ und $r_\infty = \infty$:

Hier sind die Koeffizienten $B_s = \frac{Q(0)}{s}$ und $C_s = 0$ schon determiniert. Für die Matrix M und die rechte Seite R ergibt sich aus (3.4.3):

$$M = \begin{pmatrix} u_{1,s}(Cr_{\max}) & 0 & 0 & -K_0(T_s r_{\max}) \\ u'_{1,s}(Cr_{\max}) & 0 & 0 & K_1(T_s r_{\max}) \cdot \frac{T_s}{C} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (3.4.15)$$

$$R = \begin{pmatrix} -\frac{Q(0)}{s} u_{2,s}(Cr_{\max}) \\ -\frac{Q(0)}{s} u'_{2,s}(Cr_{\max}) \\ \frac{Q(0)}{s} \\ 0 \end{pmatrix} \quad (3.4.16)$$

2. $r_{\text{well}} > 0$ und $r_\infty = \infty$:

Hier ist nur der Koeffizient $C_s = 0$ schon determiniert. Es ergibt sich:

$$M = \begin{pmatrix} u_{1,s}(Cr_{\max}) & u_{2,s}(Cr_{\max}) & 0 & -K_0(T_s r_{\max}) \\ u'_{1,s}(Cr_{\max}) & u'_{2,s}(Cr_{\max}) & 0 & K_1(T_s r_{\max}) \cdot \frac{T_s}{C} \\ u'_{1,s}(Cr_{\text{well}}) & u'_{2,s}(Cr_{\text{well}}) & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (3.4.17)$$

$$R = \begin{pmatrix} 0 \\ 0 \\ \frac{Q(r_{\text{well}})}{Cr_{\text{well}}} \cdot \frac{1}{s} \\ 0 \end{pmatrix} \quad (3.4.18)$$

3. $r_{\text{well}} = 0$ und $r_{\infty} < \infty$:

Hier ist jetzt nur der Koeffizient $B_s = \frac{Q(0)}{s}$ schon determiniert. Es ergibt sich wieder:

$$M = \begin{pmatrix} u_{1,s}(Cr_{\max}) & 0 & -I_0(T_s r_{\max}) & -K_0(T_s r_{\max}) \\ u'_{1,s}(Cr_{\max}) & 0 & -I_1(T_s r_{\max}) \cdot \frac{T_s}{C} & K_1(T_s r_{\max}) \cdot \frac{T_s}{C} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & I_0(T_s r_{\infty}) & K_0(T_s r_{\infty}) \end{pmatrix} \quad (3.4.19)$$

$$R = \begin{pmatrix} -\frac{Q(0)}{s} u_{2,s}(Cr_{\max}) \\ -\frac{Q(0)}{s} u'_{2,s}(Cr_{\max}) \\ \frac{Q(0)}{s} \\ 0 \end{pmatrix} \quad (3.4.20)$$

4. $r_{\text{well}} > 0$ und $r_{\infty} < \infty$:

Hier ist noch gar kein Koeffizient determiniert. Es ergibt sich letztlich:

$$M = \begin{pmatrix} u_{1,s}(Cr_{\max}) & u_{2,s}(Cr_{\max}) & -I_0(T_s r_{\max}) & -K_0(T_s r_{\max}) \\ u'_{1,s}(Cr_{\max}) & u'_{2,s}(Cr_{\max}) & -I_1(T_s r_{\max}) \frac{T_s}{C} & K_1(T_s r_{\max}) \frac{T_s}{C} \\ u'_{1,s}(Cr_{\text{well}}) & u'_{2,s}(Cr_{\text{well}}) & 0 & 0 \\ 0 & 0 & I_0(T_s r_{\infty}) & K_0(T_s r_{\infty}) \end{pmatrix} \quad (3.4.21)$$

$$R = \begin{pmatrix} 0 \\ 0 \\ \frac{Q(r_{\text{well}})}{Cr_{\text{well}}} \cdot \frac{1}{s} \\ 0 \end{pmatrix} \quad (3.4.22)$$

Damit ist die Lösung im Laplaceraum vollständig determiniert und man kann die Funktion in τ mit den Lösungen (3.4.1) und (3.4.2) auswerten, um sie dann mit dem Stehfest-Algorithmus zu invertieren.

Nun müssen die drei genannten Lösungsabschnitte für die Frobenius-Methode, das Rosenbrock-Verfahren und die Theis-Lösung noch untersucht werden, um die Stabilität und Güte der Lösung zu sichern.

3.5. Die Wahl der Lösungsabschnitte

Um den Fehler zur tatsächlichen Lösung zu minimieren, ist die richtige Wahl der Intervalle für die drei Lösungsabschnitte essentiell. Ich stelle nun meine Wahl der drei Lösungsintervalle

$$I_1 = [r_{\text{well}}, r_{\text{min}}] \quad (3.5.1)$$

$$I_2 = [r_{\text{min}}, r_{\text{max}}] \quad (3.5.2)$$

$$I_3 = [r_{\text{max}}, r_{\infty}] \quad (3.5.3)$$

vor und untersuche den Einfluss der Größen r_{min} und r_{max} auf das Lösungsverhalten.

3.5.1. Das Intervall $I_1 = [r_{\text{well}}, r_{\text{min}}]$ mit der Frobenius-Methode

Da im ersten Intervall $I_1 = [r_{\text{well}}, r_{\text{min}}]$ die Lösung durch eine verallgemeinerte Potenzreihe approximiert wird, muss man dort einerseits beachten, überhaupt im Konvergenzradius zu bleiben und andererseits sollte man nahe am Entwicklungspunkt $r = 0$ bleiben, um den Fehler, der durch das Abschneiden der Potenzreihe entsteht, klein zu halten. Durch die obige Substitution $\tau = C \cdot r$ erhält man nach dem Satz von Fuchs einen Konvergenzradius von 1. Ich habe mir daher als Abschneidewert für die Potenzreihe $n_{\text{max}} = 40$ und als Intervallgrenze $r_{\text{min}} = \frac{1}{4}C^{-1}$ gewählt, da die Potenzreihen der verketteten Funktionen relativ schlecht konvergieren. Allerdings wird der zweite Wert r_{max} , so wie ich ihn gleich definieren werde, für kleine Zeiten unterhalb von $\frac{1}{4}C^{-1}$ liegen, weshalb ich folgende Definition für r_{min} gebe:

$$r_{\text{min}} = \min \left\{ \frac{r_{\text{max}}}{2}, \frac{1}{4}C^{-1} \right\} \quad (3.5.4)$$

3.5.2. Das Intervall $I_2 = [r_{\text{min}}, r_{\text{max}}]$ mit dem Rosenbrock-Verfahren

Im zweiten Intervall $I_2 = [r_{\text{min}}, r_{\text{max}}]$ approximiere ich die Lösung numerisch durch das implizite Rosenbrock-Verfahren. Dieses implizite Verfahren wird notwendig, da es sich hier um eine steife Differentialgleichung handelt. Da man für jeden Zeitschritt mit dem Stehfest-Algorithmus wenigstens 6 s -Auswertungen im Laplaceraum benötigt und das implizite Verfahren sehr rechenaufwendig ist, sollte man dieses Intervall so klein wie möglich wählen. Zum anderen entsteht aber durch eine kleine Wahl von r_{max} ein größerer Fehler von $|K^{\text{CG}}(r_{\text{max}}) - K_{\text{efu}}|$ bzw. $|T_{\text{H}}^{\text{CG}}(r_{\text{max}}) - T_{\text{G}}|$, wodurch die Lösung im dritten Intervall verfälscht wird. Deshalb setze ich als Erstes einen maximalen relativen Fehler $\varepsilon_{\text{error}}$ zwischen $K^{\text{CG}}(r_{\text{max}})$ und K_{efu} bzw. $T_{\text{H}}^{\text{CG}}(r_{\text{max}})$ und T_{G} fest und leite daraus eine obere Grenze für r_{max} ab, welche ich mit r_{lim} bezeichne. Ich betrachte folgende 3 Fälle:

1. Im 2D-Modell:

$$\begin{aligned} \frac{|T_{\text{H}}^{\text{CG}}(r_{\text{max}}) - T_{\text{G}}|}{T_{\text{G}}} &< \varepsilon_{\text{error}} \\ \Leftrightarrow 1 - \exp\left(\frac{-\frac{\sigma^2}{2}}{1 + \tau^2}\right) &< \varepsilon_{\text{error}} \\ \Leftrightarrow \sqrt{-\frac{\frac{\sigma^2}{2}}{\ln(1 - \varepsilon_{\text{error}})} - 1} &< \tau \end{aligned}$$

Damit erhält man:

$$r_{\text{lim}} = \frac{1}{C} \cdot \sqrt{-\frac{\frac{\sigma^2}{2}}{\ln(1 - \varepsilon_{\text{error}})} - 1} \quad (3.5.5)$$

2. Im 3D-Modell mit harmonischer Mittelung am Brunnen, d.h. $\chi = \sigma^2(\gamma(e) - 1) \leq 0$ (vgl. Formel (1.1.16)):

$$\begin{aligned} \frac{|K^{\text{CG}}(r_{\text{max}}) - K_{\text{efu}}|}{K_{\text{efu}}} &< \varepsilon_{\text{error}} \\ \Leftrightarrow 1 - \exp\left(\frac{\chi}{\sqrt{1 + \tau^2}^3}\right) &< \varepsilon_{\text{error}} \\ \Leftrightarrow \sqrt{\left(\frac{\chi}{\ln(1 - \varepsilon_{\text{error}})}\right)^{\frac{2}{3}} - 1} &< \tau \end{aligned}$$

Damit erhält man:

$$r_{\text{lim}} = \frac{1}{C} \cdot \sqrt{\left(\frac{\chi}{\ln(1 - \varepsilon_{\text{error}})}\right)^{\frac{2}{3}} - 1} \quad (3.5.6)$$

3. Im 3D-Modell mit arithmetischer Mittelung am Brunnen, d.h. $\chi = \sigma^2\gamma(e) \geq 0$ (vgl. Formel (1.1.16)):

$$\begin{aligned} \frac{|K^{\text{CG}}(r_{\text{max}}) - K_{\text{efu}}|}{K_{\text{efu}}} &< \varepsilon_{\text{error}} \\ \Leftrightarrow \exp\left(\frac{\chi}{\sqrt{1 + \tau^2}^3}\right) - 1 &< \varepsilon_{\text{error}} \\ \Leftrightarrow \sqrt{\left(\frac{\chi}{\ln(1 + \varepsilon_{\text{error}})}\right)^{\frac{2}{3}} - 1} &< \tau \end{aligned}$$

Damit erhält man:

$$r_{\text{lim}} = \frac{1}{C} \cdot \sqrt{\left(\frac{\chi}{\ln(1 + \varepsilon_{\text{error}})}\right)^{\frac{2}{3}} - 1} \quad (3.5.7)$$

Als Standardwert wähle ich für $\varepsilon_{\text{error}}$ Folgendes:

$$\varepsilon_{\text{error}} = 0.01 \quad (3.5.8)$$

Somit soll der relative Fehler kleiner als 1% sein. Allerdings ist diese Wahl für kleine Zeiten eher ungeeignet, da die Lösung für $r \rightarrow \infty$ sehr schnell gegen 0 konvergiert. Um diese Geschwindigkeit abzuschätzen, nutze ich die homogene Theis-Lösung (vgl. Abschnitt 2.3) als Vergleichsfunktion:

$$h(r, t) = \frac{Q_w}{4\pi T} E_1(u) \quad (3.5.9)$$

$$E_1(u) = \int_u^{\infty} \frac{e^{-\lambda}}{\lambda} d\lambda \quad (3.5.10)$$

$$u = \frac{r^2 S}{4Tt} \quad (3.5.11)$$

Dabei hängt die Lösung nur von dem dimensionslosen Parameter u (3.5.11) ab und es ist hier $T = K$ für die 3D Lösung zu wählen. Nun ist mein Ansatz, dass ich ein geeignetes T wähle, dann eine Schranke für u angebe und daraus r_{max} ableite. Ich wähle folgende Parameter als gewichtete Mittelwerte zwischen den Grenzwerten der Funktionen T_{H}^{CG} bzw. K^{CG} an den Rändern $r = 0$ und $r = \infty$:

$$K_{\text{mid}} = \frac{1}{3}(K_{\text{efu}} + 2 \cdot K_{\text{well}}) \quad (3.5.12)$$

$$T_{\text{mid}} = \frac{1}{3}(T_{\text{G}} + 2 \cdot T_{\text{well}}) \quad (3.5.13)$$

Da man das Lösungsintervall durch dieses Vorgehen Richtung Brunnen⁶ verschiebt, wichte ich in den Gleichungen (3.5.12) und (3.5.13) jeweils zu Gunsten des Leitwertes am Brunnen. Das Exponential-Integral in (3.5.10) konvergiert monoton gegen 0 und deshalb setze hier eine untere Grenze für u , mit der ich später versuche, die Lösung effektiv abzuschneiden:

$$u \leq u_{\text{lim}} \quad (3.5.14)$$

Für alle Werte r und t , welche $u \geq u_{\text{lim}}$ erfüllen, möchte ich somit die Lösung der Differentialgleichung durch die homogene Fernfeldlösung approximieren. Durch verschiedene Tests nehme ich einen Standardwert von $u_{\text{lim}} = \frac{1}{4}$ an. Dann ergibt sich für die Brunnenfunktion:

$$\left| \frac{1}{4\pi} E_1 \left(\frac{1}{4} \right) \right| \approx 0.083 \quad (3.5.15)$$

$$\left| \frac{\partial}{\partial x} \left(\frac{1}{4\pi} E_1 \right) \left(\frac{1}{4} \right) \right| \approx 0.248 \quad (3.5.16)$$

Da für die Standardwerte der Pumptest Q_w und T in etwa die selben Größenordnungen haben, ist damit die Theis-Lösung ebenfalls gut abgeschätzt. Damit möchte ich r_{max} in Abhängigkeit von t in den verschiedenen Fällen angeben:

1. Im 2D-Modell:

$$\begin{aligned} \frac{r^2 S}{4T_{\text{mid}} t} &\leq u_{\text{lim}} \\ \Leftrightarrow \sqrt{\frac{4u_{\text{lim}} T_{\text{mid}} t}{S}} &\geq r \end{aligned}$$

Damit ergibt sich:

$$r_{\text{max}}(t) = \min \left\{ \sqrt{\frac{4u_{\text{lim}} T_{\text{mid}} t}{S}}, r_{\text{lim}} \right\} \quad (3.5.17)$$

2. Im 3D-Modell:

$$\begin{aligned} \frac{r^2 S}{K_{\text{mid}} t} &\leq u_{\text{lim}} \\ \Leftrightarrow \sqrt{\frac{4u_{\text{lim}} K_{\text{mid}} t}{S}} &\geq r \end{aligned}$$

Damit ergibt sich:

$$r_{\text{max}}(t) = \min \left\{ \sqrt{\frac{4u_{\text{lim}} K_{\text{mid}} t}{S}}, r_{\text{lim}} \right\} \quad (3.5.18)$$

In [Häfner et al., 1992, S. 406] wird darauf hingewiesen, dass die Lösung nach dem hier verwendeten Schema für kleine Zeiten instabil wird. Durch obiges Vorgehen wird genau dem entgegengewirkt, da ich den Bereich, in dem die Funktion im Laplaceraum nach der gegebenen Differentialgleichung entwickelt wird, mit fortschreitender Zeit erst vergrößere.

3.5.3. Das Intervall $I_3 = [r_{\text{max}}, r_{\infty})$ mit homogener Lösung

Da die gegebenen Leitfähigkeits- bzw. Durchlässigkeitsfunktionen jeweils Funktionen sind, welche einen glatten Übergang zwischen zwei Werten darstellen, ist es numerisch sinnvoll, für große Radien nur noch die Lösung für homogene Leitfähigkeiten bzw. Durchlässigkeiten zu berechnen. Da es für diese Lösungen bekannte symbolische Funktionen gibt, erhalten wir eine gute Möglichkeit, die Lösung zum einen stabil für große Radien zu halten und zum anderen können wir damit die Randbedingung in $r = r_{\infty}$ einpflegen. In Kapitel 4 werde ich untersuchen, inwiefern die Wahl von r_{max} einen Einfluss auf die Lösung hat.

⁶Also Richtung $r = 0$.

3.6. Implementierung der erweiterten Theis-Lösung

Die Implementierung der Lösung erfolgte in Fortran, da diese Sprache immer noch eine der wichtigsten und schnellsten im Bereich der numerischen Berechnung ist. Ich werde hier nur den von mir selbst verfassten Quellcode ausführlich vorstellen und übernommene Routinen lediglich nennen.

3.6.1. Implementierung der Regel von Faà di Bruno

Um die in Abschnitt 2.4.4 hergeleitete Formel für verkettete Potenzreihen

$$\forall_{n \geq 1} h_n = \sum_{j=1}^{P(n)} \binom{|\mathbf{k}_{n,j}|}{\mathbf{k}_{n,j}} \cdot f_{|\mathbf{k}_{n,j}|} \cdot \prod_{m=1}^n g_m^{k_{n,j,m}} \quad (3.6.1)$$

zu implementieren habe ich, wie in Abschnitt 2.4.3 schon beschrieben, die Menge der Partitionen $T_n = \{\mathbf{k}_{n,1}, \dots, \mathbf{k}_{n,P(n)}\}$ einer natürlichen Zahl $n \geq 2$ lexikografisch geordnet. Diese Ordnung \succ ist für Vektoren $(a_i)_{i=1}^n, (b_i)_{i=1}^n \in \mathbb{N}_0^n$ wie folgt definiert:

$$(a_1) \succ (b_1) \quad :\Leftrightarrow \quad a_1 > b_1 \quad (3.6.2)$$

$$(a_i)_{i=1}^n \succ (b_i)_{i=1}^n \quad :\Leftrightarrow \quad (a_n > b_n) \vee \left((a_n = b_n) \wedge \left((a_i)_{i=1}^{n-1} \succ (b_i)_{i=1}^{n-1} \right) \right) \quad (3.6.3)$$

Damit ist die Menge T_n eindeutig geordnet:

$$(0, \dots, 0, 1) = \mathbf{k}_{n,1} \succ \mathbf{k}_{n,2} \succ \dots \succ \mathbf{k}_{n,P(n)} = (n, 0, \dots, 0) \quad (3.6.4)$$

Um dies zu nutzen, habe ich die Funktion `nextpart` implementiert, welche zu einer gegebenen Partition die lexikografisch nächst kleinere ausgibt:

$$\text{nextpart}(\mathbf{k}_{n,i}) = \mathbf{k}_{n,i+1} \quad (3.6.5)$$

Da erste und letzte Partition immer eindeutig gegeben sind, muss man zur Berechnung von h_n den ersten Partitionsvektor auf $\mathbf{k}_{n,1}$ setzen, dann den zugehörigen Summanden aus (3.6.1) berechnen, danach `nextpart` auf den Partitionsvektor anwenden und diesen Vorgang solange wiederholen, bis der Partitionsvektor $\mathbf{k}_{n,P(n)}$ ist. In Pseudocode ist dies in Programmcode 1 skizziert. Die Implementierung dieser Regel findet man im Anhang in den Programmcodes 10 und 12.

Programmcode 1: Implementierung der Regel von Faà di Bruno

```

1 Partition      = (0, ..., 0, 1)
  h(n)          = 0.0
  do
    h(n)        = h(n) + [...]
5    if (Partition(1) == n) exit
    Partition    = nextpart(Partition)
  enddo

```

3.6.2. Der Stehfest-Algorithmus

Diesen Algorithmus habe ich nach den Betrachtungen in Kapitel 2.1.2.2 implementiert. Die Koeffizienten c_n (vgl. Formel (2.1.17)) berechne ich dabei als einen Vektor

$$\underline{c}(N) = (c_1, \dots, c_N) \quad (3.6.6)$$

in einer separaten Funktion. Die Implementierung findet sich in Programmcode 8.

Der Stehfest-Algorithmus verlangt, dass der gegebenen Funktion alle Parameter, welche diese braucht, in einem einzigen Array namens `para` übergeben werden, um den Aufruf der übergebenen Funktion zu standardisieren. Der Aufruf des Algorithmus sieht wie folgt aus:

$$\mathbf{f} = \text{nlinvsteh}(\text{func}, \text{para}, \mathbf{r}, \mathbf{t}, \mathbf{n}) \quad (3.6.7)$$

Der Name `nlinvsteh` leitet sich dabei von “numerical Laplace-inversion Stehfest” ab. Die Bedeutung der Variablen und Parameter ist in folgender Tabelle gegeben:

Variablen in <code>nlinvsteh</code>	Optional	Beschreibung
<code>f</code>	-	Ausgabe der Lösung als Matrix: $\mathbf{f} = (f_{ij})_{ij}$ mit $f_{ij} = f(r_i, t_j)$
<code>func</code>	Nein	Übergabe der zu invertierenden Funktion: <code>func(r, s, para)</code>
<code>para</code>	Nein	Parameter zur Manipulation der Funktion <code>func</code> : <code>para = (p₁, ..., p_k)</code>
<code>r</code>	Nein	Array der gegebenen Radien: $\mathbf{r} = (r_1, \dots, r_n)$
<code>t</code>	Nein	Array der gegebenen Zeitpunkte: $\mathbf{t} = (t_1, \dots, t_m)$
<code>n</code>	Nein	Grad der Stehfest-Approximation

Ich habe also zusätzlich die Übergabe einer Raumvariable $\mathbf{r} = (r_1, \dots, r_n)$ zugelassen, um auf das gegebene Problem einzugehen. Diese werden lediglich an die Funktion `func` weitergegeben und haben auf den Stehfest-Algorithmus keinen Einfluss. Die Implementierung des Algorithmus ist in Programmcode 7 gegeben.

3.6.3. Übernommene Routinen

Ich habe meine Module und Programme nach den Standards der Fortran-Bibliothek der CHS-Abteilung des UFZ Leipzig verfasst, um diese dort einzubetten. Diese Bibliothek stellt eine Fülle von Standardfunktionen bereit, von denen ich einige verwendet habe. Diese sind in folgender Tabelle aufgelistet:

Funktion	Beschreibung
<code>factorial(n)</code>	Berechnung der Fakultät: $n!$
<code>poly(x, (c₀, ..., c_n))</code>	Auswertung von Polynomen: $c_0 + c_1x + \dots + c_nx^n$
<code>binomcoeffi(x, n)</code>	Berechnung des Binomialkoeffizienten: $\binom{x}{n}$
<code>nextpart((k₁, ..., k_n))</code>	Berechnung der zu $(k_i)_i$ lexikografisch nächstgrößeren Partition von n
<code>isgoodpart((k₁, ..., k_n))</code>	Überprüfung ob $(k_i)_i$ eine Partition von n ist
<code>interpol((f_i)_i, (x_i)_i, (ξ_j)_j)</code>	Auswertung der Punkte ξ_j an der linearen Interpolation der Werte (f_i, x_i)
<code>solve_linear_equations(M, R)</code>	Lösung des linearen Gleichungssystems: $M \cdot X = R$

Dabei wurden die Funktionen `nextpart` und `isgoodpart` von mir selbst implementiert (siehe Anhang: Programmcode 5 und 6). Die restlichen Routinen sind kanonische Implementierungen der jeweiligen Funktion, weshalb ich auf diese nicht weiter eingehen möchte.

Des Weiteren habe ich einige Routinen aus den Numerical-Recipes [Press et al., 1992] übernommen. Diese sind in folgender Tabelle genannt:

Funktion	Beschreibung
<code>bessk0(x)</code>	Berechnung der modifizieren Besselfunktion: $K_0(x)$ ([Press et al., 1992, S. 231])
<code>bessk1(x)</code>	Berechnung der modifizieren Besselfunktion: $K_1(x)$ ([Press et al., 1992, S. 232])
<code>RBstiff</code>	Eine Implementierung des Rosenbrock-Verfahrens ([Press et al., 1992, S. 732ff])

Dabei ist zu erwähnen, dass die Funktion `RBstiff` genau zwei Arrays $(f_i)_i$ und $(x_i)_i$ ausgibt, welche die berechneten Funktionspunkte darstellen. Da es sich um ein Verfahren mit adaptiver Schrittweitensteuerung handelt, ist die Größe und Schrittweite von $(x_i)_i$ vor dem Ausführen der Routine nicht bekannt. Um die Funktion danach an beliebigen Punkten auszuwerten, ist man auf die Routine `interp0l` angewiesen, welche die gegebenen Funktionspunkte dann linear interpoliert. Die modifizierten Besselfunktionen brauche ich für eine stabile Lösung der Differentialgleichung im Fernfeld.

3.6.4. Die Hauptroutine

Kommen wir zu den Hauptroutinen, in denen die Lösung der Grundwassergleichung umgesetzt wurde. Die Lösung $h(r, t)$ der Differentialgleichung (1.1.3) wurde von mir mit den Funktionen `ext_theis2D` und `ext_theis3D` implementiert. Der Aufruf sieht im *2D*-Fall wie folgt aus:

$$\mathbf{h} = \text{ext_theis2D}(\text{rad}, \text{time}, \text{params}, \text{inits}, \text{prop}, \text{grad}, \text{steh_n}, \text{output}) \quad (3.6.8)$$

Die Bedeutung der Variablen und Parameter ist in folgender Tabelle gegeben:

Variablen in <i>2D</i>	Optional	Beschreibung
<code>h</code>	-	Ausgabe der Lösung als Matrix: $\mathbf{h} = (h_{ij})_{ij}$ mit $h_{ij} = h(r_i, t_j)$
<code>rad</code>	Nein	Array der gegebenen Radien: $\text{rad} = (r_1, \dots, r_n)$
<code>time</code>	Nein	Array der gegebenen Zeitpunkte: $\text{time} = (t_1, \dots, t_m)$
<code>params</code>	Nein	Benötigte Parameter der Differentialgleichung: $\text{params} = (T_G, \sigma^2, \ell, S)$
<code>inits</code>	Nein	Randbedingungen der Differentialgleichung: $\text{inits} = (Q_w)$
<code>prop</code>	Ja	Proportionalitätsfaktor ζ (Standardwert $\zeta = 1.6$)
<code>grad</code>	Ja	Grad der Taylorapproximation n_{\max} (Standardwert $n_{\max} = 40$)
<code>steh_n</code>	Ja	Grad der Stehfestapproximation N (Standardwert $N = 6$)
<code>output</code>	Ja	Wahl ob während der Berechnung Informationen ausgegeben werden sollen (Standardwert <code>false</code>)

In 3D-Fall ist der Aufruf ganz ähnlich, nur mit angepassten Übergabewerten:

$$\mathbf{h} = \text{ext_theis3D}(\text{rad}, \text{time}, \text{params}, \text{inits}, \text{bc}, \text{prop}, \text{grad}, \text{steh_n}, \text{output}) \quad (3.6.9)$$

Die Bedeutung der Variablen und Parameter für den 3D-Fall ist in folgender Tabelle gegeben:

Variablen in 2D	Optional	Beschreibung
h	-	Ausgabe der Lösung als Matrix: $\mathbf{h} = (h_{ij})_{ij}$ mit $h_{ij} = h(r_i, t_j)$
rad	Nein	Array der gegebenen Radien: $\text{rad} = (r_1, \dots, r_n)$
time	Nein	Array der gegebenen Zeitpunkte: $\text{time} = (t_1, \dots, t_m)$
params	Nein	Benötigte Parameter der Differentialgleichung: $\text{params} = (K_G, \sigma^2, \ell, S, e)$
inits	Nein	Randbedingungen der Differentialgleichung: $\text{inits} = (L, Q_w)$
bc	Ja	Art der Randbedingung für K_{well} : $\text{true} \hat{=} K_H$ und $\text{false} \hat{=} K_A$ (Standardwert $\text{bc} = \text{true}$)
prop	Ja	Proportionalitätsfaktor ζ (Standardwert $\zeta = 1.6$ bei $\text{bc} = \text{true}$, sonst $\frac{\zeta}{2}$)
grad	Ja	Grad der Taylorapproximation n_{max} (Standardwert $n_{\text{max}} = 40$)
steh_n	Ja	Grad der Stehfestapproximation N (Standardwert $N = 6$)
output	Ja	Wahl ob während der Berechnung Informationen ausgegeben werden sollen (Standardwert false)

Die Berechnungen sind so optimiert, dass man die Radien $\text{rad} = (r_1, \dots, r_n)$ und Zeitpunkte $\text{time} = (t_1, \dots, t_m)$ als Arrays übergeben kann und die Lösung nicht für jedes Paar (r_i, t_j) einzeln aufgerufen werden muss. Das bedeutet, dass die nötigen Potenzreihen nur ein einziges Mal berechnet werden müssen. Die Abhängigkeiten der Routinen untereinander ist in Abbildung 6 durch einen Abhängigkeitsbaum dargestellt.

Der Funktionsaufbau ist in Pseudocode für 2D mit Programmcode 2 gegeben.

Programmcode 2: Ablauf der Funktion `ext_theis2d`

```

1  function ext_theis2d(rad,time,params,inits,prop,grad,steh_n,output)
    !Prüfe Eingaben
5  !Berechne benötigte Größen:
    C      = ...
    !Fasse Variablen für den Stehfestalgorithmus zusammen:
    Values = (TG,sig**2,C,S,Qw,n_max,steh_n,output)
10  !Rufe Stehfestalgorithmus auf:
    NLInvsteh(lap_ext_theis2d,values,C*rad,time,steh_n)
end function ext_theis2d

```

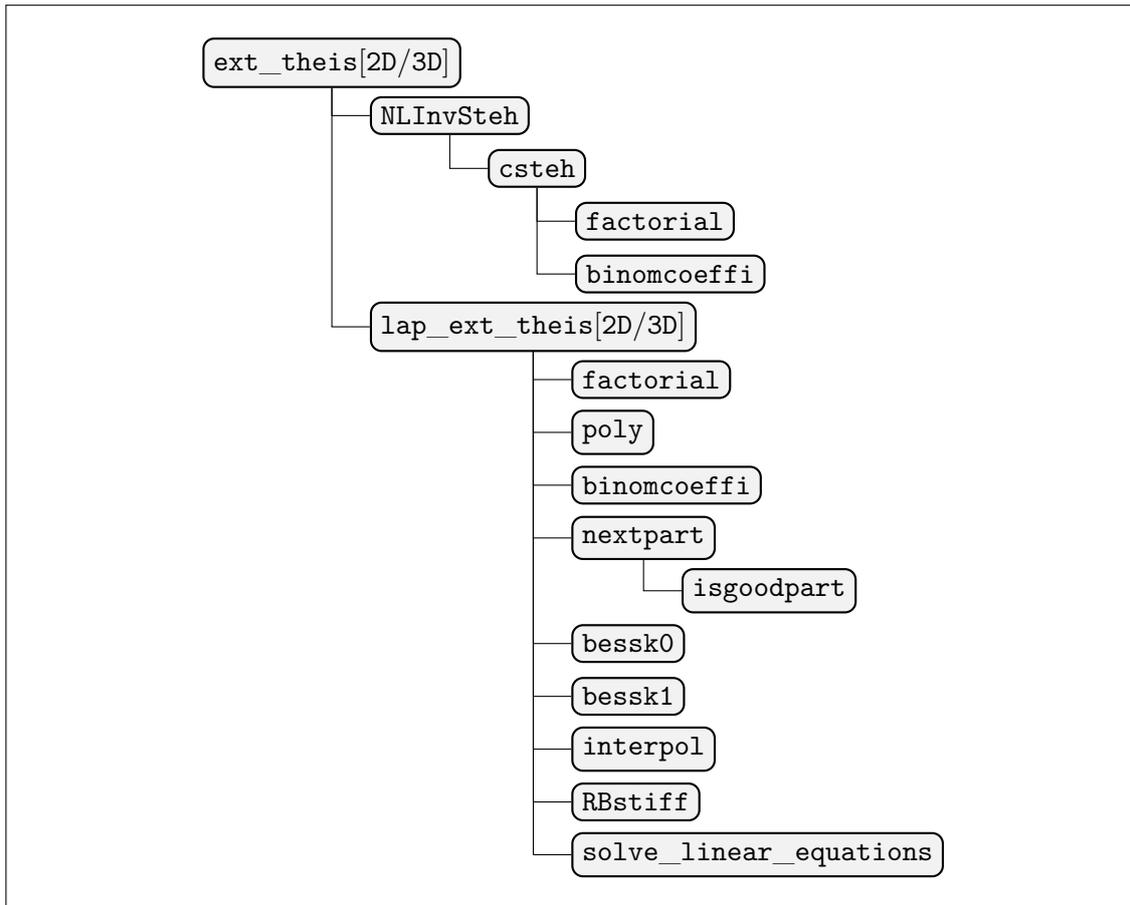


Abbildung 6: Abhängigkeitsbaum der implementierten Routinen

Im 3D-Fall ist der Funktionsaufbau ganz ähnlich, es müssen nur weitere benötigte Größen berechnet werden. Der Aufbau ist in Pseudocode mit Programmcode 3 gegeben.

Programmcode 3: Ablauf der Funktion `ext_theis3d`

```

1  function ext_theis3d(rad,time,params,inits,bc,prop,grad,ste_h_n,output)
    !Prüfe Eingaben
5
    !Berechne benötigte Größen:
    C      = ...
    gamma  = ...
    Kefu   = ...
    Kwell  = ...
10    chi   = ...

    !Fasse Variablen für den Stehfestalgorithmus zusammen:
    Values = (KG,Kwell,Kefu,chi,C,S,L,Qw,n_max,ste_h_n,output)
15

    !Rufe Stehfestalgorithmus auf:
    NLInvsteh(lap_ext_theis3d,values,C*rad,time,ste_h_n)

end function ext_theis3d

```

Die Funktionen `lap_ext_theis2d` und `lap_ext_theis3d` verlaufen danach sehr ähnlich. Der Stehfest-Algorithmus `NLInvSteh` ist so optimiert, dass er alle benötigten s -Punkte berechnet und in einem einzigen Array $\mathbf{s} = (s_1, \dots, s_k)$ übergibt. Somit muss die Funktion `lap_ext_theis[2d/3d]` nur ein einziges Mal aufgerufen werden, was die Laufzeit erheblich optimiert. Der Aufbau der Funktion `lap_ext_theis[2d/3d]` wird in Programmcode 4

skizziert.

Programmcode 4: Ablauf der Funktion `lap_ext_theis[2d/3d]`

```
1 function lap_ext_theis[2d/3d](r, s, para)
    !Berechne Potenzreihen:
    alpha(n) = ...
    beta(n) = ...           !(Faà di Bruno)
5
    do [Schleife über s]
        !Berechne die Integrationsgrenzen:
        rmax = ...
        rmin = ...
10
        !Berechne Potenzreihen der Basislösungen:
        u1(n) = ...
        u2(n) = ...
15
        !Setze Basislösungen von rmin bis rmax fort:
        u1 = ...           !(Rosenbrock)
        u2 = ...           !(Rosenbrock)
20
        !Einarbeiten der Randwerte:
        A,B,C,D = ...
25
        !Auswertung an den Radien:
        if [r<rmax]: A*u1 + B*u2
        if [r>rmax]: C*I0 + D*K0 !(mod. Besselfunktionen)
        enddo
    end function lap_ext_theis[2d/3d]
```

Wenn man m Zeitpunkte t_i übergibt und die Stehfest-Grenze mit N gegeben ist, gibt es also $m \cdot N$ s -Punkte. Der rechenaufwendigste Schritt in `lap_ext_theis[2d/3d]` ist der Aufruf der Rosenbrock-Methode, welcher genau $2 \cdot m \cdot N$ mal erfolgt. Dennoch lohnt sich diese Berechnung, da man die Lösung an einem beliebigen Zeitpunkt auswerten kann, ohne die Lösung an vorherigen Zeitpunkten kennen zu müssen, wie es bei einer Finite-Differenzen-Methode der Fall wäre.

Die vollständige Implementierung dieser Funktionen ist im Anhang mit Programmcode 9, 10, 11 und 12 gegeben. Dabei sind nur die Varianten mit den Randbedingungen $r_{\text{well}} = 0$ und $r_{\infty} = \infty$ gegeben. Man vergleiche dazu Abschnitt 3.4.

Es ist noch die Stabilität dieser Implementierung zu testen. Dazu möchte ich dem letzten Abschnitt dieser Arbeit kommen.

4. Auswertung

Ich komme nun zur Auswertung meiner Betrachtungen und Implementierungen. Dazu möchte ich als Erstes eine Fehleranalyse der Umsetzung machen, da ich durch die variable Berechnung des Wertes r_{\max} aus Kapitel 3.5, ab welchem die Leitfähigkeit durch einen konstanten Wert abschneidet, um eine stabile Lösung für große Radien zu erhalten, einen gewissen Fehler entstehen lasse. Es gilt zu verifizieren, dass diese Wahl so gesetzt ist, dass sich die erzeugte Lösung nicht zu weit von der tatsächlichen Lösung entfernt.

Danach möchte ich als Hauptresultat verschiedene Plots für mehrere Eingabewerte zeigen und die Lösung mit der homogenen Theis-Lösung aus Abschnitt 2.3 vergleichen. Zum Abschluss ziehe ich ein Fazit und gebe einen Ausblick auf die Anwendungen der generierten Lösung, um dieser Arbeit einen perspektivischen Mehrwert zu verleihen.

4.1. Fehleranalyse

Um eine Fehleranalyse zu machen, ist es zuerst einmal wichtig, herauszufinden, welche Eingabe-Parameter von Interesse sind und in welchen Größenordnungen sie sich bewegen. Als Erstes betrachte ich die Parameter S und K_G . Wie sich gleich zeigt, spielen diese Größen für die Stabilität der Lösung eine eher untergeordnete Rolle. Um dies zu zeigen, betrachte ich noch einmal die anfängliche Differentialgleichung:

$$\nabla \left(K^{\text{CG}}(\|\underline{x}\|) \cdot \nabla h(\underline{x}, t) \right) = S \frac{\partial h}{\partial t}(\underline{x}, t) \quad (4.1.1)$$

$$\nabla \left(T_{\text{H}}^{\text{CG}}(\|\underline{x}\|) \cdot \nabla h(\underline{x}, t) \right) = S \frac{\partial h}{\partial t}(\underline{x}, t) \quad (4.1.2)$$

Nun kann man die Leitfähigkeit und die Durchlässigkeit jeweils von K_G und T_G trennen:

$$K^{\text{CG}}(r) = K_G \cdot \exp \left(\frac{\chi(\sigma^2, e)}{\sqrt{1 + (Cr)^2}^3} + \left(\frac{1}{2} - \gamma(e) \right) \sigma^2 \right) \quad (4.1.3)$$

$$=: K_G \cdot \tilde{K}^{\text{CG}}(r) \quad (4.1.4)$$

$$T_{\text{H}}^{\text{CG}}(r) = T_G \cdot \exp \left(\frac{-\frac{\sigma^2}{2}}{1 + (Cr)^2} \right) \quad (4.1.5)$$

$$=: T_G \cdot \tilde{T}_{\text{H}}^{\text{CG}}(r) \quad (4.1.6)$$

Damit kann man die Differentialgleichung wie folgt formulieren:

$$\nabla \left(\tilde{K}^{\text{CG}}(\|\underline{x}\|) \cdot \nabla h(\underline{x}, t) \right) = \frac{S}{K_G} \frac{\partial h}{\partial t}(\underline{x}, t) \quad (4.1.7)$$

$$\nabla \left(\tilde{T}_{\text{H}}^{\text{CG}}(\|\underline{x}\|) \cdot \nabla h(\underline{x}, t) \right) = \frac{S}{T_G} \frac{\partial h}{\partial t}(\underline{x}, t) \quad (4.1.8)$$

In der jetzigen Form bietet sich eine Variablentransformation an, um den Faktor $\frac{S}{K_G}$ respektive $\frac{S}{T_G}$ zu eliminieren. Es stehe im folgenden K für K_G respektive T_G :

$$\tilde{h}(\underline{x}, t) := h \left(\underline{x}, \frac{S}{K} \right) \quad (4.1.9)$$

$$\Rightarrow \frac{\partial \tilde{h}}{\partial t}(\underline{x}, t) = \frac{S}{K} \frac{\partial h}{\partial t}(\underline{x}, t) \quad (4.1.10)$$

Und somit erfüllt \tilde{h} folgende Differentialgleichung:

$$\nabla \left(\tilde{K}(\|\underline{x}\|) \cdot \nabla \tilde{h}(\underline{x}, t) \right) = \frac{\partial \tilde{h}}{\partial t}(\underline{x}, t) \quad (4.1.11)$$

Dabei stehe \tilde{K} für \tilde{K}^{CG} bzw. \tilde{T}_H^{CG} . Der Faktor $\frac{S}{K}$ staucht oder streckt also die Lösung nur in Zeitrichtung. Der Lösungsverlauf wird dadurch aber nicht essentiell beeinflusst. Somit verzichte ich darauf, diese Größen zu untersuchen. Des Weiteren gehen die Faktoren Q_w und L als Randbedingungen nur als skalierende Größen in die Gleichung ein. Da aber $\lambda \cdot \tilde{h}$ die selbe Differentialgleichung löst wie \tilde{h} , verzichte ich auch auf diese Untersuchung. Die essentiellen Größen der Differentialgleichung sind also σ^2 und ℓ . Die Anisotropierate e sowie die Anisotropiefunktion $\gamma(e)$ gehen jeweils nur als Faktoren einer dieser beiden Größen ein und spielen somit auch nur eine Nebenrolle. Anlehnend an [Zech, 2013] verwende ich folgende Standardwerte:

$$\begin{aligned}
K_G &= 1.0 \text{ E} - 4 \text{ ms}^{-1} \\
T_G &= 1.0 \text{ E} - 4 \text{ m}^2 \text{ s}^{-1} \\
S &= 1.0 \text{ E} - 4 \text{ m}^3 \text{ kg}^{-1} \\
Q_w &= -1.0 \text{ E} - 4 \text{ m}^3 \text{ s}^{-1} \\
L &= 1.0 \text{ m} \\
e &= 1.0
\end{aligned} \tag{4.1.12}$$

Es sind somit nur unterschiedliche Konstellationen von σ^2 und ℓ zu testen.

4.1.1. Analyse und Problematik der Frobenius-Methode

Ich konnte mit Hilfe der Frobenius-Methode ein implementierfreundliches Fundamentalsystem zur Lösung der Differentialgleichung geben. Die gesuchte Lösung ist in der Form

$$g_s(r) = \tilde{g}_s(\tau) = A_s u_{1,s}(\tau) + B_s u_{2,s}(\tau) \tag{4.1.13}$$

gegeben und erfüllt die Differentialgleichung wenigstens in einer Umgebung von 0, da nach dem Satz von Fuchs 2.2.2 ein Konvergenzradius von 1 gewährleistet wird. Ich nehme als Lösungsgebiet $\tau \in \left(0, \frac{1}{4}\right)$ respektive $r \in \left(0, \frac{1}{4} C^{-1}\right)$ an, um den Fehler, der durch die Taylorapproximation entsteht, gering zu halten. Um die Güte der Lösung zu analysieren, konstruiere ich für die Koeffizienten-Funktionen Vergleichsfunktionen, welche den essentiellen Verlauf repräsentieren und vergleiche diese mit deren Taylorapproximationen.

In 2D sehen diese wie folgt aus:

$$\begin{aligned}
\hat{\alpha}^{2D}(\tau) &= \frac{\tau^2}{(1 + \tau^2)^2} & \hat{\alpha}_n^{2D}(\tau) &= \sum_{k=0}^n \frac{\left(\hat{\alpha}^{2D}\right)^{(k)}(0)}{k!} \tau^k \\
\hat{\beta}^{2D}(\tau, \sigma^2) &= \tau^2 \exp\left(\frac{\sigma^2}{1 + \tau^2}\right) & \hat{\beta}_n^{2D}(\tau, \sigma^2) &= \sum_{k=0}^n \frac{\left(\hat{\beta}^{2D}\right)^{(k)}(0, \sigma^2)}{k!} \tau^k
\end{aligned} \tag{4.1.14}$$

In 3D unterscheide ich nun noch, ob am Brunnen K_H oder K_A angenommen wird mit einem Anisotropierate von $e = 1$:

$$\begin{aligned}
\hat{\alpha}^{3D}(\tau) &= \frac{\tau^2}{\sqrt{1 + \tau^2}} & \hat{\alpha}_n^{3D}(\tau) &= \sum_{k=0}^n \frac{\left(\hat{\alpha}^{3D}\right)^{(k)}(0)}{k!} \tau^k \\
\hat{\beta}^A(\tau, \sigma^2) &= \tau^2 \exp\left(\frac{-\frac{\sigma^2}{3}}{1 + \tau^2}\right) & \hat{\beta}_n^A(\tau, \sigma^2) &= \sum_{k=0}^n \frac{\left(\hat{\beta}^A\right)^{(k)}(0, \sigma^2)}{k!} \tau^k \\
\hat{\beta}^H(\tau, \sigma^2) &= \tau^2 \exp\left(\frac{\frac{2\sigma^2}{3}}{1 + \tau^2}\right) & \hat{\beta}_n^H(\tau, \sigma^2) &= \sum_{k=0}^n \frac{\left(\hat{\beta}^H\right)^{(k)}(0, \sigma^2)}{k!} \tau^k
\end{aligned} \tag{4.1.15}$$

Da ich als Standard Approximationsweite $n_{\max} = 40$ annehme, vergleiche ich nur diesen Wert. Für die Varianz nehme ich jeweils einen großen ($\sigma^2 = 6$) und einen kleinen ($\sigma^2 = \frac{1}{2}$) Vergleichswert. Man erhält mit doppelter Genauigkeit in $2D$:

$$\begin{aligned} \left| \hat{\alpha}^{2D} \left(\frac{1}{4} \right) - \hat{\alpha}_{40}^{2D} \left(\frac{1}{4} \right) \right| &= 0 \\ \left| \hat{\beta}^{2D} \left(\frac{1}{4}, \frac{1}{2} \right) - \hat{\beta}_{40}^{2D} \left(\frac{1}{4}, \frac{1}{2} \right) \right| &= 1.388 \text{ E} - 17 \\ \left| \hat{\beta}^{2D} \left(\frac{1}{4}, 6 \right) - \hat{\beta}_{40}^{2D} \left(\frac{1}{4}, 6 \right) \right| &= 2.220 \text{ E} - 16 \end{aligned}$$

Sowie in $3D$:

$$\begin{aligned} \left| \hat{\alpha}^{3D} \left(\frac{1}{4} \right) - \hat{\alpha}_{40}^{3D} \left(\frac{1}{4} \right) \right| &= 0 \\ \left| \hat{\beta}^A \left(\frac{1}{4}, \frac{1}{2} \right) - \hat{\beta}_{40}^A \left(\frac{1}{4}, \frac{1}{2} \right) \right| &= 0 \\ \left| \hat{\beta}^A \left(\frac{1}{4}, 6 \right) - \hat{\beta}_{40}^A \left(\frac{1}{4}, 6 \right) \right| &= 3.469 \text{ E} - 18 \\ \left| \hat{\beta}^H \left(\frac{1}{4}, \frac{1}{2} \right) - \hat{\beta}_{40}^H \left(\frac{1}{4}, \frac{1}{2} \right) \right| &= 0 \\ \left| \hat{\beta}^H \left(\frac{1}{4}, 6 \right) - \hat{\beta}_{40}^H \left(\frac{1}{4}, 6 \right) \right| &= 1.332 \text{ E} - 15 \end{aligned}$$

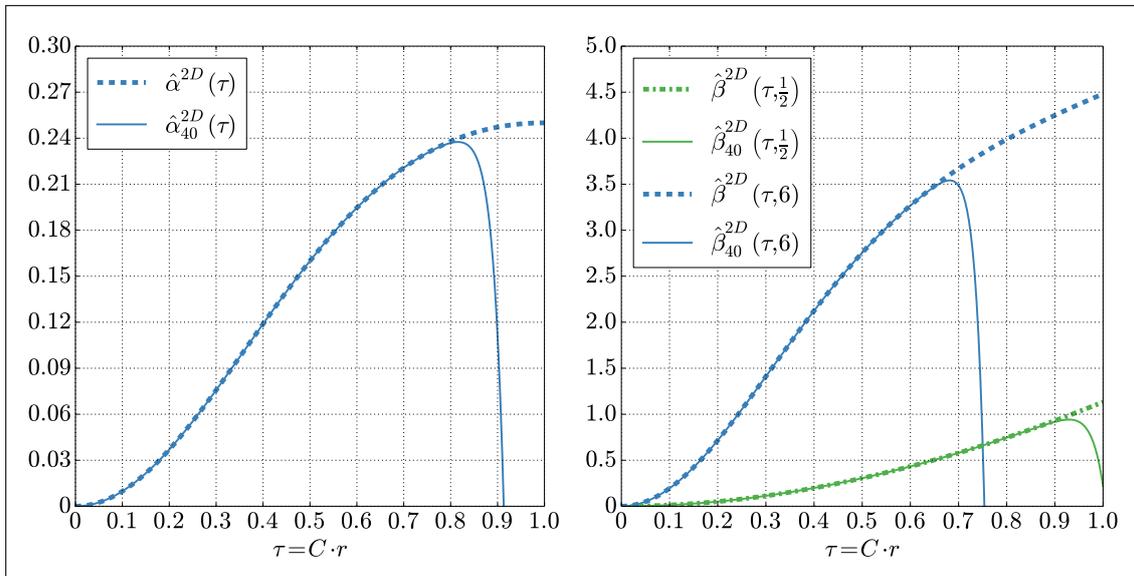


Abbildung 7: Vergleich des essentiellen Verlaufs der Koeffizientenfunktionen $\hat{\alpha}^{2D}$ (links) und $\hat{\beta}^{2D}$ (rechts) der Differentialgleichung für $2D$ im Laplaceraum mit den zugehörigen Taylor-Approximationen

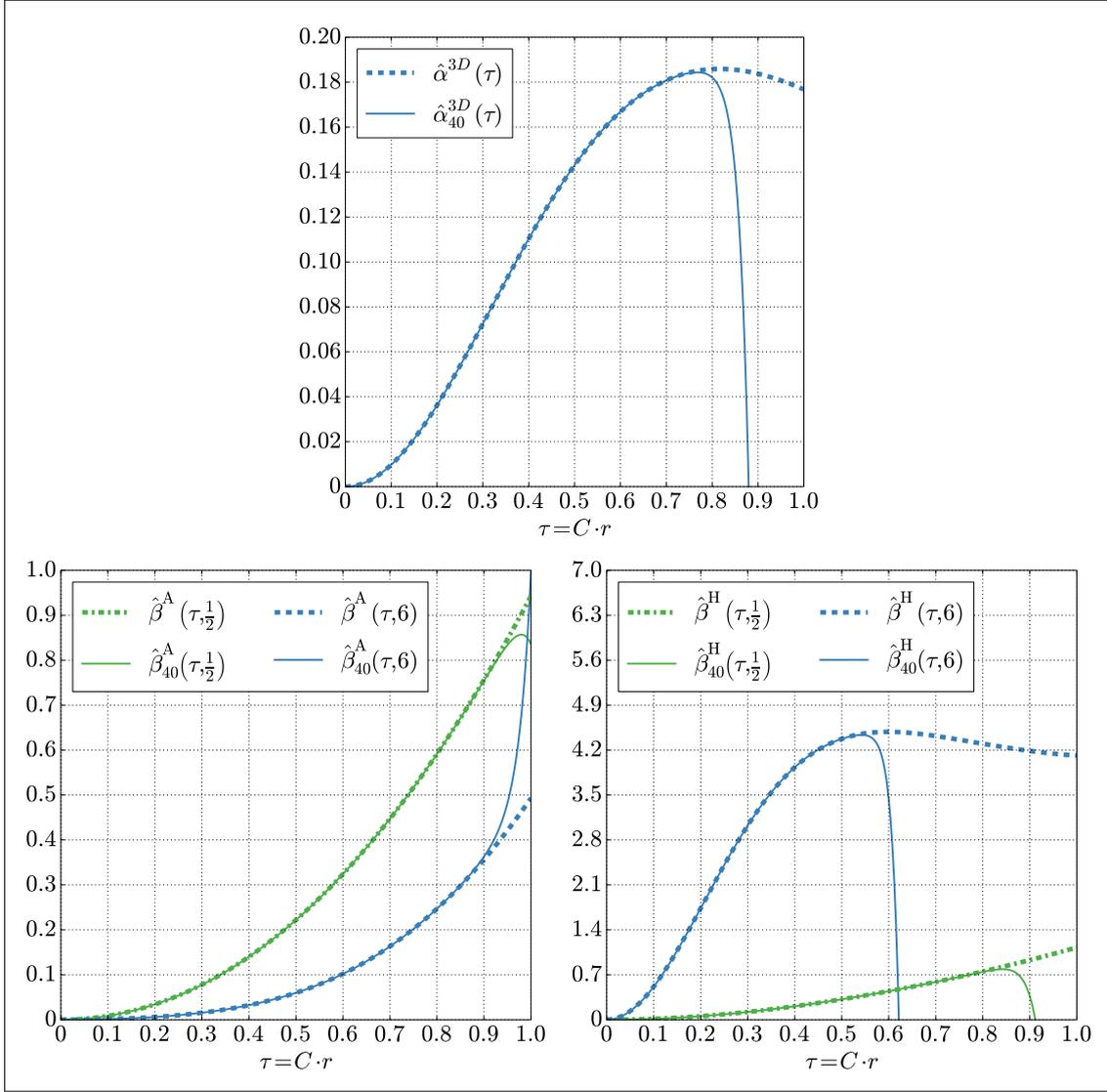


Abbildung 8: Vergleich des essentiellen Verlaufs der Koeffizientenfunktionen $\hat{\alpha}^{3D}$ (oben), $\hat{\beta}^A$ (unten links) und $\hat{\beta}^H$ (unten rechts) der Differentialgleichung für den 3D-Fall im Laplaceraum mit den zugehörigen Taylor-Approximationen

Somit sind die Approximationen im betrachteten Intervall sehr genau. Allerdings wird β mit wachsendem σ^2 schlechter approximiert. In den Abbildungen 7 und 8 sind die Taylorapproximationen der betrachteten Funktionen im Intervall $\tau \in [0, 1]$ dargestellt. Man sieht gut, dass der Konvergenzradius von 1 bedeutende Konsequenzen auf das Verhalten für $\tau \rightarrow 1$ hat. Um nun die Güte der Lösungen $u_{1,s}$ und $u_{2,s}$ zu untersuchen, vergleiche ich hier wenigstens für Standardparameter die Basislösungen in 3D mit den modifizierten Besselfunktionen im Intervall $\tau \in [0, 1]$. Als Standardparameter wähle ich die in Gleichung (4.1.12) gegebenen, wobei:

$$\sigma^2 = 1.0 \quad \ell = 10.0 \text{ m} \quad (4.1.16)$$

Ich betrachte nur den BCH-Fall $K_{\text{well}} = K_H$. Zum Vergleich nehme ich die beiden entstehenden Potenzreihen:

$$g_n(\tau, s) = \sum_{k=0}^n g_{1,s}^{(k)} \cdot \tau^k \quad h_n(\tau, s) = - \sum_{k=0}^n g_{2,s}^{(k)} \cdot \tau^k \quad (4.1.17)$$

Um diese Funktion mit den "richtigen" modifizierten Besselfunktion zu vergleichen, wähle ich wie in Gleichung (2.3.28) eine geeignete Substitution. Da $\frac{\alpha(\tau)}{\tau} \xrightarrow{\tau \rightarrow 0} 1$ und $\frac{\beta(\tau)}{\tau^2} \xrightarrow{\tau \rightarrow 0}$

$\frac{s \cdot S}{C^2 \cdot K_H}$ gilt, kann ich die Substitution (2.3.28) mit $T_s^2 = \frac{s \cdot S}{C^2 \cdot K_H}$ verwenden und erhalte damit in $\tau = 0$ dasselbe Anfangsverhalten der Basislösungen. Ich definiere also folgende Vergleichsfunktionen (vgl. Abschnitt 2.3.2):

$$V_1(\tau, s) = I_0\left(\frac{\sqrt{s}e^{\frac{1}{4}}}{0.16} \cdot \tau\right) \quad (4.1.18)$$

$$V_2(\tau, s) = K_0\left(\frac{\sqrt{s}e^{\frac{1}{4}}}{0.16} \cdot \tau\right) + \left(\ln\left(\frac{\sqrt{s}e^{\frac{1}{4}}}{0.32} \cdot \tau\right) + \gamma\right) \cdot I_0\left(\frac{\sqrt{s}e^{\frac{1}{4}}}{0.16} \cdot \tau\right)$$

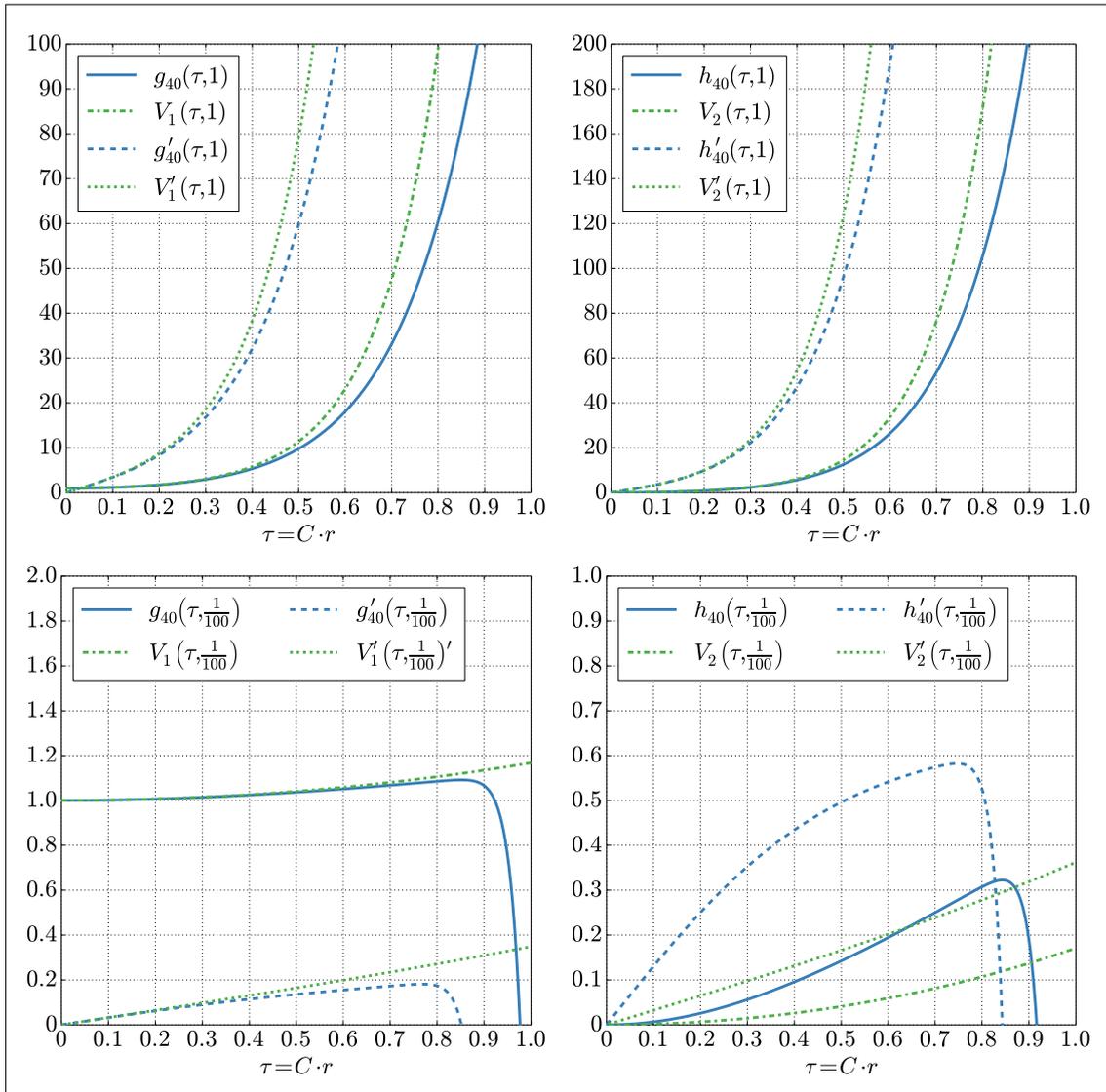


Abbildung 9: Vergleich der mit der Frobenius-Methode berechneten Basisfunktionen für den 3D-Fall mit den modifizierten Besselfunktionen, welche die homogenen Basislösungen darstellen. Man sieht die ersten Basisfunktion g und V_1 sowie deren Ableitungen für kleine Zeiten (oben links) und für große Zeiten (unten links). Weiter sieht man den analytischen Anteil der zweiten Basisfunktionen h und V_2 sowie ebenfalls deren Ableitungen für kleine Zeiten (oben rechts) und große Zeiten (unten rechts).

Da ich die Basisfunktion bis $n_{\max} = 40$ entwickelt habe, nehme ich diese obere Schranke auch für g_n und h_n . Da mit dem Stehfest-Algorithmus für wachsende Zeiten t die Funktion im Laplaceraum an immer kleineren Werte für s berechnet werden muss, werte ich die

Funktionen zum einen für $s = 1.0$ und zum anderen für $s = 1.0 \text{E} - 2$ aus. Die Vergleiche der Basisfunktionen findet man in Abbildung 9. Man sieht, dass die zusammengehörenden Basisfunktionen Funktionen jeweils in etwa den selben Verlauf haben, sich aber für wachsende τ voneinander entfernen, was auf die zugrunde liegende Heterogenität zurückzuführen ist. Man sieht außerdem gut, dass dieser Einfluss der Heterogenität für große Zeiten bei der zweiten Basisfunktion $h_n(\tau)$ am stärksten zu beobachten ist, da sie sich wesentlich von der homogenen Basisfunktion unterscheidet. Ich vertraue somit darauf, dass die approximierten Basisfunktionen wenigstens in $\tau \in \left(0, \frac{1}{4}\right)$ eine zuverlässige Näherung der tatsächlichen Basislösungen darstellen.

4.1.2. Einfluss des Abschneidewertes r_{\max}

In Kapitel 3.5 habe ich Abschätzungen gegeben, ab wann die Lösung der Grundwasser-
gleichung im Fernfeld durch die Theis-Lösung abzuschneiden ist. Dabei habe ich für die
Lösungen im 2D- und 3D-Modell folgende Formeln ermittelt:

1. Im 2D-Modell:

$$r_{\max}(t) = \min \left\{ \sqrt{\frac{4u_{\text{lim}} T_{\text{mid}} t}{S}}, r_{\text{lim}} \right\} \quad (4.1.19)$$

$$r_{\text{lim}}(\varepsilon_{\text{error}}) = \frac{1}{C} \cdot \sqrt{\frac{-\frac{\sigma^2}{2}}{\ln(1 - \varepsilon_{\text{error}})} - 1} \quad (4.1.20)$$

2. Im 3D-Modell:

$$r_{\max}(t) = \min \left\{ \sqrt{\frac{4u_{\text{lim}} K_{\text{mid}} t}{S}}, r_{\text{lim}} \right\} \quad (4.1.21)$$

$$r_{\text{lim}}(\varepsilon_{\text{error}}) = \frac{1}{C} \cdot \sqrt{\left(\frac{\chi}{\ln(1 \pm \varepsilon_{\text{error}})}\right)^{\frac{2}{3}} - 1} \quad (4.1.22)$$

Dabei ist im BCA-Fall “+” und im BCH-Fall “-” zu wählen.

In jedem Fall ist $r_{\max}(t)$ eine monoton wachsende Funktion von t , welche ab einem gewissen Zeitpunkt den konstanten Wert r_{lim} annimmt. Zur Untersuchung dieser Wahl ist es also angebracht, den Einfluss von $r_{\max}(t)$ für *kleine* und *große Zeiten* separat zu betrachten. Dafür setze ich folgende Raum- und Zeitintervalle fest:

$$\text{kleine Zeiten: } t \in T_1 := [10, 300] \quad (4.1.23)$$

$$\text{große Zeiten: } t \in T_2 := [300, 3600] \quad (4.1.24)$$

$$r \in R := [0.01, 50] \quad (4.1.25)$$

Ich betrachte also ein Zeitfenster von einer Stunde und trenne die Intervalle bei 5min. Die Grenze r_{\max} hängt dabei wesentlich von den Größen u_{lim} bei kleinen Zeiten und $\varepsilon_{\text{error}}$ bei großen Zeiten ab. Ich beschränke mich auf die Untersuchung von jeweils 2 Werten, welche die Größen annehmen können:

$$u_{\text{lim}} = \frac{1}{4}; 2 \quad (4.1.26)$$

$$\varepsilon_{\text{error}} = 1\%; 0.1\% \quad (4.1.27)$$

Den Einfluss dieser Größen möchte ich grafisch veranschaulichen. Dazu erstelle ich Grafiken nach folgendem Schema:

1. Funktionsverlauf der Lösung $h(r, t)$ mit $r \in [0.01, 50]$ und $t \in [10, 3600]$ für die Standardwerte $u_{\text{lim}} = \frac{1}{4}$ und $\varepsilon_{\text{error}} = 1\%$

2. Differenz $\Delta(r_{\text{lim}})$ der Lösungen für $\varepsilon_{\text{error}} = 1\%$ und $\varepsilon_{\text{error}} = 0.1\%$ bei großen Zeiten, also $t \in [300, 3600]$ mit $r_{\text{max}} \equiv r_{\text{lim}}$
3. Differenz $\Delta(u_{\text{lim}})$ der Lösungen für $u_{\text{lim}} = \frac{1}{4}$ und $u_{\text{lim}} = 2$ bei kleinen Zeiten, also $t \in [10, 300]$ ohne Abschneiden von $r_{\text{max}}(t)$ durch r_{lim}

Da die Funktionsverläufe der Leitfähigkeiten K_{H}^{CG} und K_{A}^{CG} bzw. der Durchlässigkeit T_{H}^{CG} jeweils unterschiedlich sind, betrachte ich diese drei Lösungstypen separat:

1. Lösung im 2D-Fall, also $K = T_{\text{H}}^{\text{CG}}(r)$
2. Lösung im 3D-Fall mit arithmetischer Mittlung am Brunnen (BCA-Fall), also $K = K_{\text{A}}^{\text{CG}}(r)$ und $K_{\text{well}} = K_{\text{A}}$
3. Lösung im 3D-Fall mit harmonischer Mittlung am Brunnen (BCH-Fall), also $K = K_{\text{H}}^{\text{CG}}(r)$ und $K_{\text{well}} = K_{\text{H}}$

Um zusätzlich den Einfluss der Bodenparameter σ^2 und ℓ mit einzubeziehen, werde ich diese Vorgehen auf die 4, in Gleichung (4.1.12) gegebenen, Standardparametersätze anwenden:

1. $\sigma^2 = 0.5$ und $\ell = 5$ m
2. $\sigma^2 = 0.5$ und $\ell = 10$ m
3. $\sigma^2 = 2$ und $\ell = 5$ m
4. $\sigma^2 = 2$ und $\ell = 10$ m

Die restlichen Parameter werden durch die in (4.1.12) genannten Standardwerte festgesetzt. Somit ergeben sich 12 zu untersuchende Konstellationen. Die Grafiken finden sich im Anhang in den Abbildungen 11, 12, 13, 14, 15 und 16.

In folgender Tabelle ist eine Übersicht der maximal entstandenen Fehler bei $\Delta(r_{\text{lim}})$ und $\Delta(u_{\text{lim}})$, sowie der konkreten Grenzen r_{lim} gegeben:

2D-Fall	$\sigma^2 = \frac{1}{2}, \ell = 5$	$\sigma^2 = \frac{1}{2}, \ell = 10$	$\sigma^2 = 2, \ell = 5$	$\sigma^2 = 2, \ell = 10$
$r_{\text{lim}} (\varepsilon_{\text{error}} = 1\%)$	15.27 m	30.54 m	31.01 m	62.03 m
$r_{\text{lim}} (\varepsilon_{\text{error}} = 0.1\%)$	49.30 m	98.60 m	98.75 m	197.49 m
$\max_{\substack{t \in T_2 \\ r \in \mathbb{R}}} \Delta(r_{\text{lim}}) $	2.045 E-3 m	1.073 E-3 m	1.102 E-3 m	3.086 E-4 m
$\max_{\substack{t \in T_1 \\ r \in \mathbb{R}}} \Delta(u_{\text{lim}}) $	5.051 E-3 m	6.968 E-3 m	3.396 E-2 m	4.329 E-2 m

3D-Fall (BCA)	$\sigma^2 = \frac{1}{2}, \ell = 5$	$\sigma^2 = \frac{1}{2}, \ell = 10$	$\sigma^2 = 2, \ell = 5$	$\sigma^2 = 2, \ell = 10$
$r_{\text{lim}} (\varepsilon_{\text{error}} = 1\%)$	14.72 m	29.44 m	24.60 m	49.21 m
$r_{\text{lim}} (\varepsilon_{\text{error}} = 0.1\%)$	33.83 m	67.65 m	54.25 m	108.50 m
$\max_{\substack{t \in T_2 \\ r \in \mathbb{R}}} \Delta(r_{\text{lim}}) $	2.034 E-3 m	1.138 E-3 m	1.200 E-3 m	5.636 E-4 m
$\max_{\substack{t \in T_1 \\ r \in \mathbb{R}}} \Delta(u_{\text{lim}}) $	2.882 E-3 m	3.246 E-3 m	5.123 E-3 m	6.796 E-2 m

3D-Fall (BCH)	$\sigma^2 = \frac{1}{2}, \ell = 5$	$\sigma^2 = \frac{1}{2}, \ell = 10$	$\sigma^2 = 2, \ell = 5$	$\sigma^2 = 2, \ell = 10$
$r_{\text{lim}} (\varepsilon_{\text{error}} = 1\%)$	9.54 m	19.08 m	15.63 m	31.26 m
$r_{\text{lim}} (\varepsilon_{\text{error}} = 0.1\%)$	21.44 m	42.87 m	34.25 m	68.49 m
$\max_{\substack{t \in T_2 \\ r \in \mathbb{R}}} \Delta(r_{\text{lim}}) $	2.677 E-3 m	1.738 E-3 m	1.706 E-3 m	9.801 E-4 m
$\max_{\substack{t \in T_1 \\ r \in \mathbb{R}}} \Delta(u_{\text{lim}}) $	5.016 E-3 m	8.617 E-3 m	2.773 E-2 m	4.258 E-2 m

Man sieht, dass der Einfluss von u_{lim} für wachsende σ^2 und ℓ immer größer wird. Allerdings werden die oben gegebenen Maxima immer für sehr kleine Zeiten und Radien angenommen. Dies liegt aber auch an numerischen Artefakten, welche durch die weite numerische Integration der Lösung entsteht. Für wachsende t und r fällt der Fehler dann sehr stark gegen 0 ab. Somit ist der Einfluss auf die Lösung insgesamt nicht gravierend, da man im Gegenzug eine wesentlich schnellere Berechnung erhält. Der Einfluss von r_{lim} nimmt mit wachsenden σ^2 und ℓ sogar ab. Insgesamt liegen die Fehler im Verhältnis zur Lösung aber in tolerierbaren Grenzen, da die Abweichungen im Bereich von einigen Millimetern bis wenigen Zentimetern liegen. Somit erachte ich es als hinnehmbar, die Fehler zuzulassen um im Gegenzug eine bessere Laufzeit zu erhalten, da man den Bereich der numerischen Integration klein halten kann. Im Bereich von wenigen Sekunden ist die Lösung im Verhältnis zur Theis-Lösung ohnehin noch nicht aussagekräftig.

4.2. Plots und Vergleich mit der Theis-Lösung

Zum Schluss und als Ergebnis dieser Arbeit möchte ich jetzt einige Grafiken geben, welche die Lösungen für Standardparametersätze zeigen und den Einfluss einzelner Parameter verdeutlichen. Dabei orientiere ich mich wieder an der Arbeit in [Zech, 2013]. Des Weiteren möchte ich die Unterschiede der erarbeiteten Lösung zur Theis-Lösung für homogene Wasserleiter zeigen und einige Vergleiche anstellen. Als *Standardparametersatz* wähle ich die in Gleichung (4.1.12) gegebenen Parameter und setze dabei zusätzlich σ^2 und ℓ fest:

$$\begin{aligned}\sigma^2 &= 1.0 \\ \ell &= 10.0 \text{ m}\end{aligned}\tag{4.2.1}$$

4.2.1. Plots der erweiterten Theis-Lösung

In Abbildung 10 sind die Lösungen der Grundwassergleichung für den 2D- und 3D-Fall, wobei noch zwischen den Randbedingungen BCA und BCH unterschieden wird, auf einer gemeinsamen Skala gezeigt.

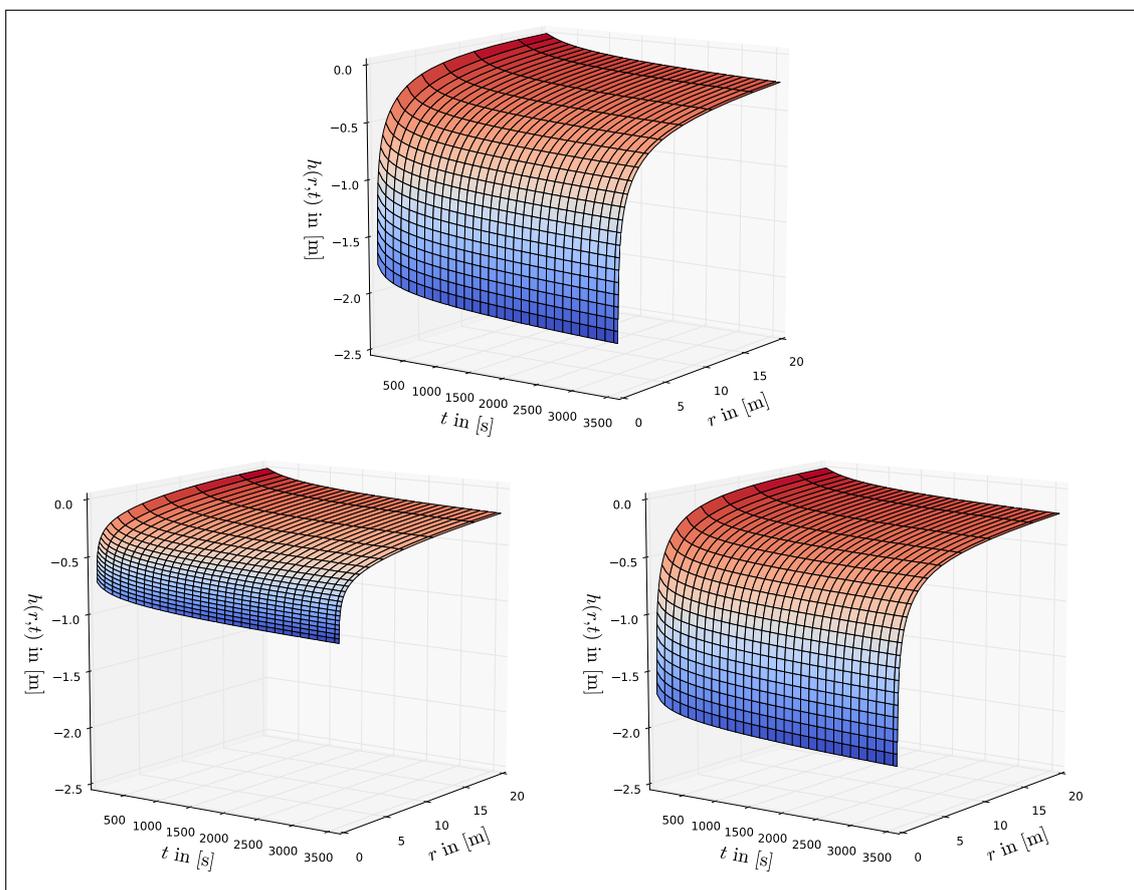


Abbildung 10: Plots der Lösungen für den Standardparametersatz im 2D-Fall (oben) und im 3D-Fall (unten) wobei zum einen $K_{\text{well}} = K_A$ (links) und zum anderen $K_{\text{well}} = K_H$ (rechts) gewählt wurde

Dabei ist $t \in [60, 3600]$ und $r \in [0.01, 20]$. Es entstehen dabei folgende Randwerte der

Coarse-Graining-Leitfähigkeit respektive -Durchlässigkeit:

$$\begin{aligned}
T_{\text{H}}^{\text{CG}}(r=0) &= T_{\text{H}} &= 6.0653065971263348 \text{ E}-5 \text{ m}^2\text{s}^{-1} & (4.2.2) \\
T_{\text{H}}^{\text{CG}}(r \rightarrow \infty) &= T_{\text{G}} &= 1.0 \text{ E}-4 \text{ m}^2\text{s}^{-1} \\
K_{\text{A}}^{\text{CG}}(r=0) &= K_{\text{A}} &= 1.6487212707001284 \text{ E}-4 \text{ ms}^{-1} \\
K_{\text{H}}^{\text{CG}}(r=0) &= K_{\text{H}} &= 6.0653065971263348 \text{ E}-5 \text{ ms}^{-1} \\
K^{\text{CG}}(r \rightarrow \infty) &= K_{\text{efu}} &= 1.1813604128656462 \text{ E}-4 \text{ ms}^{-1}
\end{aligned}$$

Man erkennt gut, dass die Wahl der Anfangsbedingung im $3D$ -Fall erheblichen Einfluss auf den Lösungsverlauf hat, da K_{H} wesentlich größer ist als K_{A} .

In den Abbildungen 17 und 18 im Anhang wird der Einfluss der Anisotropierate e auf die Lösung im $3D$ -Fall mit BCA- sowie BCH-Randbedingung gezeigt. Dabei wurde zum einen $e = 1$ und zum anderen $e = \frac{1}{4}$ gewählt. Für $e = \frac{1}{4}$ entsteht folgender Wert für K_{efu} der Coarse-Graining-Leitfähigkeit:

$$K^{\text{CG}}(r \rightarrow \infty) = K_{\text{efu}} = 1.4216536599734490 \text{ E}-4 \text{ ms}^{-1} \quad (4.2.3)$$

Man erkennt, dass die Anisotropierate einen eher geringen Einfluss auf die Lösung hat, der aber die größten Auswirkungen in Brunnennähe hat und mit wachsender Zeit größer wird.

Da die Varianz σ^2 und die Korrelationslänge ℓ in alle drei Lösungstypen im selben Maße eingehen, werde ich deren Einfluss nur im $2D$ -Modell zeigen. In Abbildung 19 im Anhang wird der Einfluss der Varianz σ^2 auf die Lösung im $2D$ -Fall gezeigt. Dabei wurde zum einen $\sigma^2 = 1$ und zum anderen $\sigma^2 = 2$ gewählt. Für $\sigma^2 = 2$ entsteht folgender Wert T_{H} der Coarse-Graining-Durchlässigkeit:

$$T_{\text{H}}^{\text{CG}}(r=0) = T_{\text{H}} = 3.6787944117144237 \text{ E}-5 \text{ m}^2\text{s}^{-1} \quad (4.2.4)$$

Dieser Wert ist nur etwa halb so groß wie im Fall $\sigma^2 = 1$. Der Einfluss von σ^2 ist in Brunnennähe erheblich und bleibt über die Zeit fast konstant. Mit wachsendem Abstand zum Brunnen konvergiert der Unterschied zwischen den Lösungen allerdings sehr schnell gegen 0.

In Abbildung 20 im Anhang wird letztlich noch der Einfluss der Korrelationslänge ℓ auf die Lösung im $2D$ -Fall gezeigt. Dabei wurde zum einen wieder der Standardwert $\ell = 10$ und zum anderen der Wert $\ell = 20$ gewählt. Durch eine unterschiedliche Wahl von ℓ ändert sich der Wertebereich der Coarse-Graining-Durchlässigkeit respektive -Leitfähigkeit nicht. Die Verteilungen werden lediglich radial gestreckt oder gestaucht. Damit bleiben die Randwerte der Verteilungen unberührt. Den stärksten Einfluss hat auch ℓ wieder in Brunnennähe und für wachsende Radien fällt die Differenz zwischen den Lösungen schnell ab. Für große Zeiten bleibt die Differenz dann nahezu konstant.

4.2.2. Vergleich zur Theis-Lösung

Da es bei der erweiterten Theis-Lösung darum geht, heterogene Leitfähigkeitsverteilungen zu berücksichtigen, ist es von Interesse, zu untersuchen, inwiefern sich diese Lösung von der homogenen Theis-Lösung unterscheidet. Dazu ist es natürlich wichtig, welchen Leitfähigkeitswert respektive Durchlässigkeitswert man in die Theis-Lösung gibt. Im $2D$ -Fall sind natürlich die beiden Randwerte der Durchlässigkeitsverteilung T_{G} und T_{H} interessant. Im $3D$ -Fall kommt hinzu, dass das geometrische Mittel der Leitfähigkeit K_{G} nicht als Randwert der Leitfähigkeitsverteilung auftritt. Hier sind also die drei Werte K_{well} , welcher entweder K_{A} im BCA-Fall oder K_{H} im BCH-Fall ist, K_{efu} und K_{G} von Interesse.

4.2.2.1. 2D-Lösung: In Abbildung 21 im Anhang ist die Theis-Lösung für $T = T_G$ und $T = T_H$, wie in den Gleichungen (4.2.1) und (4.2.2) gegeben, gezeigt und deren Differenz zur 2D-Lösung für Standardwerte. Man sieht dabei, dass für $T = T_{\text{well}} = T_H$ die Differenz in Brunnennähe am kleinsten ist und mit wachsendem Radius zunimmt. Für $T = T_G$ ist die Differenz in Brunnennähe erheblich, nimmt aber mit wachsendem Abstand zum Brunnen sehr schnell ab. Die erweiterte Theis-Lösung interpoliert also grob die Theis-Lösungen für Nah- und Fernfeld-Leitwerte.

4.2.2.2. 3D-Lösung im BCA-Fall: In Abbildung 22 im Anhang ist die Theis-Lösung für $K = K_A$, $K = K_{\text{efu}}$ und $K = K_G$, wie in den Gleichungen (4.2.1) und (4.2.2) gegeben, gezeigt sowie deren Differenz zur 3D-Lösung im BCA-Fall für die Standardwerte. Man sieht auch hier, dass für $K = K_{\text{well}} = K_A$ die Differenz in Brunnennähe gering ausfällt und für wachsende Radien größer wird. Das selbe Szenario für den Fernfeldwert wie in der 2D-Lösung ist auch hier zu beobachten. Für $K = K_{\text{efu}}$ ist die Differenz der Theis-Lösung zur 3D-Lösung in Brunnennähe sehr groß, nimmt aber mit wachsendem Abstand zum Brunnen schnell ab. Somit ist hier die Übereinstimmung der Lösungen im Fernfeld wie zu erwarten gegeben. Für $K = K_G$ ist die Differenz in Brunnennähe nicht ganz so groß wie bei $K = K_{\text{efu}}$, aber immer noch erheblich. Der Fehler nimmt mit wachsenden Radien zwar ab, aber auch nicht so stark wie bei K_{efu} . Insgesamt weicht die 3D-Lösung im BCA-Fall weniger von der Theis-Lösung ab, als im BCH-Fall. Das liegt daran, dass die Werte K_A und K_{efu} wesentlich näher aneinander liegen, als K_H und K_{efu} . Somit ist die Lösung im BCA-Fall wesentlich homogener als im BCH-Fall und somit auch wesentlich näher an der Theis-Lösung. Auch hier interpoliert die vorliegende Lösung die Theis-Lösungen für Nah- und Fernfeldleitwerte.

4.2.2.3. 3D-Lösung im BCH-Fall: Letztlich ist in Abbildung 23 im Anhang die Theis-Lösung für $K = K_H$, $K = K_{\text{efu}}$ und $K = K_G$, wie in den Gleichungen (4.2.1) und (4.2.2) gegeben, gezeigt und ebenfalls deren Differenz zur 3D-Lösung im BCH-Fall für die Standardwerte. Man sieht auch hier, dass für $K = K_{\text{well}} = K_H$ die Differenz in Brunnennähe gering ausfällt und für wachsende Radien größer wird. Für $K = K_{\text{efu}}$ ist die Differenz der in Brunnennähe sehr groß, nimmt aber mit wachsendem Abstand zum Brunnen schnell ab. Somit ist auch hier die Übereinstimmung der Lösungen im Fernfeld gegeben. Für $K = K_G$ ist die Differenz in Brunnennähe nicht ganz so groß wie bei $K = K_{\text{efu}}$, aber immer noch erheblich. Der Fehler wird mit wachsende Radien kleiner, geht aber nicht so schnell gegen 0 wie für $K = K_{\text{efu}}$. Wie schon im BCA-Fall interpoliert die vorliegende Lösung die Theis-Lösungen für Nah- und Fernfeldleitwerte und hat somit den erwünschten Verlauf.

4.3. Fazit und Ausblick

Die implementierten Lösungen stellen stabile Routinen zur Auswertung der Lösung der Grundwassergleichung im Coarse-Graining-Modell bereit. Die Grundwassergleichung (1.1.3) ist an sich eine parabolische Differentialgleichung, zu der man im allgemeinen heterogenen Fall keine symbolische Grundlösung geben kann. In der Anwendung ist man also auf numerische Methoden angewiesen. Das Ziel der Coarse-Graining-Methode war es, die hydraulischen Gegebenheit so zu vereinfachen, dass das entstehende Modell leichter zu berechnen ist, was heißt, dass die Implementierung einer Lösung schneller und effizienter ist, als bei der Anwendung von numerischen Standardlösmethoden für partielle Differentialgleichungen. Dazu traf man die Annahme, dass durch eine effektive Leitfähigkeit, welche nur vom radialen Abstand zum Brunnen abhängt, auch eine einfacher zu berechnende Modellfunktion entsteht. Es folgte die radialsymmetrische Grundwassergleichung (1.2.19) mit lediglich einer Raumvariablen und ausschließlich zeitunabhängigen Koeffizienten und Randwerten. Es bot sich also an, diese Struktur auszunutzen, um die Differentialgleichung noch weiter zu reduzieren, damit man auf eine gewöhnliche Differentialgleichung kommt.

Die Laplacetransformation ist, ebenso wie andere Integraltransformationen, zur Lösung von partiellen und gewöhnlichen Differentialgleichungen deshalb von Interesse, weil mit ihr Ableitungen im Ursprungsraum in algebraische Operationen im Laplaceraum überführt werden können. Es ist also möglich, die Struktur von Differentialgleichungen so zu algebraisieren. Da die Zeitableitung der Differentialgleichung (1.2.19) lediglich linear eingeht, konnte man mit dem Differentiationssatz (2.1.14) der Laplacetransformation die radialsymmetrische Grundwassergleichung auf eine gewöhnliche Differentialgleichung (2.2.1) mit einem Parameter s im Laplaceraum transformieren.

Nach dieser Vorarbeit war es nun möglich den Apparat zur Lösung gewöhnlicher Differentialgleichungen zu nutzen. Es entstand eine lineare Differentialgleichung zweiter Ordnung mit nicht konstanten Koeffizienten. Also gab es auch hier keine symbolischen Lösungen der Differentialgleichung. Hier war die Frobeniusmethode bestens geeignet, um der singulären Struktur der Differentialgleichung gerecht zu werden. Die Lösung nach Frobenius hatte zwei bedeutende Vorteile. Zum einen wurde die Lösung durch eine Potenzreihe konstruiert, was mit wenig Programmieraufwand umzusetzen war. Zum anderen konnte man durch das logarithmische Verhalten der zweiten Basislösung (2.2.35) die Randbedingung (1.2.24) bzw. (1.2.25) kanonisch einpflegen. Diese elegante Einarbeitung der Randbedingung konnte man schon bei der Theis-Lösung in Kapitel 2.3 beobachten. Somit war davon auszugehen, dass die Lösung im Coarse-Graining-Modell ein ähnliches Verhalten zeigen wird. Der Nachteil der Frobeniusmethode ist, dass die entstehenden Potenzreihen nur einen begrenzten Konvergenzradius haben. Somit konnte ich mit dieser Methode weder die Randwerte im Fernfeld einarbeiten, noch die Lösung für große Radien auswerten.

Da die effektive Leitfähigkeit eine Übergangsfunktion zwischen einem Nah- und einem Fernfeldwert darstellt, konnte ich die Leitfähigkeit für große Radien als konstant annehmen. Es war folglich möglich eine stabile Lösung der Differentialgleichung im Fernfeld zu erhalten, da die Basislösungen für homogene Wasserleiter die gut untersuchten Besselfunktionen sind, welche mit stabilen Implementierungen vorliegen. Diese waren wiederum gut geeignet, die Randbedingungen im Fernfeld einzuarbeiten, wie es schon bei der Theis-Lösung der Fall war.

Letztlich musste ich nur noch beide Bereiche entsprechend der Differentialgleichung miteinander verbinden. Da die Frobeniusmethode zwei linear unabhängige Basislösungen generiert hatte, konnte ich diese mit dem Rosenbrock-Verfahren bis zu einem maximal benötigten Radius fortsetzen. Da die Wahl dieses Radius dadurch bestimmt wird, ab wann die effektive Leitfähigkeit nahezu konstant ist, musste ich noch sicherstellen, dass die Lösung stabil bleibt. Für kleine Zeiten hat die Lösung die Eigenschaft, sehr schnell gegen 0 abzufallen, was durch meine Analyse der Theis-Lösung sichergestellt wurde. Damit konnte

ich den Algorithmus für kleine Zeiten durch einen Vergleich mit der Lösung für homogene Wasserleiter optimieren.

Um die Lösung aus dem Laplaceraum wieder zurück zu transformieren, wurde der Stehfest-Algorithmus gewählt, welcher lediglich eine Auswertung der Funktion im Laplaceraum an reellen Werten erfordert. Da der Algorithmus für nicht periodische Funktionen, zum Beispiel für die Brunnenfunktion, gute Ergebnisse liefert, stellt er eine zuverlässige Wahl für die betrachteten Funktionen dar.

Dass das Abschneiden der Funktionen im Fernfeld zulässig ist, wurde von mir analysiert und es zeigte sich, dass ab einer relativen Abweichung von unter 1% das gewählte Vorgehen keine erheblichen Fehler generiert.

Die genutzte Methode hat den Vorteil, dass die Auswertung an beliebigen Zeitpunkten erfolgen kann. Würde man die Lösung numerisch durch eine Finite-Differenzen-Methode berechnen, wäre man, der Natur von parabolischen Differentialgleichungen nach, auf eine Schrittverhältnis von

$$\frac{\max K(x) \cdot \Delta t}{S(\Delta x)^2} \leq \frac{1}{2} \quad (4.3.1)$$

angewiesen (vgl. [Großmann, 2005]). Zur Lösung wäre dieser Ansatz also nicht geeignet, da hier von einem unbegrenzten Wasserleiter ausgegangen wird. Andere numerische Approximationen wie die Linienmethode hätten zur Folge, dass man ebenfalls auf ein steifes Anfangswertproblem in Zeitrichtung kommen würde (vgl. [Großmann, 2005]). Somit ist die verwendete Methode für die gegebenen Randwerte optimal gewählt.

Die Lösung kann nun dazu verwendet werden, reale Bodenparameter zu schätzen. Dazu ist es zuerst von Nöten, zu zeigen, dass die generierte Lösung wirklich eine qualitative Aussage über Böden zu den gegebenen Parametern μ , σ^2 und ℓ machen kann. Dazu kann man wie in der Arbeit von [Zech, 2013] für den stationären Fall vorgehen. Man generiert mit einem Feldgenerator synthetische Böden zu festen Parametern und lässt auf diesen Böden die Grundwassergleichung mit einem beliebigen numerischen Verfahren lösen. Dann bildet man über ein ganzes Ensemble von Böden zu gleichen Parametern einen mittleren Absenktrichter, indem man über die entstehenden Lösungen mittelt. Diesen mittleren Absenktrichter muss man dann mit der hier berechneten erweiterten Theis-Lösung vergleichen, in der Hoffnung, dass diese im Wesentlichen übereinstimmen. Da dies schon im stationären Fall gezeigt werden konnte, ist davon auszugehen, dass die erweiterte Theis-Lösung wirklich einen effektiven Fluss beschreibt.

Hat man dies verifiziert, kann man die Lösung nutzen, um für reale Böden Parameter zu bestimmen. Man nutzt also die generierte Lösung um sie an gemessene Zeitreihen zu fitten, um damit die heterogene Struktur eines Wasserleiters zu charakterisieren. Algorithmen zum inversen Schätzen sind darauf angewiesen, dass die zu fittende Funktion schnell aufgerufen und berechnet werden kann. Da die entwickelten Routinen genau das leisten, ist mit dieser Arbeit eine gute Basis für die zukünftige Forschung gegeben.

A. Anhang

A.1. Algorithmen

Programmcode 5: Implementierung der Funktion `nextpart` welche zu einer Partition die nächst kleinere ausgibt

```
1 function nextpart(part)
    implicit none
    integer(i4), dimension(:), intent(in) :: part
    integer(i4), dimension(size(part)) :: nextpart
    integer(i4) :: n, firstindex, summ, temp

    !check if input is valid
    if (.not. isgoodpart(part)) stop 'not a valid input partition.'
    if (part(1) .eq. size(part)) stop 'was the last partition.'

    nextpart = part

    !set the first entry to zero and remember it
    temp = nextpart(1)
    nextpart(1) = 0

    !search for the first index that is not equal to zero
    firstindex = 1
    do n = 2, size(nextpart)
        if (nextpart(n) .ne. 0) then
            firstindex = n
            exit
        endif
    end do

    !decrease the first entry that is not equal to zero by one
    nextpart(firstindex) = nextpart(firstindex)-1

    !calculate the actual sum of the partition
    summ = size(nextpart)-firstindex-temp

    !fill up the partition below the index from above
    !until the sum equals n again
    do n=firstindex-1,1,-1
        temp = nextpart(n)
        !set the next entry as big as possible; terminates for n=1
        nextpart(n) = floor(real(size(nextpart)-summ,sp)/real(n,sp))
        !calculate the actual sum of the partition
        summ = summ + n*(nextpart(n) - temp)
        !if the sum is n we are done
        if (summ == size(nextpart)) exit
    end do

end function nextpart
```

Programmcode 6: Implementierung der Funktion `isgoodpart` welche überprüft, ob eine Partition vorliegt

```
1  function isgoodpart(part)
    implicit none
    integer(i4), dimension(:), intent(in) :: part
    logical :: isgoodpart
    integer(i4) :: n, testn
10  isgoodpart = .false.
    testn = 0
    if (size(part) .le. 1) stop 'n_must_be_at_least_2'
15  !check if p_i >= 0
    if (any(part<0)) return
    !check if it is a partition: sum(i*p_i)=n
    do n=size(part),1,-1
20      testn = testn+n*part(n)
        if (testn > size(part)) return
    end do
    if (testn==size(part)) isgoodpart = .true.
25 end function isgoodpart
```

Programmcode 7: Implementierung des Stehfest-Algorithmus NLinVSteh welcher zu einer gegebenen Funktion func die Laplaceinvertierte berechnet

```

1  function nlinvsteh( func, para, r, t, n)
   !interface for the given function with additional radial-values r
   interface
     function func(r, s, para)
5      use mo_kind, only: dp
       implicit none
       real(dp), dimension(:), intent(in)    :: r
       real(dp), dimension(:), intent(in)    :: s
10      real(dp), dimension(:), intent(in)    :: para
       real(dp), dimension(size(r),size(s))  :: func
     end function func
   end interface
   real(dp),    dimension(:), intent(in)    :: para
   real(dp),    dimension(:), intent(in)    :: r
15  real(dp),    dimension(:), intent(in)    :: t
   integer(i4), intent(in)                 :: n
   real(dp),    dimension(size(r),size(t))  :: nlinvsteh
   !intern variables
   integer(i4)&
20   :: m,i,j
   real(dp), dimension(size(t))&
   :: a
   real(dp), dimension(2*ceiling(real(n,dp)/2.0))&
   :: csteh
25  real(dp), dimension(size(t), 2*ceiling(real(n,dp)/2.0))&
   :: timepoints
   real(dp), dimension(size(r), size(t), 2*ceiling(real(n,dp)/2.0))&
   :: funcpoints

30  !check if the given boundary is a positiv number
   if (n<=0) stop 'the_stehfest-boundary_n_must_be_positiv.'
   !check if there are time-values and if they are valid
   if (size(t)==0) stop 'there_are_no_given_time-values.'
   !check if there are space-values and if they are valid
35  if (size(r)==0) stop 'there_are_no_given_space-values.'

   !if n is odd take the next bigger even number n=2m
   m = ceiling(real(n,dp)/2.0)
   !define the coefficients c_n
40  csteh = csteh(m)
   !evaluate all s-points to call the function only one time
   !set all necessary s-values into a matrix
   do i=1, 2*m
     timepoints(:,i) = real(i,dp)/t*log(2.0)
45  end do
   !get the function-values for all needed s-values
   funcpoints
     = reshape( func(r,pack(timepoints,.true.), para),&
                (/size(r),size(t),2*m/) )
50  !calculate the lapalce-inversion
   a = log(2.0)/t
   do i = 1, size(r)
     do j = 1, size(t)
       nlinvsteh(i,j) &
55      = dot_product(csteh, funcpoints(i,j,:))
     end do
     nlinvsteh(i,:) &
       = a*nlinvsteh(i,:)
   end do
60  end function nlinvsteh

```

Programmcode 8: Implementierung der Routine `csteh` welche die Koeffizienten c_n für den Stehfest-Algorithmus berechnet

```

1  function csteh(m)

    implicit none

5     integer(i4), intent(in)  :: m
    real(dp), dimension(2*m) :: csteh

    !local variables
10    integer(i4)                :: k, n

    !check if the given boundary is not to big
    if (m>=11) stop 'the_stehfest-boundary_must_be_less_than_22'

    csteh = 0.0

15    !calculate the coefficients
    do n=1, 2*m
        do k=floor((real(n,dp)+1.0)/2.0), min(n,m)
            csteh(n) =  csteh(n)
20                + real(k,dp)**real(m+1,dp)&
                    * real(binomcoeffi(int(2*k,i8),int(k,i8)),dp)&
                    / real(factorial(int(m-k,i8))&
                    * factorial(int(n-k,i8))*factorial(int(2*k-n,i8)),dp)

            end do
25    csteh(n) = (-1.0)**(n+m)*csteh(n)
    end do

end function csteh

```

Programmcode 9: Implementierung der Funktion `ext_theis2D` welche die erweiterte Theis-Lösung im 2D-Fall berechnet

```

1  function ext_theis2d(rad,time,params,inits,prop,grad,steh_n,output)

      implicit none

5     !radius
      real(dp), dimension(:),          intent(in) :: rad
      !time
      real(dp), dimension(:),          intent(in) :: time
      !params=(tg,sigma**2,corr,s)
10    real(dp), dimension(4),          intent(in) :: params
      !inits=(qw)
      real(dp), dimension(1),          intent(in) :: inits
      !proportionality-factor [def=2]
      real(dp), optional, intent(in) :: prop
15    !limit of the series expansion [def=40]
      integer(i4), optional, intent(in) :: grad
      !the boundary for the stehfest-alg. [def=6]
      integer(i4), optional, intent(in) :: steh_n
      !enables output while computation
20    logical, optional, intent(in) :: output
      real(dp), dimension(size(rad),size(time)) :: ext_theis2d
      real(dp) :: pr,output_real,c
      real(dp), dimension(8) :: values
      integer(i4) :: stehlim,limit,n
25    logical, dimension(size(rad), size(time)) :: mask

      ! -----
      ! input tests
      ! -----

30    if (.not. all(rad>0.0)) &
          stop 'radius_must_be_positive.'
      if (any(time < 0.0)) &
          stop 'the_time_must_be_non-negative.'
35    if (ge(inits(1), 0.0)) &
          stop 'the_pumping-rate_must_be_negative.'
      if (le(params(1), 0.0)) &
          stop 'the_transmissivity_must_be_positive.'
40    if (le(params(2), 0.0)) &
          stop 'the_variance_must_be_positive.'
      if (le(params(3), 0.0)) &
          stop 'the_correlation_length_must_be_positive.'
      if (le(params(4), 0.0)) &
          stop 'the_storage_must_be_positive.'
45

      if (present(output)) then
          if (output) then
              output_real = 1.0
          else
50             output_real = 0.0
          endif
      else
          output_real = 0.0
      endif

55

      if (present(prop)) then
          if (le(prop, 0.0)) &
              stop 'the_proportionality_factor_must_be_positive.'
          pr = prop
60      else
          pr = 1.6
      endif
  
```

```

65   if (present(steh_n)) then
        if (steh_n<=0)           &
            stop 'the stehfest-boundary must be positiv.'
        stehlim      =  steh_n
    else
        stehlim      =  6
70   endif

    if (present(grad)) then
        if (grad<=3)           &
            stop 'the limit of the expansion must be at least 4.'
75        limit       =  grad
    else
        limit        =  40
    endif

80   ! -----
    ! setting of the intern variables
    ! -----

    c          =  pr/params(3)
85   values     =  (/  params(1),&
                    params(2),&
                    c,&
                    params(4),&
                    inits(1),&
90             real(limit,dp),&
                    real(stehlim,dp),&
                    output_real/)

    !the optional output
95   if (nint(output_real,i4)==1) then
        write (*,*) ""
        write (*,*) "tg: ",params(1)
        write (*,*) "sig2: ",params(2)
        write (*,*) "corr: ",params(3)
        write (*,*) "s: ",params(4)
100        write (*,*) "qw: ",inits(1)
        write (*,*) "prop: ", pr
    end if

105   ! -----
    ! calculate the head
    ! -----

    do n=1, size(rad)
        mask(n,:) = (time>0.0)
110    end do

    ext_theis2d = 0.0

115    ext_theis2d =&
        unpack(&
            pack(nlinvsteh(lap_ext_theis2d,&
                values,&
                c*rad,&
                pack(time, (time>0.0)),&
                stehlim),&
120            .true.),&
            mask,&
            ext_theis2d)

125 end function ext_theis2d

```

Programmcode 10: Implementierung der Funktion lap_ext_theis2D welche die Lösung der erweiterten Theis-Lösung für den 2D-Fall im Laplaceraum berechnet

```

1  function lap_ext_theis2d(r, s, para)

    implicit none

5   real(dp), dimension(:),          intent(in) :: r
    real(dp), dimension(:),          intent(in) :: s
    !para=(tg,var^2,c,s,ref,qw,tref,limit, stehlim)
    real(dp), dimension(:),          intent(in) :: para

10  real(dp), dimension(size(r), size(s))      :: lap_ext_theis2d

    !the standard limit for the series-expansion
    integer(i4), parameter              :: stdsiz=41
    !series-expansion of the given functions
15  real(dp), dimension(max(nint(para(6),i4)+1,stdsiz)) &
        :: dg, dh, g, h, alpha, beta, innerfunc
    !all partitionarrays in a row to calculate beta
    integer(i4), &
20      dimension((size(beta)*(size(beta)-1))/2) :: partitions
    !functions for the rosenbrock-integration
    ! initial conditions
    real(dp), dimension(2)              :: yi
    ! output time
    real(dp), dimension(:),             allocatable :: g1_xout, g2_xout
25  ! output solutions
    real(dp), dimension(:, :),         allocatable :: g1_out, g2_out
    ! fundamental solutions evaluated at r(rad_n)
    real(dp), dimension(1)             :: g1temp, g2temp
30  ! parameters for the diff-equation
    real(dp), dimension(5)             :: g_para

    !variables for the rosenbrock-integration
    ! boundary
    real(dp)                             :: rinf, rmin, rmax, rlim
35  ! guessed stepsize
    real(dp)                             :: hstep
    ! minimum allowed stepsize
    real(dp)                             :: hmin
40  ! accuracy
    real(dp)                             :: eps

    !intern variables
    real(dp) &
45      :: prod, t, coef, th, tmid, t_temp, error
    ! matrix for the les to clue the solutions together
    real(dp), dimension(2,2)            :: mat
    ! left- and righthandside of the les
    real(dp), dimension(2)              :: lhs, rhs

50  !loop-variables
    integer(i4) &
        :: n, m, rad_n, laps_n, nn, mm

    !intern variable-names from para
55  real(dp)                             :: tg, sig2, c, st, qw, n_series
    integer(i4)                          :: n_steh
    logical                               :: output

    !define the intern variables
60  tg = para(1)
    sig2 = para(2)

```

```

c          = para(3)
st         = para(4)
qw         = para(5)
65 n_series = para(6)
n_steh     = nint(para(7),i4)
output     = nint(para(8),i4)==1

70 !calculate the series-expansion of alpha
alpha      = 0.0
alpha(1)   = 1.0

do n=0, min(floor(n_series/2.0 - 1.0), ((stdsiz-1)/2 - 1))
75   alpha(2*(n+1)+1) = ((-1.0)**n)*sig2*real(n+1,dp)
end do

!calculate the series-expansion of beta=-r^2*st/tcg(r)/c^2
80 beta      = 0.0
partitions = 0

!set all partitions to the "biggest" one
do n=1,size(beta)-1
85   partitions((n*(n+1))/2) = 1
end do

!define the series expansion of the inner function in beta
innerfunc  = 0.0
innerfunc(1) = 0.5*sig2

90 do n=1, min(floor(n_series/2.0), ((stdsiz-1)/2))
   innerfunc(2*n+1) = 0.5*((-1.0)**n)*sig2
end do

95 beta(3)      = 1.0

!define the series expansion of beta by the rule of faa di bruno
do n=2,min(2*floor(n_series/2.0), stdsiz-1)-2,2
100  do
    prod=1.0
    do m=1, n
      if ((partitions(((n-1)*n)/2+m) .ne. 0)) then
105        if (ne(innerfunc(m+1), 0.0)) then
            prod = &
            prod*(innerfunc(m+1)**partitions(((n-1)*n)/2+m))&
            /real(factorial(int(partitions(((n-1)*n)/2+m),i8)),dp)
          else
            prod = 0.0
            exit
110          end if
        end if
      end do
    beta(n+3)      = beta(n+3)+prod

115    if (partitions(((n-1)*n)/2+1)==n) exit

    partitions(((n-1)*n)/2+1:((n+1)*n)/2) &
      = nextpart(partitions(((n-1)*n)/2+1:((n+1)*n)/2))

120    end do
  end do

125 beta          = -(st/tg/c**2)*exp(0.5*sig2)*beta

!define the boundaries for the different solutions

```

```

rinf                = 0.50
!define the upper limit for the solution such that
!the relativ error between tcg(r) and tg is less then 1% (error)
130 error            = 0.01
rlim                = sqrt( -0.5*sig2/log(1.0-error) - 1.0 )
!define a weighted mean between th and tg
th                  = tg*exp(-sig2*0.5) !nearf. transmissivity
135 tmid             = (tg+2.0*th)/3.0

!parameters for the runge-kutta-integration with adaptive stepsize
hstep              = 1e-8                ! incremental time step
140 hmin             = 0.0                ! minimum allowed stepsize
eps                = 1e-8                ! accuracy

!optional output
if (output) then
  write (*,*) ""
  write (*,*) "rinf:_"      , rinf/c, "rinf*c:_" , rinf
145 write (*,*) "rlim:_"    , rlim/c, "rlim*c:_" , rlim
  write (*,*) "t_err:_"    , error*100.0,"percent"
  write (*,*) "c:_"        , c
  write (*,*) "sig^2:_"    , sig2
  write (*,*) "tg:_"       , tg          , "th:_" , th
150 write (*,*) "tlim:_"   , tg*exp(-0.5*(sig2)/(1.0+(rlim)**2))
  write (*,*) "tmid:_"    , tmid
  write (*,*) ""
endif

155 !loop over s(:)
do nn=1, size(s)/n_steh
  t_temp            = 1.0/(s(nn)/log(2.0))

  !define the boundaries by comparisson to
160 !theis with the value u=r^2*st/(4*l*tmid*t)<0.25
  rmax              = min(sqrt(1.0*t_temp*tmid/st)*c, rlim)
  rmin              = min(rmax/2.0, rinf)

  if (output) then
165   write (*,*) "t_temp:_" , t_temp,&
     "rmin:_" , rmin/c,&
     "rmax:_" , rmax/c,&
     "eps:_" , eps,&
170   "t(rmax):_" , tg*exp(-0.5*sig2/(1.0+rmax**2))
  endif

do mm=0, n_steh-1
  laps_n           = mm*size(s)/n_steh + nn

175 !define the series expansion of the first fundamental solution
!g with the aid of alpha and beta by the frobenius-method
g                  = 0.0
g(1)               = 1.0
dg                 = 0.0

180 do n=1, size(g)-1
  do m=0, n-1
    g(n+1)         = g(n+1)+(real(m,dp)*alpha(n+1-m)&
185     + s(laps_n)*beta(n+1-m))*g(m+1)
  end do
  g(n+1)           = -1.0/(real(n,dp)**2)*g(n+1)
  dg(n)            = real(n,dp)*g(n+1)
end do

190 !define the series of the the first part of the second
!fundamental solution h with the aid of alpha, beta & g

```

```

h          = 0.0
h(1)      = 0.0 !1.0
dh        = 0.0

195
do n=1, size(h)-1
  do m=0, n-1
    h(n+1) = h(n+1)&
200           +alpha(n+1-m)*(real(m,dp)*h(m+1)+g(m+1))&
           +s(laps_n)*beta(n+1-m)*h(m+1)

    end do
    h(n+1) = -1.0/(real(n,dp)**2) &
205           * (h(n+1)+2.0*real(n,dp)*g(n+1))

    dh(n) = real(n,dp)*h(n+1)
  end do

!calculate 2 fundamental solutions in [rmin,rmax]
!with rosenbrock-integration
g_para    = (/sig2, s(laps_n), st, tg, c/)

210
coef      = -qw/(2.0*pi_d*th*s(laps_n))

! 1. solution
yi        = (/ poly(rmin,g),&
215           poly(rmin,dg) /)

call rbstiff( yi, rmin, rmax, hstep, g2d, dg2d,&
              g_para, g1_xout, g1_out, hmin=hmin, eps=eps)

220
! 2. solution
yi        = (/ coef*(poly(rmin,h)&
225           + log(rmin)*poly(rmin,g))&
           + coef*(poly(rmin,dh)&
           + log(rmin)*poly(rmin,dg))&
           + poly(rmin,g)/rmin) /)

call rbstiff( yi, rmin, rmax, hstep, g2d, dg2d,&
              g_para, g2_xout, g2_out, hmin=hmin, eps=eps)

230
!clue the solution together with theis in the farfield
mat       = 0.0
rhs       = 0.0

t         = sqrt(s(laps_n)*st&
235           / (tg*exp(-0.5*sig2/(1.0+rmax**2))))/c

mat(1,1:1) = interpol(g1_out(:,1),g1_xout,(/rmax/))
mat(1,2)   = -bessk0(t*rmax)
mat(2,1:1) = interpol(g1_out(:,2),g1_xout,(/rmax/))
240 mat(2,2)   = t*bessk1(t*rmax)

rhs(1:1)   = -interpol(g2_out(:,1),g2_xout,(/rmax/))
rhs(2:2)   = -interpol(g2_out(:,2),g2_xout,(/rmax/))

245 lhs      = solve_linear_equations(mat,rhs)

!calculate the head in laplace-space
do rad_n=1, size(r)
  if (r(rad_n)<rmin) then
250     lap_ext_theis2d(rad_n, laps_n)&
     = (coef*log(r(rad_n))+ lhs(1))*poly(r(rad_n),g)&
     + coef*poly(r(rad_n),h)
  else if (r(rad_n)>rmax) then
     lap_ext_theis2d(rad_n, laps_n)
255     = lhs(2)*bessk0(t*r(rad_n))
  else

```

```

                g1temp = interpol(g1_out(:,1),g1_xout,(/r(rad_n)/))
                g2temp = interpol(g2_out(:,1),g2_xout,(/r(rad_n)/))
lap_ext_theis2d(rad_n, laps_n)
                = lhs(1)*g1temp(1) + g2temp(1)
260         end if
        end do

!laps_n
265     end do !m
    end do !n

end function lap_ext_theis2d

270     ! returns derivatives dydx at x for the fundamental solution
    subroutine g2d( x, y, para, dydx )

        implicit none

275         ! time
        real(dp), intent(in) :: x
        ! unknowns of the equations
        real(dp), dimension(:), intent(in) :: y
        ! para=(sigma**2, s, s, tg, c)
280         real(dp), dimension(:), intent(in) :: para
        ! derivatives of y
        real(dp), dimension(:), intent(out) :: dydx

        dydx(1) = y(2)
285         dydx(2) = (-para(1)*x/((1+x**2)**2) - 1.0/x) * y(2) &
            + (exp(0.5*para(1)/(1+x**2))*para(2)&
                * para(3)/para(4)/((para(5)**2)) * y(1)

    end subroutine g2d

290     ! returns derivatives dydx at x for the fundamental solution
    subroutine dg2d( x, y, para, dfdx, dfdy )

        implicit none

295         ! time
        real(dp), intent(in) :: x
        ! unknowns of the equations
        real(dp), dimension(:), intent(in) :: y
        ! para=(sigma**2, s, s, tg, c)
300         real(dp), dimension(:), intent(in) :: para
        ! derivatives of f
        real(dp), dimension(:), intent(out) :: dfdx
        ! derivatives of f
305         real(dp), dimension(:,,:), intent(out) :: dfdy

        dfdx(1) = 0.0
        dfdx(2) = ( -para(1)*(1.0 - 3.0*x**2)&
            / ((1+x**2)**3) + 1.0/(x**2) ) * y(2) &
310             + ( exp(0.5*para(1)/(1+x**2))*para(2)&
                * para(3)/para(4)/((para(5)**2))&
                * para(1)*x/((1+x**2)**2) ) * y(1)

        dfdy(1,1) = 0.0
315         dfdy(1,2) = 1.0
        dfdy(2,1) = (exp(0.5*para(1)/(1+x**2))*para(2)&
            * para(3)/para(4)/((para(5)**2))
        dfdy(2,2) = (-para(1)*x/((1+x**2)**2) - 1.0/x)

320     end subroutine dg2d

```

Programmcode 11: Implementierung der Funktion `ext_theis3D` welche die erweiterte Theis-Lösung im 3D-Fall berechnet

```

1  function ext_theis3d(rad,time,params,init, bc,prop,grad, steh_n,output)

    implicit none

5     !radius
    real(dp), dimension(:),          intent(in)  :: rad
    !time
    real(dp), dimension(:),          intent(in)  :: time
    !params=(kg,sigma**2,corr,s,e)
10    real(dp), dimension(5),         intent(in)  :: params
    !init=(1,qw)
    real(dp), dimension(2),          intent(in)  :: init
    !bc-kind (harmonic: true[default], arithmetic: false)
    logical ,                        optional,  intent(in)  :: bc
15    !prop-fac. for harm.case, arith.case autocorr. [def=1.6]
    real(dp),                        optional,  intent(in)  :: prop
    !limit of the series expansion [def=40]
    integer(i4),                     optional,  intent(in)  :: grad
    !the boundary for the stehfest-alg. [def=6]
20    integer(i4),                     optional,  intent(in)  :: steh_n
    !enables output while computation
    logical ,                        optional,  intent(in)  :: output

    real(dp), dimension(size(rad),size(time))      :: ext_theis3d

25    real(dp) &
        :: pr, aniso, kwell, kefu, chi, output_real, c
    real(dp), dimension(11)          :: values
    integer(i4)                      :: stehlim, limit, n
30    logical                        :: bckind

    logical, dimension(size(rad), size(time))      :: mask

35    ! -----
    ! input tests
    ! -----

40    if (.not. all(rad>0.0))          &
        stop 'radius must be positiv.'
    if (any(time < 0.0))              &
        stop 'the time must be non-negativ.'
    if (le(init(1), 0.0))             &
        stop 'the aquifer thickness must be positiv.'
45    if (ge(init(2), 0.0))            &
        stop 'the pumping-rate must be negativ.'
    if (le(params(1), 0.0))           &
        stop 'the geometric mean conductivity must be positiv.'
    if (le(params(2), 0.0))           &
        stop 'the variance must be positiv.'
50    if (le(params(3), 0.0))         &
        stop 'the correlation length must be positiv.'
    if (le(params(4), 0.0))           &
        stop 'the storage must be positiv.'
    if (params(5) .lt. 0.0)           &
55    stop 'the anisotropy ratio must be in [0,1].'
    if (params(5) .gt. 1.0)           &
        stop 'the anisotropy ratio must be in [0,1].'

60    if (present(bc)) then
        bckind = bc
    else
        bckind = .true.

```

```

endif
65  if (present(output)) then
      if (output) then
          output_real = 1.0
      else
70      output_real = 0.0
      endif
    else
      output_real = 0.0
    endif

75
    if (present(prop)) then
        if (le(prop, 0.0)) &
            stop 'the proportionality factor must be positiv.'
        pr = prop
80    else
        pr = 1.6
    endif

    if (present(steh_n)) then
        if (steh_n <= 0) &
            stop 'the stehfest-boundary must be positiv.'
        stehlim = steh_n
90    else
        stehlim = 6
    endif

    if (present(grad)) then
        if (grad <= 3) &
            stop 'the limit of the expansion must be at least 4.'
95        limit = grad
    else
        limit = 40
    endif

100 ! -----
! setting of the intern variables
! -----

105 if (.not. bckind) pr = 0.5*pr

c = pr / ( (params(5)**(1.0/3.0))*params(3) )

if (eq(params(5), 1.0)) then
    aniso = 1.0/3.0
110 else if (eq(params(5), 0.0)) then
    aniso = 0.0
else
    aniso = (params(5)/2.0) * &
115           (&
            ( (1.0-params(5)**2)**(-1.5) ) &
            * atan((1.0/params(5)**2 - 1.0)**(0.5)) &
            - params(5)/(1.0-params(5)**2) &
            )
endif

120 kefu = params(1)*exp(params(2)*(0.5 - aniso))

if (bckind) then
    kwell = params(1)*exp(-params(2)/2.0)
125 else
    kwell = params(1)*exp(params(2)/2.0)
endif

```

```

130     chi          =  log(kwell/kefu)
values      =  (/&
              params(1),          &! 1
              kwell,              &! 2
              kefu,                &! 3
135             chi,                &! 4
              c,                    &! 5
              params(4),          &! 6
              inits(1),           &! 7
              inits(2),           &! 8
140             real(limit,dp),    &! 9
              real(stehlim,dp),   &! 10
              output_real         &! 11
              /)

145     !the optional output
if (nint(output_real,i4)==1) then
    write (*,*) ""
    write (*,*) "kg    ",params(1)
    write (*,*) "sig2:  ",params(2)
150    write (*,*) "corr:  ",params(3)
    write (*,*) "s    :  ",params(4)
    write (*,*) "e    :  ",params(5)
    write (*,*) "l    :  ",inits(1)
    write (*,*) "qw    :  ",inits(2)
155    write (*,*) "prop:  ", pr
    if (bckind) then
        write (*,*) "bc  :  harmonic"
    else
        write (*,*) "bc  :  arithmetic"
160    endif
end if

! -----
! calculate the head
! -----

!set all heads to zero where time is zero
do n=1, size(rad)
    mask(n,:) = (time>0.0)
170 end do

ext_theis3d = 0.0

ext_theis3d =&
175     unpack( pack(nlinvsteh( lap_ext_theis3d,&
                             values, c*rad,&
                             pack(time, (time>0.0)),&
                             stehlim),&
               .true.),&
180     mask,&
     ext_theis3d)

end function ext_theis3d

```

Programmcode 12: Implementierung der Funktion lap_ext_theis3D welche die Lösung der erweiterten Theis-Lösung für den 3D-Fall im Laplaceraum berechnet

```

1  function lap_ext_theis3d(r, s, para)

    implicit none

5   real(dp), dimension(:),          intent(in)  :: r
    real(dp), dimension(:),          intent(in)  :: s
    !para=(kefu,chi,c,s,qw,l,kwell,ref,kref,limit,stehl,kg)
    real(dp), dimension(:),          intent(in)  :: para

10  real(dp), dimension(size(r), size(s))      :: lap_ext_theis3d

    !the standard limit for the series-expansion
    integer(i4), parameter              :: stdsiz=41
    !expansions of the given function
15  real(dp), dimension(max(nint(para(9),i4)+1,stdsiz)) &
        :: dg,dh,g,h,alpha,beta,innerfunc
    !all partitionarrays in a row to calculate beta
    integer(i4), &
20      dimension((size(beta)*(size(beta)-1))/2)  :: partitions
    !functions for the rosenbrock-integration
    ! initial conditions
    real(dp), dimension(2)              :: yi
    ! output time
    real(dp), dimension(:),             allocatable  :: g1_xout, g2_xout
25  ! output solutions
    real(dp), dimension(:, :),         allocatable  :: g1_out, g2_out
    ! fundamental solutions evaluated at r(rad_n)
    real(dp), dimension(1)             :: g1temp, g2temp
30  ! parameters for the diff-equation
    real(dp), dimension(5)              :: g_para

    !variables for the rosenbrock-integration
    ! boundarys
    real(dp)                             :: rinf,rmin,rmax,rlim
35  ! guessed stepsize
    real(dp)                             :: hstep
    ! minimum allowed stepsize
    real(dp)                             :: hmin
40  ! accuracy
    real(dp)                             :: eps

    !intern variables
    real(dp) &
45      :: prod,coef,t,t_temp,error,kmid
    real(dp), dimension(2,2)             :: mat
    real(dp), dimension(2)              :: lhs, rhs

    !loop-variables
    integer(i4) &
50      :: n,m,rad_n,laps_n,nn,mm

    !internt variable-names from para
    real(dp)                             :: kg,kwell,kefu,chi,c,st,l,qw,n_series
    integer(i4)                          :: n_steh
55  logical                              :: output

    !define the intern variables
    kg = para(1)
60  kwell = para(2)
    kefu = para(3)
    chi = para(4)

```

```

c          = para(5)
st         = para(6)
l          = para(7)
65  qw      = para(8)
n_series   = para(9)           !is real
n_steh     = nint(para(10),i4)
output     = nint(para(11),i4)==1

70  !calculate the series-expansion of
!alpha=1-3chi*(c*r)^2*(1+(c*r)^2)^(5/2)
alpha      = 0.0
alpha(1)   = 1.0

75  do n=0,min(floor(n_series/2.0-1.0),((stdsiz-1)/2-1))
      alpha(2*(n+1)+1) = -3.0*chi*binomcoeffi(-2.5,n)
end do
!calculate the series-expansion of beta=-r^2*st/c/kgf(r)
beta       = 0.0
80  partitions = 0
!set all partitions to the "biggest" one
do n=1,size(beta)-1
      partitions((n*(n+1))/2) = 1
end do

85  !define the series expansion of the inner function in beta
innerfunc  = 0.0
innerfunc(1) = -chi

90  do n=1, min(floor(n_series/2.0), ((stdsiz-1)/2))
      innerfunc(2*n+1) = -chi*binomcoeffi(-1.5,n)
end do
beta(3)    = 1.0
!define the series expansion of beta by the rule of faa di bruno
95  do n=2,min(2*floor(n_series/2.0), stdsiz-1)-2,2
      do
          prod=1.0
          do m=1, n
              if ((partitions(((n-1)*n)/2+m) .ne. 0)) then
100             if (ne(innerfunc(m+1), 0.0)) then
                    prod =&
                    prod*(innerfunc(m+1)**partitions(((n-1)*n)/2+m))&
                    /real(factorial(int(partitions(((n-1)*n)/2+m),i8)),dp)
              else
105             prod = 0.0
                    exit
              end if
            end if
          end do
110         beta(n+3) = beta(n+3)+prod

          if (partitions(((n-1)*n)/2+1)==n) exit

115         partitions(((n-1)*n)/2+1:((n+1)*n)/2)&
            = nextpart(partitions(((n-1)*n)/2+1:((n+1)*n)/2))

          end do
        end do

120  beta = -(st/kefu)*exp(-chi)/(c**2)*beta

!define the boundaries for
!the different solutions (just reference-values)
125  rinf = 0.25
!define the upper limit for the solution such that

```

```

!the relativ error between kcg(r) and kefu is less than 1% (error)
error = 0.01
if (chi>0.0) then
130   rlim = sqrt( (chi/log(1.0+error))**(2.0/3.0) - 1.0 )
elseif (chi<0.0) then
   rlim = sqrt( (chi/log(1.0-error))**(2.0/3.0) - 1.0 )
else
   rlim = 0.5
135 endif
!define a weighted mean between kwell and kefu
kmid = (kefu+2.0*kwell)/3.0

!parameters for the rosenbrock-integration with adaptive stepsize
140 hstep = 1e-12 ! incremental time step
hmin = 0.0 ! minimum allowed stepsize
eps = 1e-12 ! accuracy

!optional output
145 if (output) then
   write (*,*) ""
   write (*,*) "rinf:␣", rinf/c, "␣rinf*c:␣", rinf
   write (*,*) "rlim:␣", rlim/c, "␣rlim*c:␣", rlim
150   write (*,*) "k_err:", error*100.0, "percent"
   write (*,*) "c:␣␣␣␣", c
   write (*,*) "chi:␣␣", chi
   write (*,*) "kg:␣␣␣␣", kg
   write (*,*) "kefu:␣", kefu, "␣kwell:␣␣", kwell
155   write (*,*) "klim:␣", kefu*exp(chi/(sqrt((1.0+(rlim)**2))**3))
   write (*,*) "kmid:␣", kmid
   write (*,*) ""
endif

!loop over s(:)
160 do nn=1, size(s)/(n_steh)

   t_temp = 1.0/(s(nn)/log(2.0))
   !define the boundaries by comparisson to
   !this with the value u=r^2*st/(4*l*kmid*t)<0.25
165   rmax = min(sqrt(1.0*t_temp*kmid/st)*c, rlim)
   rmin = min(rmax/2.0, rinf)

   if (output) then
170     write (*,*) "t_temp:␣", t_temp, &
       "␣rmin:␣", rmin/c, &
       "␣rmax:␣", rmax/c, &
       "␣eps:␣", eps, &
       "␣k_cg(rmax):␣", &
175     kefu*exp(chi/(sqrt((1.0+(rmax)**2))**3))
   endif

do mm=0, n_steh-1
   laps_n = mm*size(s)/(n_steh) + nn

180   !define the series expansion of the first fundamental solution
   !g with the aid of alpha and beta by the frobenius-method
   g = 0.0
   g(1) = 1.0
   dg = 0.0
185

   do n=1, size(g)-1
     do m=0, n-1
       g(n+1) = g(n+1)+(real(m,dp)*alpha(n+1-m) &
190         + s(laps_n)*beta(n+1-m))*g(m+1)
     end do
     g(n+1) = -1.0/(real(n,dp)**2)*g(n+1)

```

```

        dg(n)          =  real(n,dp)*g(n+1)
    end do

195    !define the series of the the first part of the second fund.
    !solution h with the aid of alph, beta & g by the frob.-m.
    h              =  0.0
    h(1)           =  0.0 !1.0
    dh             =  0.0

200
    do n=1,size(h)-1
        do m=0,n-1
            h(n+1)=h(n+1)+alpha(n+1-m)*(real(m,dp)*h(m+1)+g(m+1))&
                + s(laps_n)*beta(n+1-m)*h(m+1)
205        end do
        h(n+1)      =  -1.0/(real(n,dp)**2)&
            * (h(n+1)+2.0*real(n,dp)*g(n+1))
        dh(n)       =  real(n,dp)*h(n+1)
    end do

210
    !calculate 2 fundamental solutions
    !in [rmin,rmax] with rosenbrock-integration
    ! g_para=(chi, s, st, kefu, c)
    g_para         =  (/chi, s(laps_n), st, kefu, c/)
215
    coef           =  -qw/(2.0*pi_d*1*kwell*s(laps_n))

    ! 1. solution
    yi             =  (/ poly(rmin,g),&
220                poly(rmin,dg) /)
    call rbstiff( yi,rmin,rmax,hstep,g3d,dg3d,&
        g_para,g1_xout,g1_out,hmin=hmin,eps=eps )

    ! 2. solution
    yi             =  (/coef*(poly(rmin,h)+log(rmin)*poly(rmin,g)),&
225                coef*(poly(rmin,dh)+log(rmin)*poly(rmin,dg)&
                    +poly(rmin,g)/rmin) /)
    call rbstiff( yi,rmin,rmax,hstep,g3d,dg3d,&
        g_para,g2_xout,g2_out,hmin=hmin,eps=eps )

230
    !clue the solution together with theis in the farfield (g c~1)
    mat            =  0.0
    rhs            =  0.0
    t              =  sqrt(s(laps_n)*st/kefu)/c

235
    mat(1,1:1)    =  interpol(g1_out(:,1) , g1_xout, (/rmax/))
    mat(1,2)      =  -bessk0(t*rmax)
    mat(2,1:1)    =  interpol(g1_out(:,2) , g1_xout, (/rmax/))
    mat(2,2)      =  t*bessk1(t*rmax)

240
    rhs(1:1)      =  -interpol(g2_out(:,1) , g2_xout, (/rmax/))
    rhs(2:2)      =  -interpol(g2_out(:,2) , g2_xout, (/rmax/))

    lhs           =  solve_linear_equations(mat,rhs)

245
    !calculate the head in laplace-space
    do rad_n=1, size(r)
        if (r(rad_n)<rmin) then
            lap_ext_theis3d(rad_n, laps_n)&
                =  (coef*log(r(rad_n))+lhs(1))*poly(r(rad_n),g)&
250                + coef*poly(r(rad_n),h)
        else if (r(rad_n)>rmax) then
            lap_ext_theis3d(rad_n, laps_n)&
                =  lhs(2)*bessk0(t*r(rad_n))
        else
255            g1temp= interpol(g1_out(:,1) , g1_xout, (/r(rad_n)/))
            g2temp= interpol(g2_out(:,1) , g2_xout, (/r(rad_n)/))

```

```

                lap_ext_theis3d(rad_n, laps_n)&
                = lhs(1)*g1temp(1) + g2temp(1)
            end if
260         end do

        end do !m
        end do !n

265 end function lap_ext_theis3d

! returns derivatives dydx at x for the fundamental solution
subroutine g3d( x, y, para, dydx )

270     implicit none

        ! time
        real(dp), intent(in) :: x
        ! unknowns of the equations
275     real(dp), dimension(:), intent(in) :: y
        ! para=(chi, s, s, kefu, c)
        real(dp), dimension(:), intent(in) :: para
        ! derivatives of y
        real(dp), dimension(:), intent(out) :: dydx

280     dydx(1) = y(2)
        dydx(2) = (3.0*para(1)*x&
                    / (sqrt(1+x**2)**5) - 1.0/x) * y(2) &
                    + (exp(-para(1)/(sqrt(1+x**2)**3))&
285     * para(2)*para(3)/para(4)/((para(5)**2))*y(1)

    end subroutine g3d

! returns derivatives dfdx and dfdy at x for the solution
subroutine dg3d( x, y, para, dfdx, dfdy )

290     implicit none

        ! time
        real(dp), intent(in) :: x
        ! unknowns of the equations
295     real(dp), dimension(:), intent(in) :: y
        ! para=(chi, s, s, kefu, c)
        real(dp), dimension(:), intent(in) :: para
        ! derivatives of f for x
        real(dp), dimension(:), intent(out) :: dfdx
        ! derivatives of f for y
        real(dp), dimension(:, :), intent(out) :: dfdy

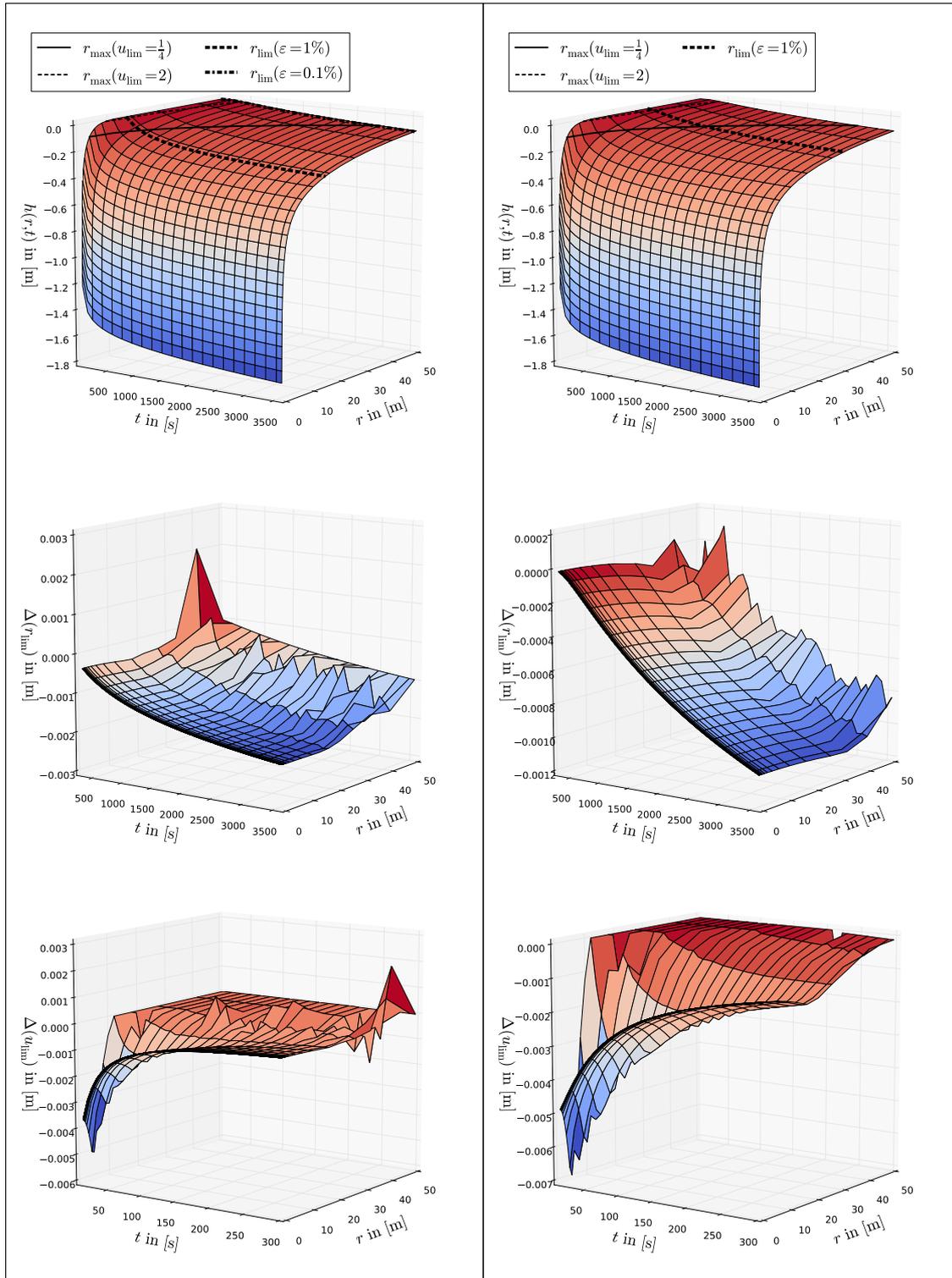
300     dfdx(1) = 0.0
        dfdx(2) = ( 3.0*para(1)*(1.0 - 4.0*x**2)&
                    / (sqrt(1+x**2)**7) + 1.0/(x**2) ) * y(2) &
                    + ( exp(-para(1)/(sqrt(1+x**2)**3))&
310     * para(2)*para(3)/para(4)/((para(5)**2))*&
                    3.0*para(1)*x/(sqrt(1+x**2)**5) ) * y(1)

        dfdy(1,1) = 0.0
        dfdy(1,2) = 1.0
        dfdy(2,1) = (exp(-para(1)/(sqrt(1+x**2)**3))&
315     * para(2)*para(3)/para(4)/((para(5)**2))
        dfdy(2,2) = (3.0*para(1)*x/(sqrt(1+x**2)**5) - 1.0/x)

    end subroutine dg3d

```

A.2. Visualisierung der Fehleranalyse



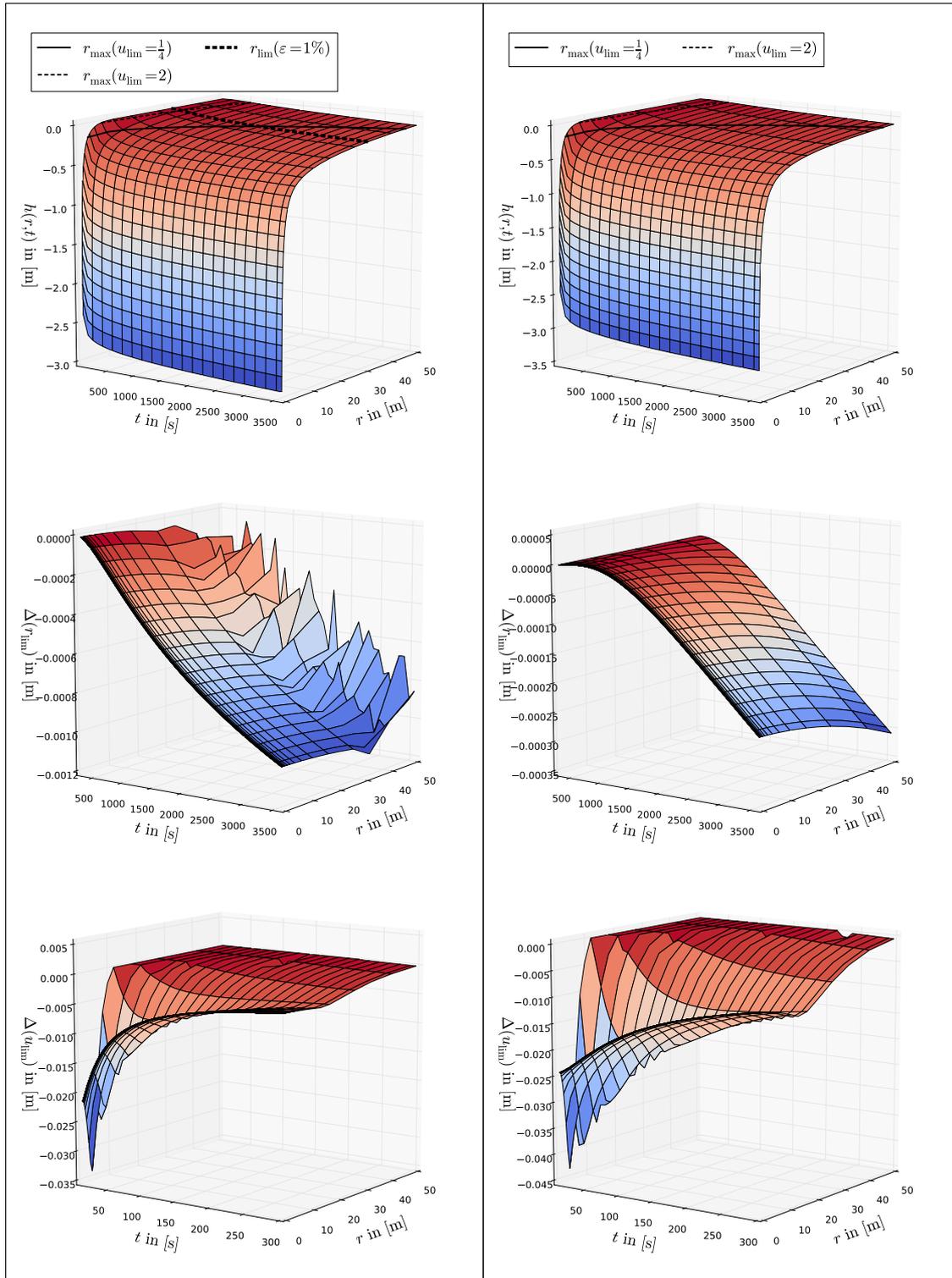
Parameter: $\sigma^2 = 0.5, \ell = 5$

Parameter: $\sigma^2 = 0.5, \ell = 10$

Abbildung 11: Oben: Lösung der Grundwassergleichung für das 2D-Coarse-Graining-Modell

Mitte: Differenz der Lösungen für $\epsilon_{\text{error}} \in \{1\%, 0.1\%\}$ mit $t > 5\text{min}$

Unten: Differenz der Lösungen für $u_{\text{lim}} \in \{\frac{1}{4}, 2\}$ mit $t < 5\text{min}$



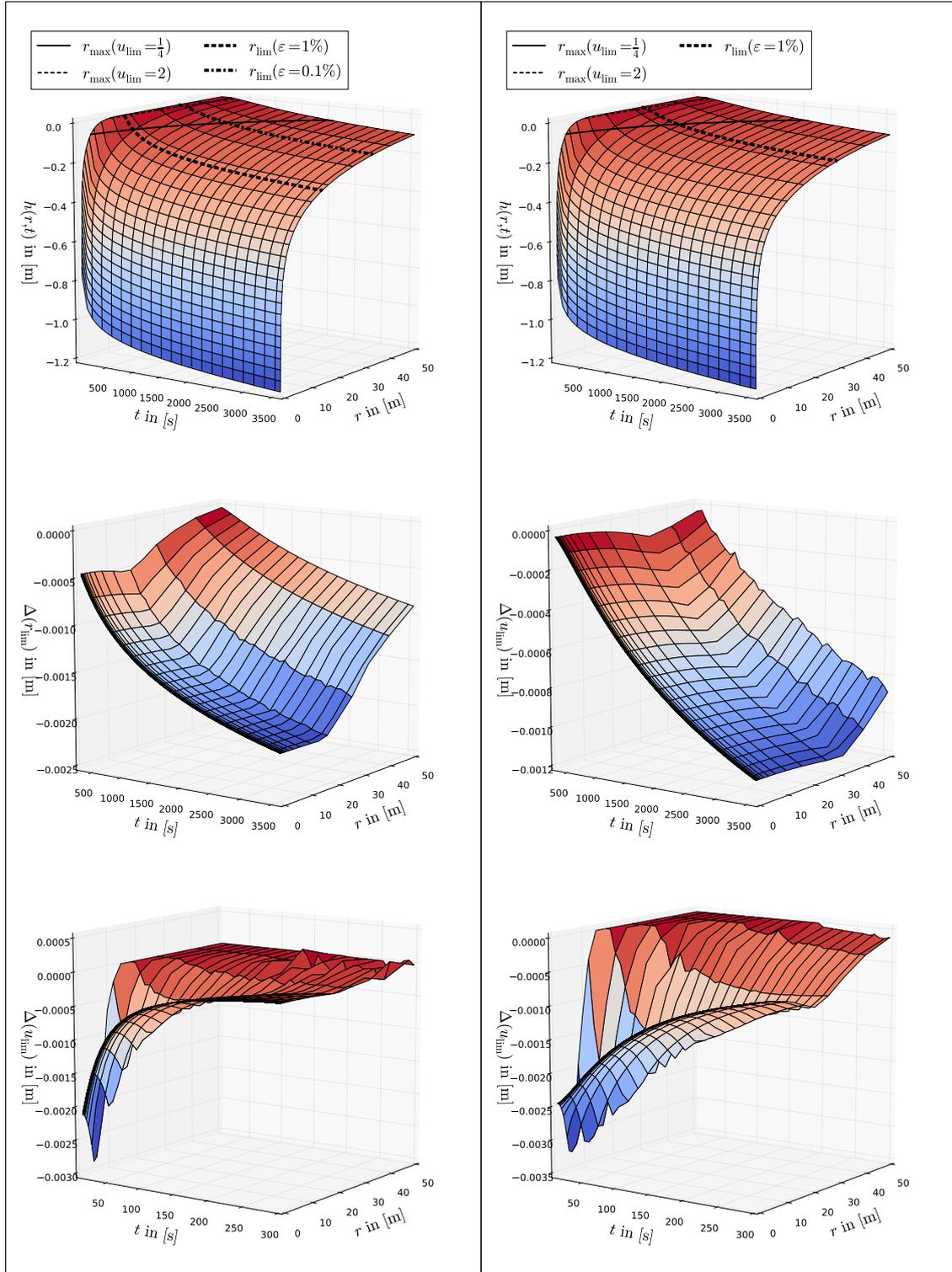
Parameter: $\sigma^2 = 2$, $\ell = 5$

Parameter: $\sigma^2 = 2$, $\ell = 10$

Abbildung 12: Oben: Lösung der Grundwassergleichung für das 2D-Coarse-Graining-Modell

Mitte: Differenz der Lösungen für $\varepsilon_{\text{error}} \in \{1\%, 0.1\%\}$ mit $t > 5\text{min}$

Unten: Differenz der Lösungen für $u_{\text{lim}} \in \{\frac{1}{4}, 2\}$ mit $t < 5\text{min}$



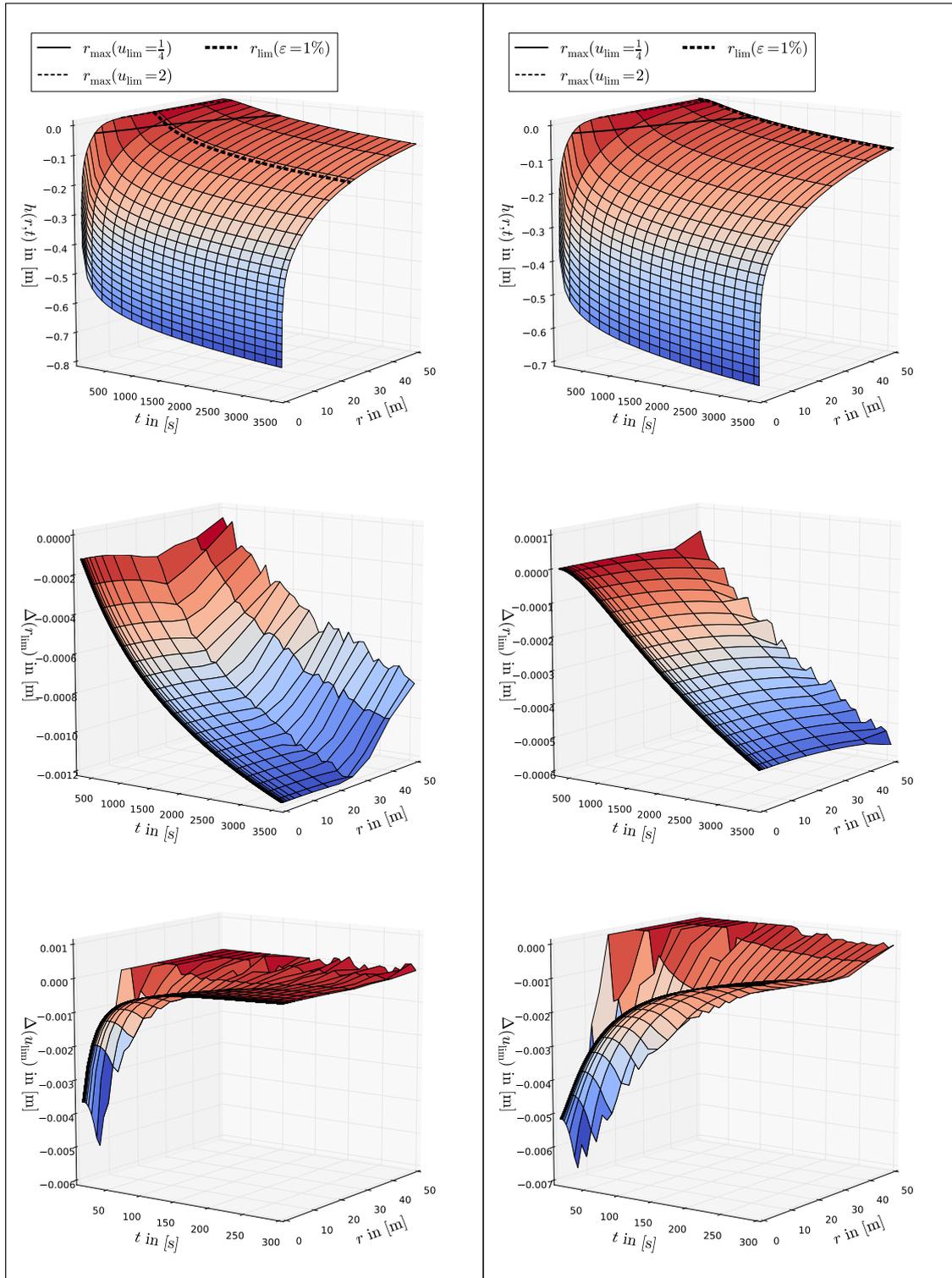
Parameter: $\sigma^2 = 0.5$, $\ell = 5$

Parameter: $\sigma^2 = 0.5$, $\ell = 10$

Abbildung 13: Oben: Lösung der Grundwassergleichung für das 3D-Coarse-Graining-Modell im BCA-Fall, also mit arithmetischer Mittlung am Brunnen ($K_{\text{well}} = K_A$)

Mitte: Differenz der Lösungen für $\varepsilon_{\text{error}} \in \{1\%, 0.1\%\}$ mit $t > 5\text{min}$

Unten: Differenz der Lösungen für $u_{\text{lim}} \in \{\frac{1}{4}, 2\}$ mit $t < 5\text{min}$



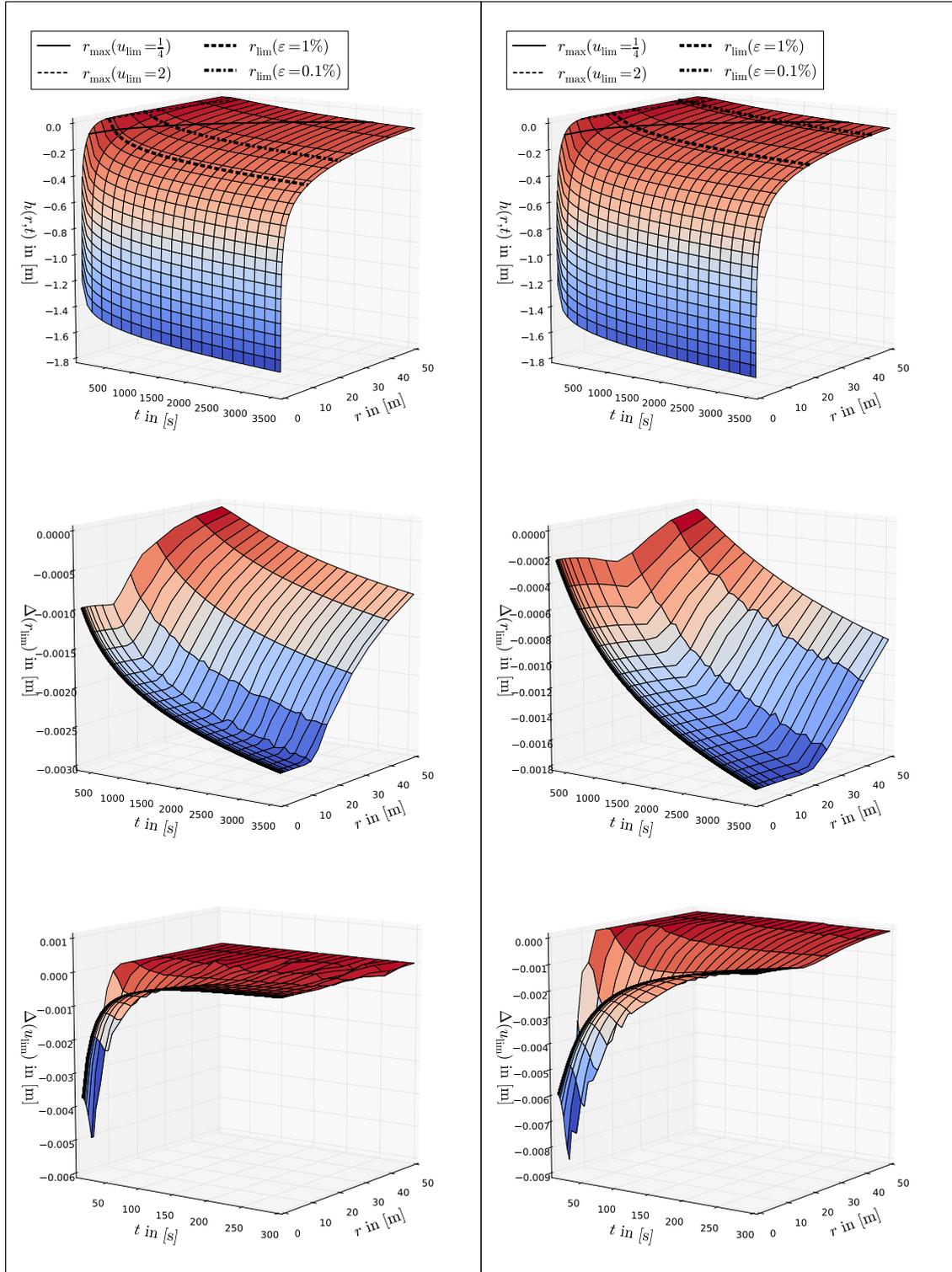
Parameter: $\sigma^2 = 2$, $\ell = 5$

Parameter: $\sigma^2 = 2$, $\ell = 10$

Abbildung 14: Oben: Lösung der Grundwassergleichung für das 3D-Coarse-Graining-Modell im BCA-Fall, also mit arithmetischer Mittlung am Brunnen ($K_{\text{well}} = K_A$)

Mitte: Differenz der Lösungen für $\varepsilon_{\text{error}} \in \{1\%, 0.1\%\}$ mit $t > 5\text{min}$

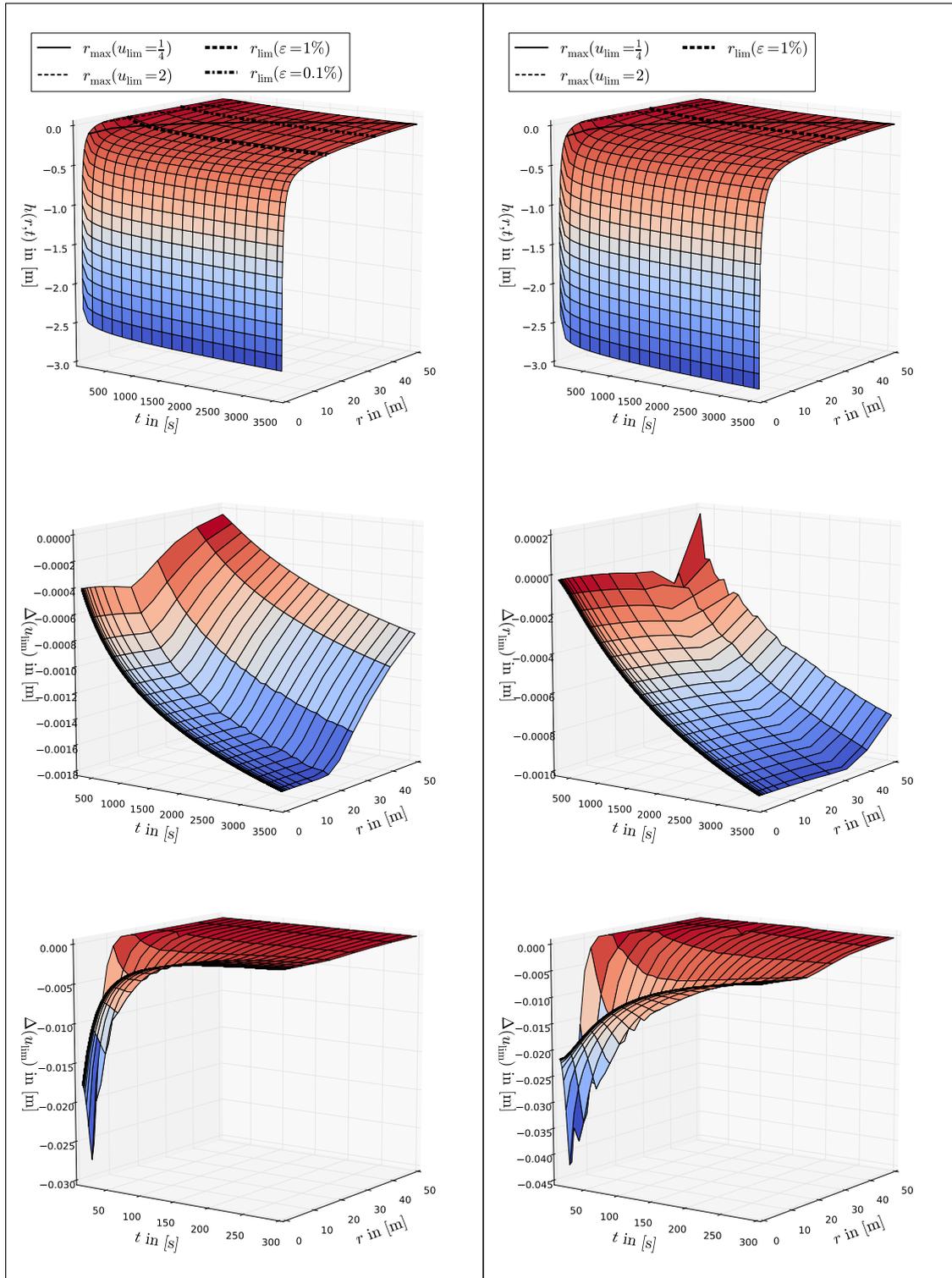
Unten: Differenz der Lösungen für $u_{\text{lim}} \in \{\frac{1}{4}, 2\}$ mit $t < 5\text{min}$



Parameter: $\sigma^2 = 0.5$, $\ell = 5$

Parameter: $\sigma^2 = 0.5$, $\ell = 10$

Abbildung 15: Oben: Lösung der Grundwassergleichung für das 3D-Coarse-Graining-Modell im BCH-Fall, mit harmonischer Mittelung am Brunnen ($K_{\text{well}} = K_{\text{H}}$)
 Mitte: Differenz der Lösungen für $\varepsilon_{\text{error}} \in \{1\%, 0.1\%\}$ mit $t > 5\text{min}$
 Unten: Differenz der Lösungen für $u_{\text{lim}} \in \{\frac{1}{4}, 2\}$ mit $t < 5\text{min}$



Parameter: $\sigma^2 = 2$, $\ell = 5$

Parameter: $\sigma^2 = 2$, $\ell = 10$

Abbildung 16: Oben: Lösung der Grundwassergleichung für das 3D-Coarse-Graining-Modell im BCH-Fall, also mit harmonischer Mittelung am Brunnen ($K_{\text{well}} = K_H$)

Mitte: Differenz der Lösungen für $\varepsilon_{\text{error}} \in \{1\%, 0.1\%\}$ mit $t > 5\text{min}$

Unten: Differenz der Lösungen für $u_{\text{lim}} \in \{\frac{1}{4}, 2\}$ mit $t < 5\text{min}$

A.3. Einfluss der Parameter auf die Lösung

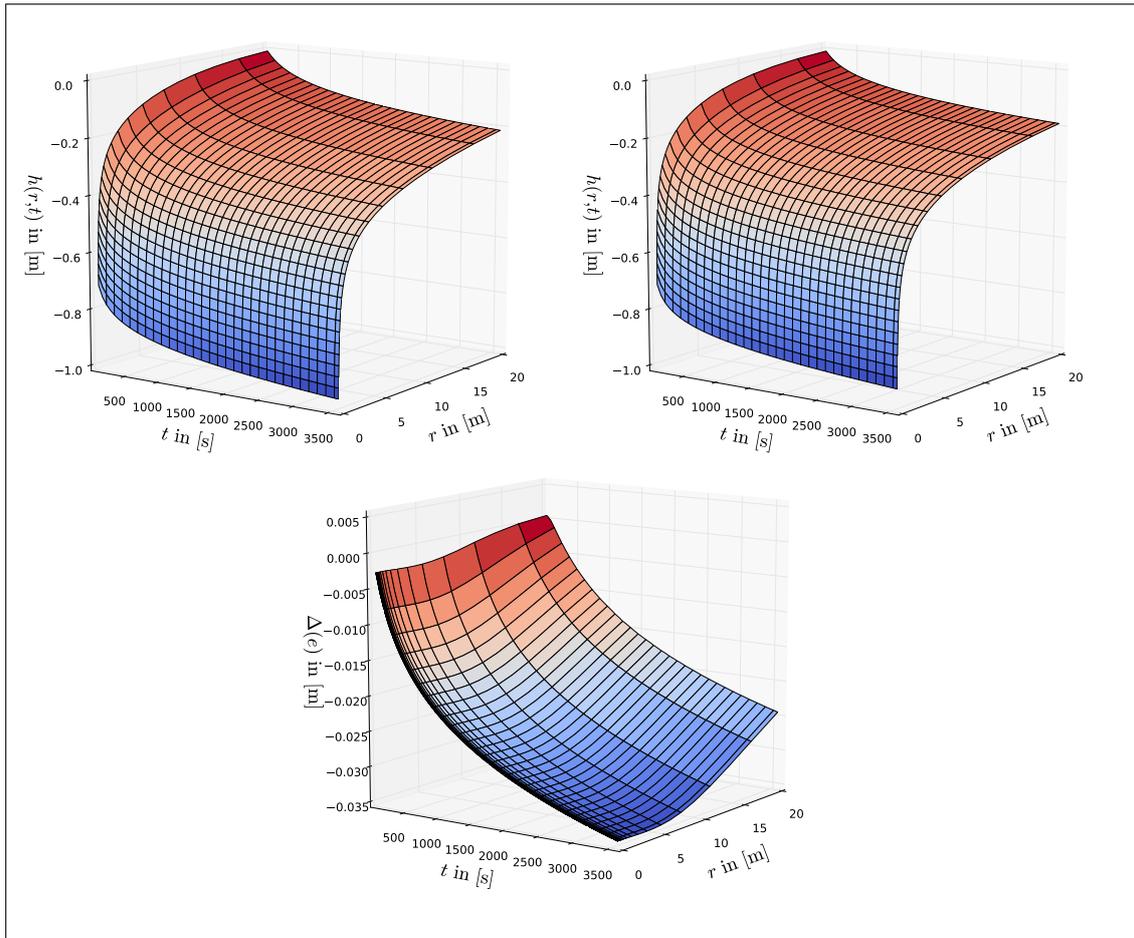


Abbildung 17: Einfluss der Anisotropierate e auf die 3D-Lösung im BCA-Fall, also $K_{\text{well}} = K_A$. Dabei wurde zum einen $e = 1$ (oben links) und zum anderen $e = \frac{1}{4}$ (oben rechts) gewählt. In der unteren Grafik sieht man die Differenz $\Delta(e)$ beider Lösungen.

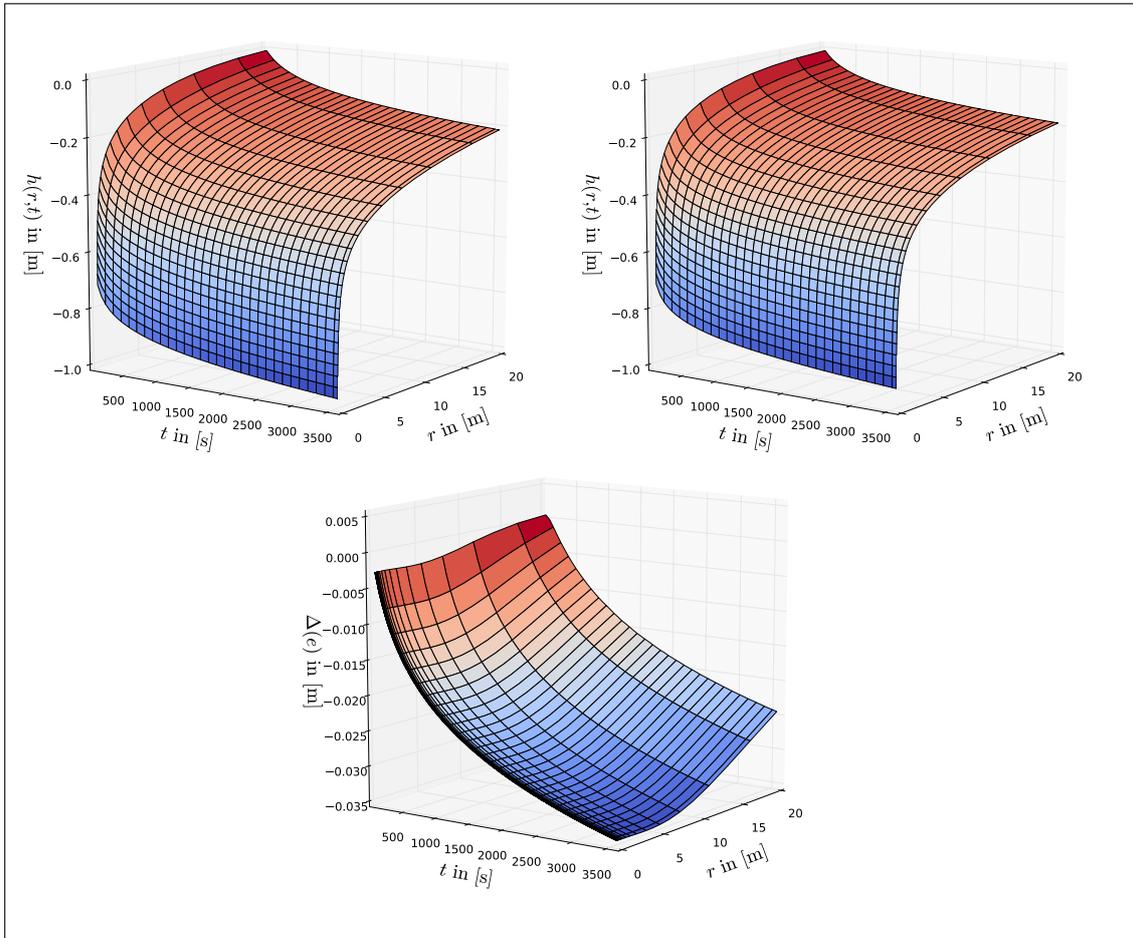


Abbildung 18: Einfluss der Anisotropierate e auf die $3D$ -Lösung im BCH-Fall, also $K_{\text{well}} = K_{\text{H}}$. Dabei wurde zum einen $e = 1$ (oben links) und zum anderen $e = \frac{1}{4}$ (oben rechts) gewählt. In der unteren Grafik sieht man die Differenz $\Delta(e)$ beider Lösungen.

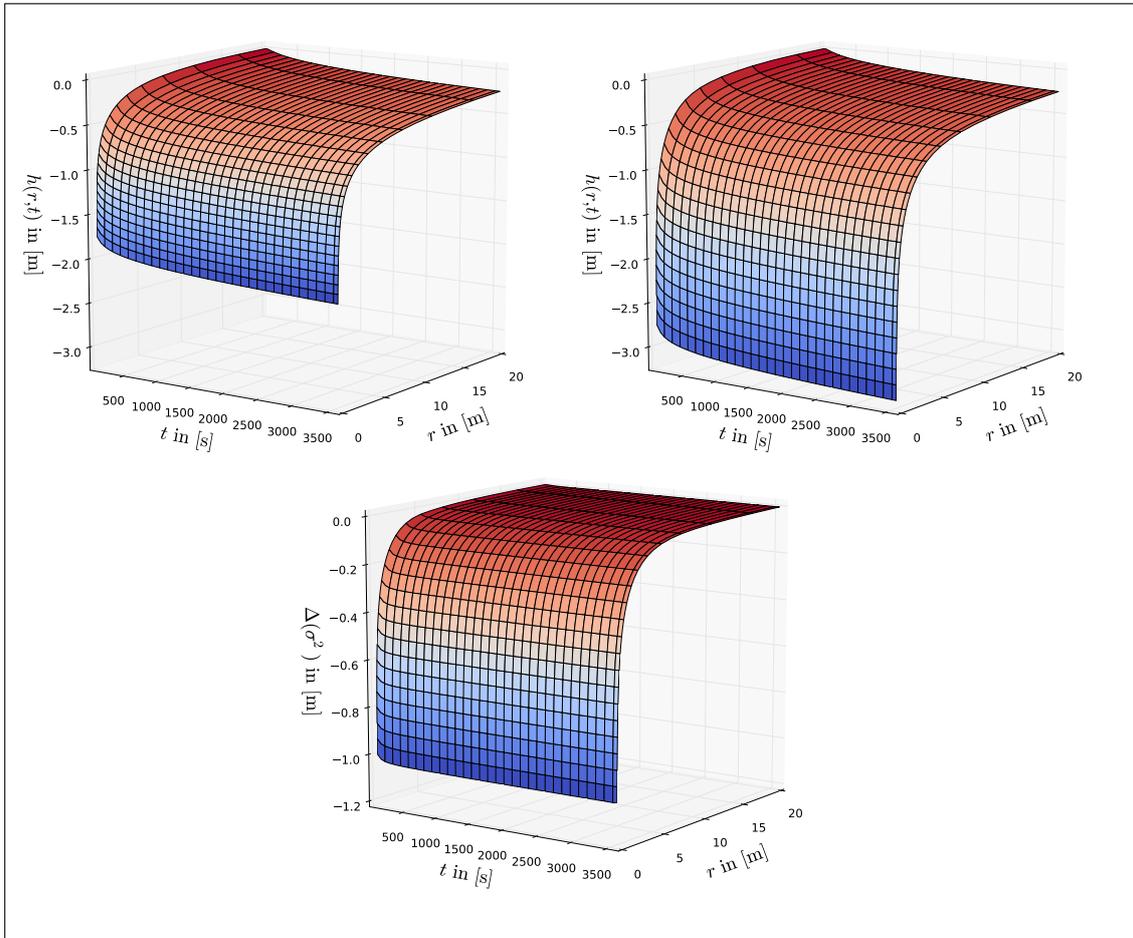


Abbildung 19: Einfluss der Varianz σ^2 auf die 2D-Lösung. Dabei wurde zum einen $\sigma^2 = 1$ (oben links) und zum anderen $\sigma^2 = 2$ (oben rechts) gewählt. In der unteren Grafik sieht man die Differenz $\Delta(\sigma^2)$ beider Lösungen.

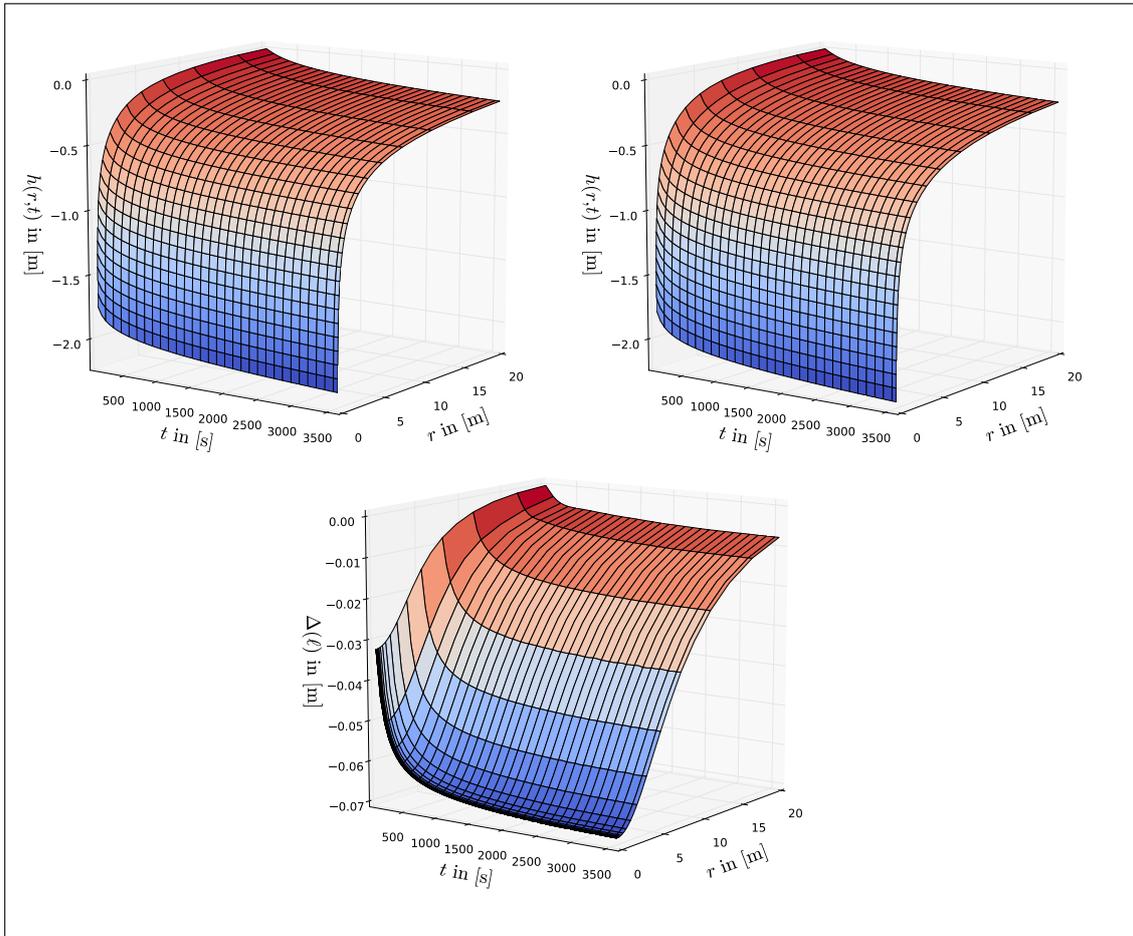


Abbildung 20: Einfluss der Korrelationslänge ℓ auf die 2D-Lösung. Dabei wurde zum einen $\ell = 10$ (oben links) und zum anderen $\ell = 20$ (oben rechts) gewählt. In der unteren Grafik sieht man die Differenz $\Delta(\ell)$ beider Lösungen.

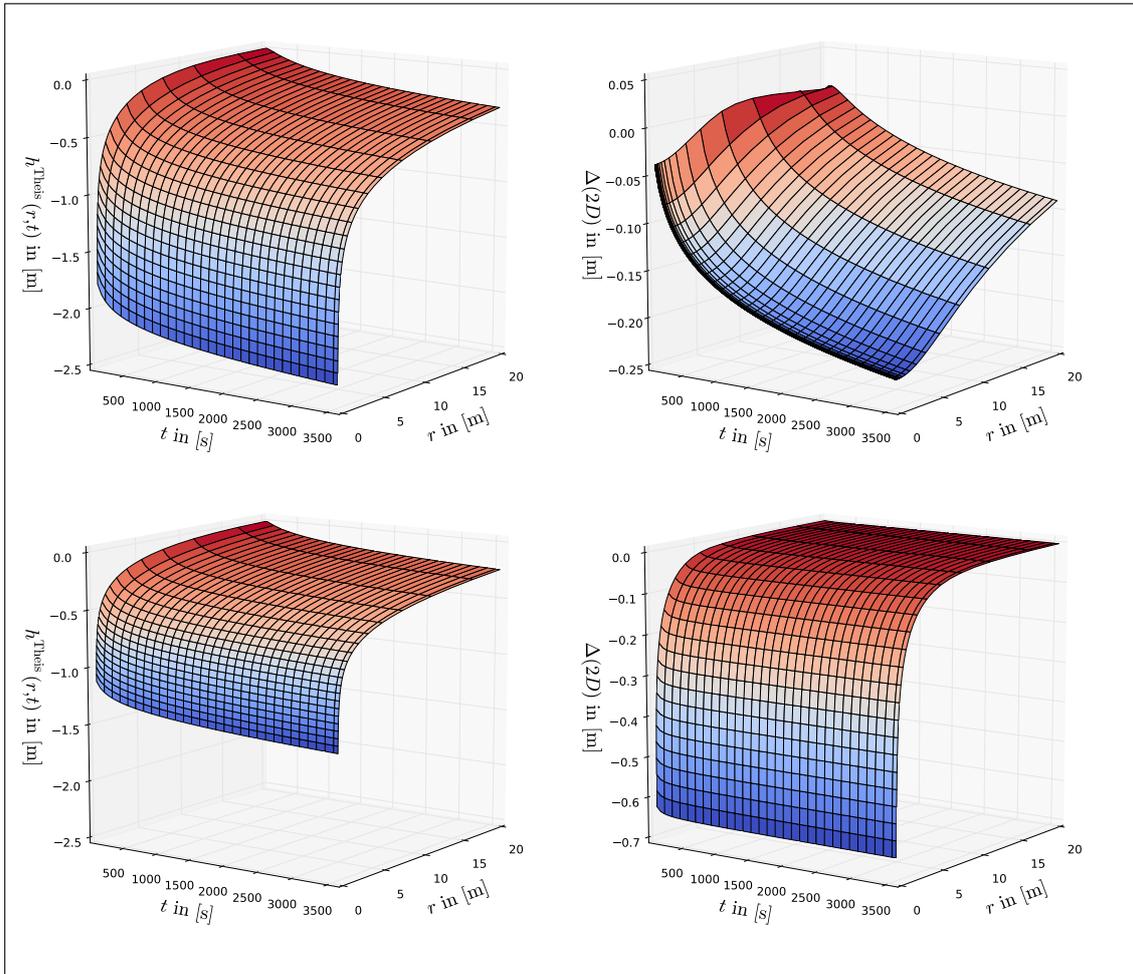


Abbildung 21: Vergleich der Theis-Lösung mit der 2D-Lösung für Standardwerte (vgl. Abbildung 10). Dabei wurde zum einen $T = T_H$ gewählt (oben links) und die Differenz $\Delta(2D)$ zur 2D-Lösung berechnet (oben rechts). Zum anderen wurde $T = T_G$ gewählt (unten links) und ebenfalls die Differenz $\Delta(2D)$ zur 2D-Lösung berechnet (unten rechts).

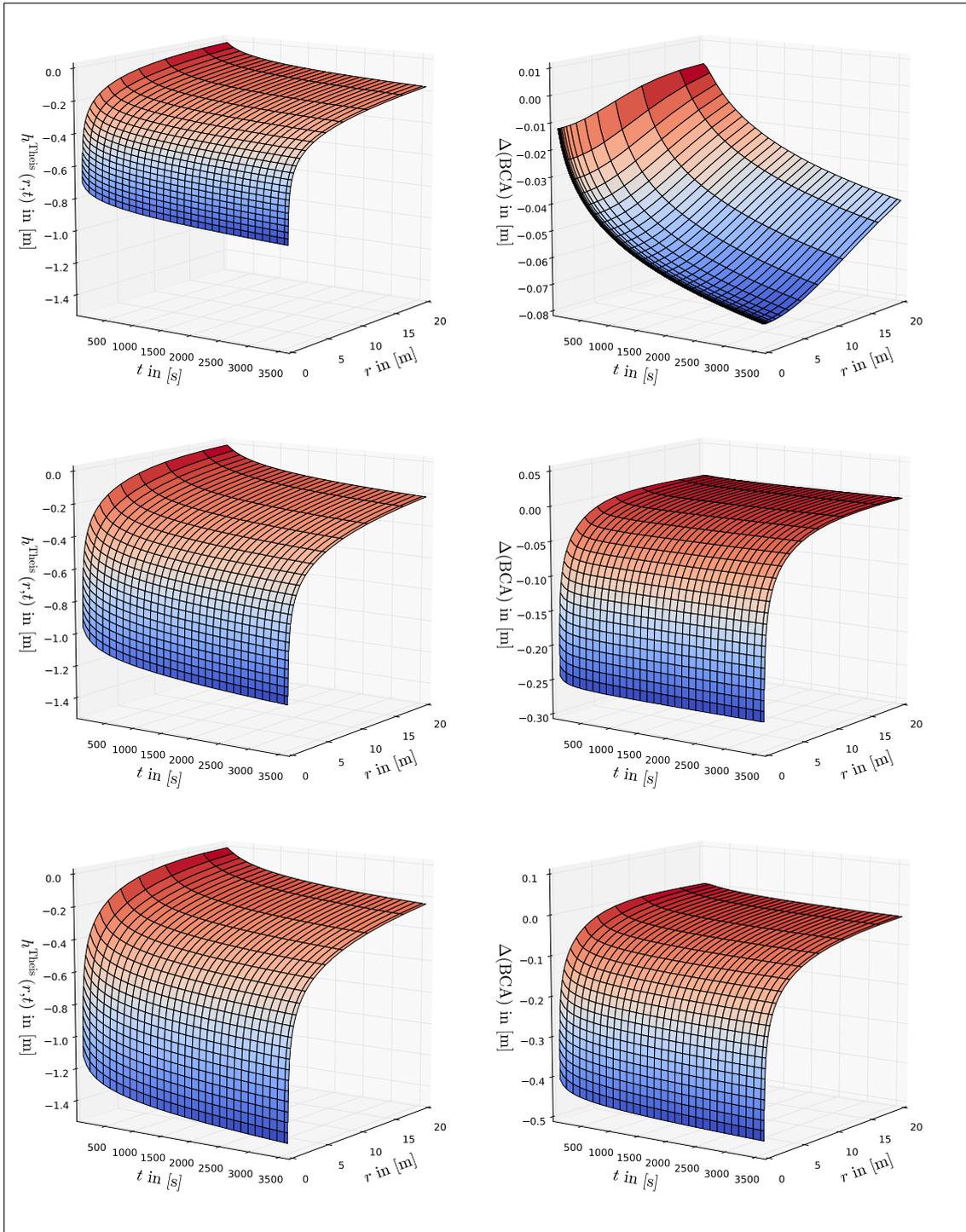


Abbildung 22: Vergleich der Theis-Lösung mit der 3D-Lösung im BCA-Fall für Standardwerte (vgl. Abbildung 10). Dabei wurde zum einen $K = K_A$ gewählt (oben links) und die Differenz $\Delta(\text{BCA})$ zur 3D-Lösung berechnet (oben rechts). Dann wurde $K = K_{\text{efu}}$ gewählt (mitte links) und die Differenz $\Delta(\text{BCA})$ zur 3D-Lösung berechnet (mitte rechts). Zuletzt wurde $K = K_G$ gewählt (unten links) und ebenfalls die Differenz $\Delta(\text{BCA})$ zur 3D-Lösung berechnet (unten rechts).

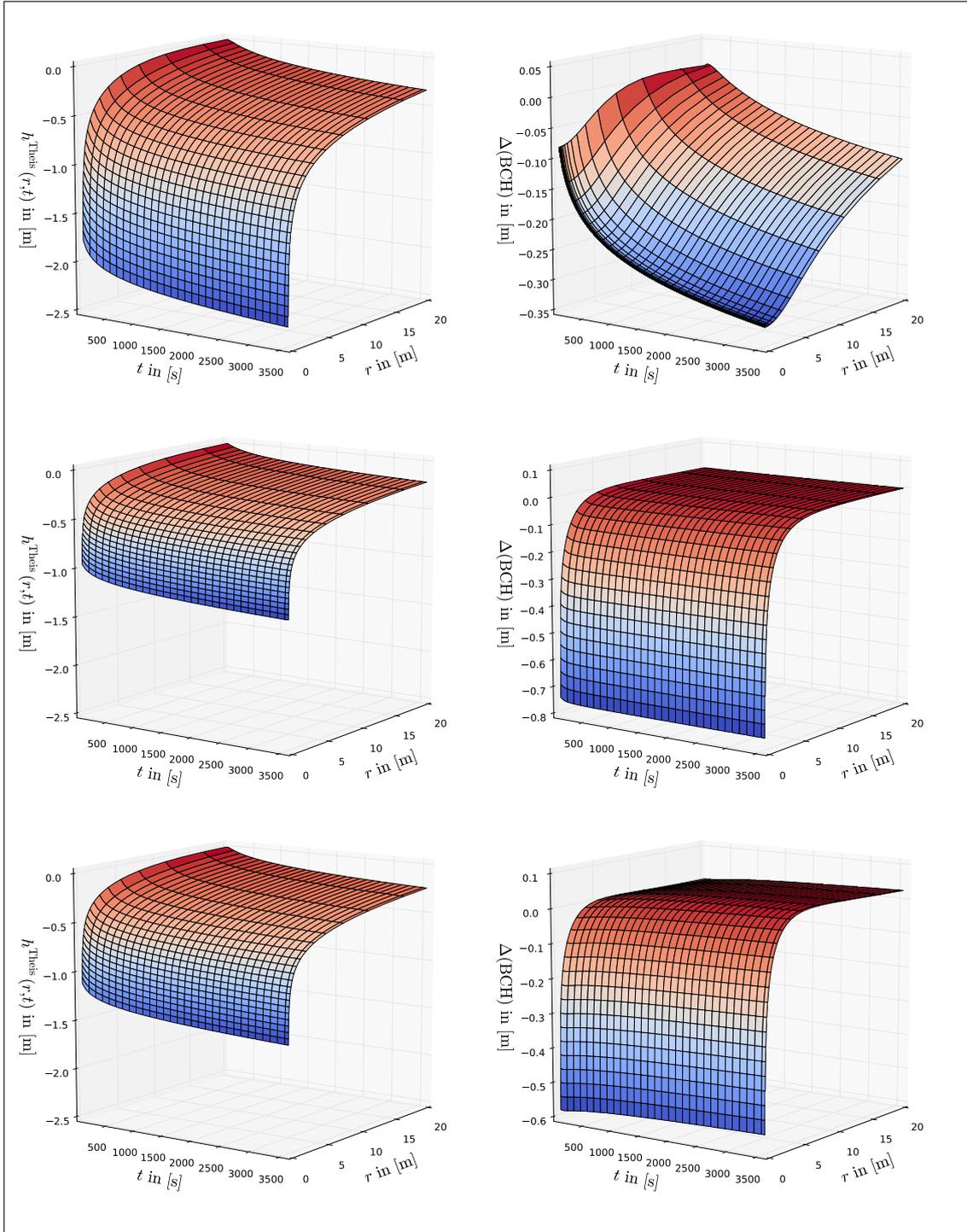


Abbildung 23: Vergleich der Theis-Lösung mit der 3D-Lösung im BCH-Fall für Standardwerte (vgl. Abbildung 10). Dabei wurde zum einen $K = K_H$ gewählt (oben links) und die Differenz $\Delta(\text{BCH})$ zur 3D-Lösung berechnet (oben rechts). Dann wurde $K = K_{\text{efu}}$ gewählt (mitte links) und die Differenz $\Delta(\text{BCH})$ zur 3D-Lösung berechnet (mitte rechts). Zuletzt wurde $K = K_G$ gewählt (unten links) und ebenfalls die Differenz $\Delta(\text{BCH})$ zur 3D-Lösung berechnet (unten rechts).

Symbolverzeichnis

Symbol	Bedeutung	Einheit
\mathbb{R}	die reellen Zahlen	
\mathbb{N}	die natürlichen Zahlen	
\mathbb{C}	die komplexen Zahlen	
$\Re(z)$	Realteil einer komplexen Zahl z	
$\Im(z)$	Imaginärteil einer komplexen Zahl z	
ζ	Proportionalitätsfaktor	
φ	Dichtefunktion	
μ	Mittelwert der log-Leitfähigkeit	
σ^2	Varianz der log-Leitfähigkeit	
ℓ	horizontale Korrelationslänge	[m]
ℓ_z	vertikale Korrelationslänge	[m]
Var	Varianz	
CV	Kovarianzfunktion	
e	Anisotropierate	
$\gamma(e)$	Anisotropiefunktion	
K_f	Durchlässigkeitsbeiwert	$[\text{m} \cdot \text{s}^{-1}]$
K_A	arithmetisches Mittel der Leitfähigkeit	$[\text{m} \cdot \text{s}^{-1}]$
K_G	geometrisches Mittel der Leitfähigkeit	$[\text{m} \cdot \text{s}^{-1}]$
K_H	harmonisches Mittel der Leitfähigkeit	$[\text{m} \cdot \text{s}^{-1}]$
K_{efu}	effektive Leitfähigkeit für uniformen Fluss	$[\text{m} \cdot \text{s}^{-1}]$
K_{well}	effektive Leitfähigkeit am Brunnen	$[\text{m} \cdot \text{s}^{-1}]$
T_G	geometrisches Mittel der Durchlässigkeit	$[\text{m}^2 \cdot \text{s}^{-1}]$
T_H	harmonisches Mittel der Durchlässigkeit	$[\text{m}^2 \cdot \text{s}^{-1}]$
$K(\underline{x})$	Leitfähigkeitsverteilung	$[\text{m} \cdot \text{s}^{-1}]$
$T(\underline{x})$	Durchlässigkeitsverteilung	$[\text{m}^2 \cdot \text{s}^{-1}]$
$K^{\text{CG}}(r)$	Coarse-Graining-Leitfähigkeit	$[\text{m} \cdot \text{s}^{-1}]$
$T_{\text{H}}^{\text{CG}}(r)$	Coarse-Graining-Durchlässigkeit	$[\text{m}^2 \cdot \text{s}^{-1}]$
χ	Abkürzung für $\ln \frac{K_{\text{efu}}}{K_{\text{well}}}$	
h	Standrohrspiegelhöhe	[m]
\underline{q}	Darcyscher Geschwindigkeitsvektor	$[\text{m} \cdot \text{s}^{-1}]$
L	vertikale Stärke des Wasserleiters	[m]
Γ_{well}	Mantelfläche des Brunnen	
Γ_{∞}	äußere Mantelfläche des Wasserleiters	
Q_{well}	Pumprate am Brunnen	$[\text{m}^3 \cdot \text{s}^{-1}]$
S	spezifischer Speicherkoeffizient	$[\text{m}^3 \cdot \text{kg}^{-1}]$
r	radialer Abstand zum Brunnen	[m]
t	Zeit	[s]
$\mathcal{L}\{f\}$	Laplacetransformierte der Funktion f	
s	Variable im Laplaceraum	
s_k	Konvergenzabszisse der Laplacetransformation	
s_a	Abszisse absoluter Konvergenz der Laplacetr.	
$g_s(r)$	Laplacetransformierte der Funktion $h(r, t)$	
r_{well}	Radius des Brunnen	[m]
r_{∞}	radiale Ausdehnung des Wasserleiters	[m]

Symbol	Bedeutung	Einheit
r_{\min}	unterer Radius zur numerischen Integration	[m]
r_{\max}	oberer Radius zur numerischen Integration	[m]
r_{\lim}	obere Grenze für r_{\max}	[m]
$\varepsilon_{\text{error}}$	Fehler der Leitfähigkeit im Fernfeld	
$E_1(x)$	Integralexponentialfunktion	
$I_\lambda(x)$	erste modifizierte Besselfunktion der Ordnung λ	
$K_\lambda(x)$	zweite modifizierte Besselfunktion der Ordnung λ	
T_n	Menge der Partitionen einer Zahl n	
$P(n)$	Anzahl der Partitionen von n	
C	Abkürzung für $\frac{\zeta}{\ell}$ in $2D$ und $\frac{\zeta}{\sqrt[3]{\ell}}$ in $3D$	
τ	Abkürzung für $C \cdot r$	
α, β	Koeffizientenfunktion der Differentialgleichung	
$I(\nu)$	indiziales Polynom	
ν_1, ν_2	charakteristische Exponenten	
T_s	Abkürzung für $\sqrt{\frac{s \cdot S}{K}}$	
γ	Euler-Mascheroni-Konstante	
H_k	Partialsomme der harmonischen Reihe	
A_s, B_s, C_s, D_s	Koeffizienten der Basisfunktionen im Laplaceraum	
I_1, I_2, I_3	Intervalle der Lösungsabschnitte	

Abbildungsverzeichnis

1.	Schema eines Pumpversuchs.	1
2.	Log-normal verteilte Zufallsfelder $K(x)$ mit Gaußscher Korrelationsstruktur	3
3.	Die Coarse-Graining-Durchlässigkeit und -Leitfähigkeit	4
4.	Die Anisotropiefunktion $\gamma(e)$	5
5.	Abfolge der Lösungsschritte	9
6.	Abhängigkeitsbaum der implementierten Routinen	55
7.	Vergleich des essentiellen Verlaufs der Koeffizientenfunktionen $\hat{\alpha}^{2D}$ und $\hat{\beta}^{2D}$	59
8.	Vergleich des essentiellen Verlaufs der Koeffizientenfunktionen $\hat{\alpha}^{3D}$, $\hat{\beta}^A$ und $\hat{\beta}^H$	60
9.	Vergleich der Basisfunktionen mit den modifizierten Besselfunktionen	61
10.	Plots der Lösungen für den Standardparametersatz	65
11.	Fehleranalyse der 2D-Lösung zu den Parametern $\sigma^2 = 0.5$ und $\ell = 5$ sowie $\sigma^2 = 0.5$ und $\ell = 10$	90
12.	Fehleranalyse der 2D-Lösung zu den Parametern $\sigma^2 = 2$ und $\ell = 5$ sowie $\sigma^2 = 2$ und $\ell = 10$	91
13.	Fehleranalyse der 3D-Lösung im BCA-Fall zu den Parametern $\sigma^2 = 0.5$ und $\ell = 5$ sowie $\sigma^2 = 0.5$ und $\ell = 10$	92
14.	Fehleranalyse der 3D-Lösung im BCA-Fall zu den Parametern $\sigma^2 = 2$ und $\ell = 5$ sowie $\sigma^2 = 2$ und $\ell = 10$	93
15.	Fehleranalyse der 3D-Lösung im BCH-Fall zu den Parametern $\sigma^2 = 0.5$ und $\ell = 5$ sowie $\sigma^2 = 0.5$ und $\ell = 10$	94
16.	Fehleranalyse der 3D-Lösung im BCA-Fall zu den Parametern $\sigma^2 = 2$ und $\ell = 5$ sowie $\sigma^2 = 2$ und $\ell = 10$	95
17.	Einfluss der Anisotropierate e auf die 3D-Lösung im BCA-Fall	96
18.	Einfluss der Anisotropierate e auf die 3D-Lösung im BCH-Fall	97
19.	Einfluss der Varianz σ^2 auf die 2D-Lösung	98
20.	Einfluss der Korrelationslänge ℓ auf die 2D-Lösung	99
21.	Vergleich der Theis-Lösung mit der 2D-Lösung	100
22.	Vergleich der Theis-Lösung mit der 3D-Lösung im BCA-Fall	101
23.	Vergleich der Theis-Lösung mit der 3D-Lösung im BCH-Fall	102

Liste der Programmcodes

1.	Implementierung der Regel von Faà di Bruno	51
2.	Ablauf der Funktion <code>ext_theis2d</code>	54
3.	Ablauf der Funktion <code>ext_theis3d</code>	55
4.	Ablauf der Funktion <code>lap_ext_theis[2d/3d]</code>	56
5.	Implementierung der Funktion <code>nextpart</code>	71
6.	Implementierung der Funktion <code>isgoodpart</code>	72
7.	Implementierung des Stehfest-Algorithmus <code>NLInvSteh</code>	73
8.	Implementierung der Routine <code>csteh</code>	74
9.	Implementierung der Funktion <code>ext_theis2D</code>	75
10.	Implementierung der Funktion <code>lap_ext_theis2D</code>	77
11.	Implementierung der Funktion <code>ext_theis3D</code>	82
12.	Implementierung der Funktion <code>lap_ext_theis3D</code>	85

Literatur

- [Abramowitz et al., 1972] Abramowitz, M., Stegun, I. A., et al. (1972). *Handbook of mathematical functions*, volume 1. Dover New York.
- [Cheng et al., 1994] Cheng, A. H., Sidauruk, P., und Abousleiman, Y. (1994). Approximate inversion of the laplace transform. *Mathematica Journal*, 4(2):76–82.
- [Delleur, 1999] Delleur, J. W., editor (1999). *The handbook of groundwater engineering*. Springer Science & Business Media.
- [Fuchs, 1866] Fuchs, L. I. (1866). Zur theorie der linearen differentialgleichungen mit veränderlichen coefficienten. *Journal für die reine und angewandte Mathematik*, 66:121.
- [Gelhar, 1993] Gelhar, L. W. (1993). *Stochastic subsurface hydrology*. Prentice-Hall.
- [Großmann, 2005] Großmann, C. (2005). *Numerische Behandlung partieller Differentialgleichungen*. Teubner.
- [Häfner et al., 1992] Häfner, F., Sames, D., und Voigt, H.-D. (1992). *Wärme- und Stofftransport : Mathematische Methoden*. Springer, Berlin.
- [Heße et al., 2014] Heße, F., Prykhodko, V., Schlüter, S., und Attinger, S. (2014). Generating random fields with a truncated power-law variogram: A comparison of several numerical methods. *Environmental Modelling & Software*, 55:32–48.
- [Matoušek, 2007] Matoušek, Jiří, N. J. (2007). *Diskrete Mathematik Eine Entdeckungsreise*. Springer Berlin Heidelberg.
- [Press et al., 1992] Press, W. H., Teukolsky, S. A., Vetterling, W. T., und Flannery, B. P. (1992). *Numerical recipes in FORTRAN 77, vol. 1*. Press Syndicate of the University of Cambridge.
- [Rudin, 1999] Rudin, W. (1999). *Reelle und komplexe Analysis*, volume 3. Oldenbourg.
- [Schneider und Attinger, 2008] Schneider, C. und Attinger, S. (2008). Beyond thiem: A new method for interpreting large scale pumping tests in heterogeneous aquifers. *Water resources research*, 44(4).
- [Stehfest, 1970] Stehfest, H. (1970). Algorithm 368: Numerical inversion of laplace transforms [d5]. *Communications of the ACM*, 13(1):47–49.
- [Strehmel, 1995] Strehmel, K. (1995). *Numerik gewöhnlicher Differentialgleichungen : Mit zahlreichen Beispielen und Übungsaufgaben*. Teubner.
- [Teschl, 2012] Teschl, G. (2012). *Ordinary differential equations and dynamical systems*. American Mathematical Soc.
- [Theis, 1935] Theis, C. V. (1935). *The relation between the lowering of the piezometric surface and the rate and duration of discharge of a well using ground water storage*. US Department of the Interior, Geological Survey, Water Resources Division, Ground Water Branch Washington, DC.
- [Timmann, 2010] Timmann, S. (2010). *Repetitorium der gewöhnlichen Differentialgleichungen*. Binomi.
- [Weber und Ulrich, 2007] Weber, H. und Ulrich, H. (2007). *Laplace-Transformation: Grundlagen-Fourierreihen und Fourierintegral-Anwendungen*. Springer-Verlag.
- [Zech, 2013] Zech, A. (2013). *Impact of Aqifer Heterogeneity on Subsurface Flow and Salt Transport at Different Scales: from a method determine parameters of heterogeneous permeability at local scale to a large-scale model for the sedimentary basin of Thuringia*. PhD thesis, Friedrich-Schiller-Universität Jena.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als angegeben verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Ort:

Datum:

Unterschrift: