Entwicklung und Validierung eines modularen Simulationswerkzeuges für hydrogeologische und geothermische Fragestellungen mit einer interaktiven Schnittstelle zur direkten Implementierung dynamischer und rückkoppelnder Randbedingungen – ModSimple

Phase 2

Abschlussbericht über ein Entwicklungsprojekt, gefördert unter dem Az: DBU-AZ 37733/01 von der Deutschen Bundesstiftung Umwelt

Autoren: Dr.-Ing. Mike Müller
Lúcia Pedrosa, M.Sc.
Dr.-Ing. Martin Binder

Wissenschaftlicher Beirat: Prof. Dr. Traugott Scheytt
Dr.-Ing. Falk Händel
PD Dr. habil. Christoph Neukum
PD Dr. habil. Jannis Epting
Dr. Peter Börke

August 2025

06/02

Projektkennblatt
der
Deutschen Bundesstiftung Umwelt

Az	37733/01	Referat	23	Fördersumme	124.874 €

Antragstitel	**Entwicklung und Validierung eines Modularen Simulationswerkzeuges für hydrogeologische und geothermische Fragestellungen mit einer interaktiven Schnittstelle zur direkten Implementierung dynamischer und rückkoppelnder Randbedingungen {ModSimple – Phase 2}**
Stichworte	Simulation, Wasser/Gewässer

Laufzeit	Projektbeginn	Projektende	Projektphase(n)
30 Monate	**01.10.2022**	**31.03.2025**	**2**

Zwischenberichte

Bewilligungsempfänger	hydrocomputing GmbH & Co. KG Zur Schule 20 04158 Leipzig	Tel 0341 525 599 54 Fax 0341 520 4495 Projektleitung Dr.-Ing. Mike Müller Bearbeiter Dr.-Ing. Mike Müller
Kooperationspartner	TU Bergakademie Freiberg (Lehrstuhl für Hydrogeologie und Hydrochemie) Gustav-Zeuner-Str.12 09599 Freiberg	

Zielsetzung und Anlass des Vorhabens

Dieses Vorhaben führt die Arbeiten der Phase 1 des Projektes "*Entwicklung und Validierung eines modularen Simulationswerkzeuges für hydrogeologische und geothermische Fragestellungen mit einer interaktiven Schnittstelle zur direkten Implementierung dynamischer und rückkoppelnder Randbedingungen {ModSimple}" weiter.* Das Simulationswerkzeug soll für geothermische Anwendungen weiterentwickelt, validiert und auf einen Standort angewendet werden.

Darstellung der Arbeitsschritte und der angewandten Methoden

Die Arbeiten erfolgten in fünf Arbeitspaketen (AP). AP 1.1 umfasste die Entwicklung eines Simulationswerkzeuges zur numerischen Simulation von Stoff- und Wärmetransport. Der Projektpartner (PP) hydrocomputing erweiterte das ueflow-Modul pymf6 aus Phase 1 durch Integration des Groundwater Energy (GWE) Model von MODFLOW 6 (MF6) weiter. Der Austausch mit MF6 wurde von der Nutzung des Memory-Managers auf das Basic Model Interface (BMI) umgestellt. In AP 1.2 erfolgte die Validierung der Implementierung anhand synthetischer Testszenarien. Der PP hydrocomputing entwickelte einen automatisierten Test-Workflow für die vom United States Geological Survey (USGS) bereitgestellten 155 Test-Modelle. AP 2.1 beinhaltete die Kopplung von analytischen Randbedingungen mit den numerischen Strömungs- sowie Stoff- und Wärmetransportmodulen. Der PP TU Bergakademie Freiberg (TUBAF) setzte eine analytische Lösung für den Wärmeübergang aus der Atmosphäre um. Der PP hydrocomputing entwickelte eine Kopplung mit einem Modell der Analytic Element Method (AEM). Ein praxisnaher Anwendungsfall zur Evaluierung der Anwendung des Werkzeuges auf das geothermische Modell Kleinbasel war Gegenstand des AP 2.2. Der PP TUBAF exportierte die Daten des FEFLOW-Modells von Kleinbasel in geeignete Formate. Der PP hydrocomputing entwickelte ein Konvertierungswerkzeug zur Umwandlung dieser Daten in Eingabedaten für MF6 und fügte dem Modell eine pymf6-basierte thermische Cauchy-Randbedingung hinzu. In AP 3 entstanden eine Programmdokumentation und ein Schulungskonzept. Beide Projektpartner entwickelten dafür geeignete Beispielanwendungen von pymf6.

Ergebnisse und Diskussion

Das Ergebnis der Phase 2 ist das entscheidend verbesserte Werkzeug pymf6 aus Phase 1 dieses Projektes. Das technische Konzept integriert jetzt die in der Zwischenzeit neu entstandenen Bibliotheken, wie das BMI-basierte Python-Modul modflowap*i*. Dies vereinfacht die Nutzbarkeit von pymf6 durch eine höhere Abstraktionsebene bei der Interaktion enorm. Die Kopplung mit analytischen Lösungen und die Anwendung auf das komplexe Modell Kleinbasel zeigen das breite Anwendungsspektrum von pymf6. Die Entwicklungsstrategie wurde im Projekt mehrfach an neue technische Entwicklungen angepasst. Der Fokus hat sich dabei von der technischen Integration über Programmiersprachengrenzen hinweg zur Praktikabilität des Werkzeuges für die Nutzung durch Hydrologen und Ingenieure verschoben. Die weitere Entwicklung wird durch das Einarbeiten des Feedbacks der sich ausweitenden Nutzergemeinde wahrscheinlich weiter in diese Richtung gehen. Die geplanten Schulungen spielen hierbei eine zentrale Rolle, um direktes Feedback zu erhalten.

Öffentlichkeitsarbeit und Präsentation

Projektpartner präsentierten Zwischenergebnisse mit Vorträgen und Posterbeiträgen auf den ASIM-Workshops 2023 und 2024, der FH-DGGV-Konferenz 2024, der Konferenz MODFLOW and More 2024 sowie der EuroSciPy 2023. Dabei kam es zu einem Gedankenaustausch zu unterschiedlichsten Aspekten der Simulation- und Softwaretechnik. Insbesondere der persönliche Austausch mit den Kernentwicklern von MODFLOW 6 und modflowapi auf der MODFLOW and More erbrachte viele neue Ansatzpunkte für die Strategie der Entwicklung von pymf6. Erste Ergebnisse wurden in einer Publikation veröffentlicht (Müller et al, 2023). Weitere Publikationen sind in Vorbereitung.

Fazit

Das Projekt konnte erfolgreich abgeschlossen werden. Die grundsätzliche Herangehensweise hat sich bewährt. Die Arbeiten in Phase 2 konnten auf denen der Phase 1 aufbauen. Gleichzeitig konnten grundlegend neue Konzepte aufgenommen und erfolgreich umgesetzt werden. Als nächste Schritte stehen die Bekanntmachung des Projektes und der Praxis-Einsatz des Werkzeugs an.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Verzeichnis von Begriffen, Abkürzungen und Definitionen

Begriffe

MODFLOW 6
: weit verbreite Software für die Grundwassermodellierung, vom USGS entwickelt und kostenfrei vertrieben

Abkürzungen

BMI
: Basic Model Interface (https://bmi.readthedocs.io/en/stable/)

bmipy
: BMI for Python (https://github.com/csdms/bmi-python)

MF6
: MODFLOW 6

modflowapi
: Erweiterung von xmipy für MODFLOW (https://github.com/MODFLOW-USGS/modflowapi)

pymf6
: Python-Interface zu MODFLOW 6, in diesem Projekt entwickelt

RB
: Randbedingung

ueflow
: **U**ser **E**xtensible **Flow** Model

xmipy
: Erweiterung von bmipy für hydrologische Modelle (https://github.com/Deltares/xmipy)

1. Zusammenfassung

Das Projekt *ModSimple* entwickelt benutzerfreundliche Software-Werkzeuge zur Abbildung von Grundwasserströmung sowie Stoff- und Wärmetransport. Eine wichtige Anwendung dieser Werkzeuge ist die effiziente Bewertung von geothermischen Maßnahmen in urbanen Räumen. Die hier vorgestellte Phase 2 schließt das Gesamtprojekt erfolgreich ab.

Im Rahmen der Phase 2 dieses Projektes wurden:

- das in Phase 1 dieses Projektes entwickelte `pymf6` entscheidend weiterentwickelt, sodass nun der Wärmetransport unterstützt wird und die programmtechnischen Nutzungsmöglichkeiten stark erweitert werden konnten,
- `pymf6` gegen mehr als 150 Modelle für unterschiedlichste Fragestellungen validiert,
- `pymf6` mit analytischen Modellen gekoppelt, um die Funktionalität von MODFLOW 6 (MF6) zu erweitern,
- das komplexe geothermische Grundwassermodell Kleinbasel von FEFLOW nach MF6 konvertiert und `pymf6` um eine thermische Cauchy-Randbedingung erweitert,
- eine umfangreiche Programmdokumentation mit Beispielen für verschiedene Anwendungsfälle erstellt und
- ein Schulungskonzept erarbeitet, das für den ersten Workshop im Oktober 2025 zum Einsatz kommt

Die Komponente `pymf6` von `ueflow` (user-extensible flow model) erlaubt es Nutzern, MODFLOW-6-Modelle für Strömungs-, Stofftransport- und Wärmetransport-Fragestellungen um entscheidende Funktionalität zu erweitern. Damit steht ein leistungsfähiges Werkzeug für die bessere Abbildung geothermischer Prozesse im urbanen Raum für die Planungen und Nutzung von insbesondere kleineren, urbanen Geothermieanlagen zur Verfügung. Dies eröffnet zahlreiche Umweltentlastungspotenziale.

Das Werkzeug `pymf6` ist Open-Source-Software. Daher ist es sowohl für die Lehre als auch für kommerzielle Anwendungen nutzbar. Schulungen, Workshops und Vorträge auf Konferenzen sollen `pymf6` bekannt machen und seine Möglichkeiten zeigen. Das Feedback der Nutzer soll in die weitere Entwicklung einfließen, um das Werkzeug kontinuierlich zu verbessern. Die kommerzielle Verwertung fokussiert auf Schulungen und kundenspezifische Entwicklung von `pymf6`-basierten MF6-Erweiterungen.

Dieses Projekt hat die Deutsche Bundesstiftung Umwelt unter dem Aktenzeichen 37733/01 gefördert. Die durchführenden Partner waren die hydrocomputing GmbH & Co. KG, Leipzig (Bewilligungsempfänger) und die TU Bergakademie Freiberg, Lehrstuhl für Hydrogeologie und Hydrochemie (Kooperationspartner).

2. Einleitung

Urbane Räume sind hinsichtlich ihres Stoff- und Wärmeausbreitungsverhaltens im Untergrund sehr komplex. Betrachtet man die Hydrologie und Hydrogeologie solcher Ballungsräume auf holistischer Skala, so können diese in die folgenden Teilkompartimente unterteilt werden: die atmosphärische Grenzschicht, die Erdoberfläche, die variabel wassergesättigte Bodenzone (vadose Zone) sowie die Grundwasserzone. Zwischen diesen Kompartimenten existieren bidirektionale Wechselwirkungsprozesse hinsichtlich Wärme- und Stoffausbreitung (siehe Abbildung 2.1), insbesondere die vadose Zone und die Grundwasserzone stehen hier im Fokus.

Die vadose Zone leitet als Transitzone die Wettereinflüsse der Atmosphäre (wie Niederschlag und Temperatur) zum Grundwasser weiter. In gegensätzlicher Richtung findet unter anderem Verdunstung statt. Natürliche wie auch anthropogene Wärme- und (ungewollte) Stoffeinträge breiten sich in vertikaler Richtung aus; mit zunehmender Bodentiefe werden diese infolge der Eigenschaften des hydrogeologischen Untergrundes teilweise abgeschwächt. Künstliche Systeme wie Kanalisation und Trinkwasserverteilernetze stellen je nach Zustand erhebliche Quellen und Senken dar. Gefahren für das Schutzgut Grundwasser ergeben sich hierbei beispielsweise durch den Eintrag von hoch-persistenten organischen Schadstoffen. In der Grundwasserzone selbst findet eine vorrangig horizontale Fließbewegung des unterirdischen Wassers statt. In Abhängigkeit der hydraulischen Gradienten sowie der hydrogeologischen Stratigrafie werden gelöste Stoffe advektiv-dispersiv mit der Strömung transportiert (und damit teilweise großräumig verteilt), an der Gesteinsmatrix adsorbiert sowie teilweise chemisch-biologisch abgebaut oder anderweitig umgewandelt. Analog zum Transport von Stoffen findet der Transport von Wärme statt, welcher zusätzlich von den Wärmeleitungseigenschaften des Gesteins abhängt. In Ergänzung zum natürlichen, saisonal oft fluktuierenden Eingangssignal führt der Betrieb geothermischer Anlagen zur Ausbildung erheblicher Temperaturanomalien im Grundwasser.

Aus einer möglichen negativen Beeinflussung der Wassergüte des Grundwassers, insbesondere hervorgerufen durch die anthropogenen Komponenten, kann eine Vielzahl an Nutzungskonflikten hervorgehen. Ein nachhaltiges, aber dynamisches Management dieser Ressource ist dringend nötig, nicht nur bezüglich des thermischen Haushalts,

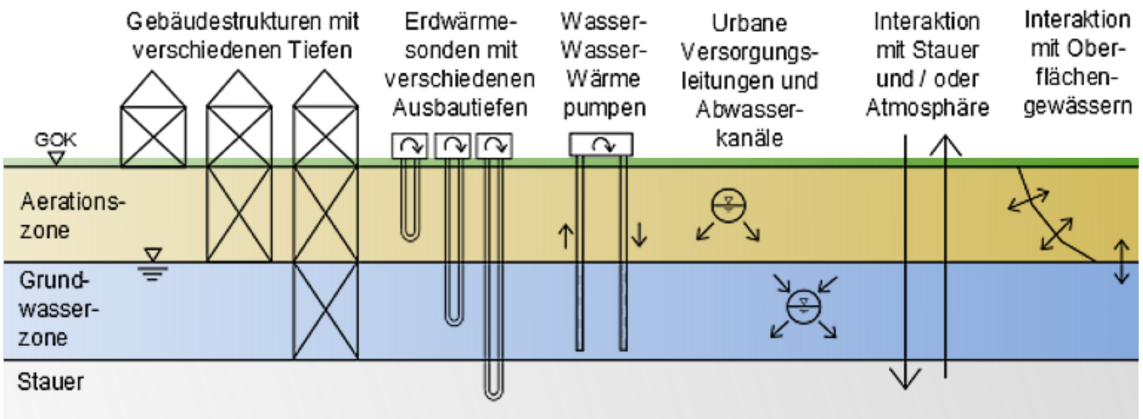

Abbildung 2.1.: Geothermische Randbedingungen als Beispiel für die vielfältigen Austauschprozesse in urbanen Räumen (Köhler et al., 2015)

sondern auch für weitere qualitative und quantitative Parameter. Dies wiederum setzt das Vorhandensein adäquater und an die Anforderungen anpassbarer Bewertungswerkzeuge voraus.

In diesem Kontext war die Aufgabe des Projektes *ModSimple*, ein geeignetes, modular erweiterungsfähiges Softwaresystem zur Modellierung der Grundwasserströmung sowie des Stoff- und Wärmetransports zu entwickeln und zu validieren. Diese Modellumgebung soll es effizient ermöglichen, anthropogene wie auch natürliche Einflüsse auf die Umweltressource Grundwasser dynamisch, teilweise unter Verwendung von Approximationslösungen, zu modellieren.

Eine der Innovationen von *ModSimple* stellt an dieser Stelle die vorgesehene enge, aber zugleich dynamische Kopplung zwischen einem frei verfügbaren, numerischen Strömungs-, Stoff- und Wärmetransportmodell-Framework (MODFLOW (USGS, 2020), MT3D (Bedekar et al., 2016)) und, umgesetzt in der Skriptsprache Python, ausgewählter technischer Randbedingungen dar. Realisiert wird dies durch die Schaffung einer interaktiven Schnittstelle, welche es dem Nutzer ermöglichen soll, komplexe Randbedingungen benutzerdefiniert in bestehende Modelle zu integrieren. Bei bestehenden, teilweise auch sehr teuren Softwarelösungen, ist dies oftmals nicht möglich oder mit hohen Anforderungen an Programmierkenntnisse verbunden. Die Software soll zudem unter einer Open-Source-Lizenz stehen, um deren Verbreitung und Nutzerkreis zu erweitern.

Das Projekt *ModSimple* liedert sich hierbei in zwei Bearbeitungsphasen. Phase 1 mit dem Titel *Entwicklung und Validierung eines modularen Simulationswerkzeuges für hydrogeologische und geothermische Fragestellungen mit einer interaktiven Schnittstelle zur direkten Implementierung dynamischer und rückkoppelnder Randbedingungen – ModSimple* wurde im Rahmen des vorherigen Bewilligungszeitraumes von 01/2018 bis 06/2020 bearbeitet (Müller et al., 2020).

Die hier beschriebene Phase 2 führt die Phase 1 weiter, baut auf den Ergebnissen der Phase 1 auf und entwickelt die darin erstellten Werkzeuge und Konzepte weiter. In Phase 1 ist das Werkzeug `pymf6` als Bestandteil von `ueflow` entstanden. Die Open-Source-Software `pymf6` kann die Strömungsvorgänge im Untergrund mittels MODFLOW 6 abbilden. Es ermöglicht dabei eine Laufzeitinteraktion mit allen Modell-Variablen mit der Programmiersprache Python. Modell-Anwender können mit diesem System dynamisch, rückkoppelnde Randbedingungen umsetzen. Dabei kann es sich um sehr einfache Anwendungen handeln. Ein Beispiel ist die Anpassung der Werte für die Brunnenentnahme, um den Brunnenwasserstand in einem bestimmten Bereich zu halten. Aber auch komplexere Anwendungen wie die Kopplung an andere analytische oder numerische Modelle sind vergleichsweise einfach umsetzbar. So wäre zum Beispiel eine Kopplung an ein Niederschlags-Abfluss-Modell möglich, das Sickerwasser die Grundwasserneubildung als obere Randbedingung an das Grundwassermodell liefert und Grundwasser-Flurabstände vom Grundwassermodell als untere Randbedingung für den kapillaren Aufstieg erhält.

Das Hauptziel von Phase 2 war die Erweiterung der bestehenden Lösung auf Stoff- und Wärmetransport-Probleme und deren Anwendung auf ein urbanes Beispielgebiet. Nach der Validierung des Werkzeugs für diese Zwecke sollten einige der in Phase 1 aufbereiteten analytischen Lösungen als dynamische Randbedingung mittels `pymf6` eingebunden werden. Die Modelliersoftware wurde anhand eines realen Untersuchungsstandortes im urbanen Raum des Schweizer Kantons Basel-Stadt validiert. Hierbei erfolgte die

Implementierung einer eigenen, thermischen Cauchy-Randbedingung. Eine umfassende Dokumentation und Schulungen sollten zur Verbreitung des Werkzeuges führen.

Im nachfolgenden Kapitel 3 werden der hierfür notwendige Lösungsansatz von *ModSimple*, Phase 2, im Detail beschrieben und der Projektablauf gemäß der ursprünglichen Planung kurz dargestellt. Dem folgt eine auf die jeweiligen Arbeitspakete (AP) bezogene Darstellung der wichtigsten Arbeiten und Ergebnisse innerhalb des Projektes.

Das waren die geplanten Arbeitspakete:

- AP 1: Numerisches Transportmodell
- AP 2: Evaluierung anhand von Praxisbeispielen
- AP 3: Dokumentation und Schulungen

die in diesen Teil-AP bearbeitet wurden:

- AP 1.1 – Entwicklung der ueflow-Module zur numerischen Simulation von Stoff- und Wärmetransport (Abschnitt 3.1)
- AP 1.2 – Validierung der Implementierung anhand synthetischer Testszenarien (Abschnitt 3.2)
- AP 2.1 – Kopplung von analytischen Randbedingungen des Stoff- und Wärmetransports mit numerischen Stoff-Wärmetransportmodulen in ueflow (Abschnitt 3.3)
- AP 2.2 – Praxisnaher Anwendungsfall zur Evaluierung – Die Fallstudie Kleinbasel (Abschnitt 3.4)
- AP 3 – Erstellung einer Programmdokumentation und von Schulungensunterlagen (Abschnitt 3.5)

3. Hauptteil

3.1. AP 1.1 – Entwicklung der ueflow-Module zur numerischen Simulation von Stoff- und Wärmetransport

Bezeichnungen ueflow und pymf6
Zu Beginn von Phase 1 dieses Projektes wurde der Name `ueflow` für **u**ser **e**xtensible **flow** gewählt. `pymf6` ist ein Bestandteil von `ueflow`. Die Arbeiten haben sich bisher auf `pymf6` fokussiert. Die ursprünglichen Überschriften der Arbeitspakete wurden aber beibehalten.

3.1.1. Abbildung des Wärmetransports in MODFLOW 6

Groundwater Energy Transport (GWE) Model

Die Bibliothek *pymf6* wurde um die Komponenten Stoff- und Wärme-Transport erweitert. MF6 beinhaltet das Groundwater Transport Model (GWT). Damit lassen sich Stofftransport-Prozesse im Grundwasser im Modell abbilden. Geplant war, den Wärmetransport mit Hilfe von GWT und passenden Ersatz-Parametern abzubilden. Dieser Ansatz ist etabliert. So haben (Ma & Zheng, 2010) und (Hecht-Méndez et al., 2010) Stofftransport-Modelle für die Wärmetransport-Modellierung genutzt, indem sie die Parameter für die molekulare Diffusion als Ersatz-Parameter für die thermische Leitfähigkeit des Grundwasserleiters eingesetzt haben. Ursprünglich war es geplant, diesen Ansatz zu nutzen und Werkzeuge zur Parameter-Übersetzung in *pymf6* zu integrieren.

Seit Version 6.5, die im Mai 2024 veröffentlicht wurde, enthält MF6 ein numerisches Modell für den Wärmetransport. Dieses Groundwater Energy Transport (GWE) macht die Übersetzung von Parametern überflüssig, da Wärmetransport-Parameter direkt als Input dienen. Anstatt von Stoffkonzentrationen wie in GWT nutzt GWE die Temperatur als Simulations-Variable. GWE löst die Wärmetransport-Gleichung numerisch mit der Methode der finiten Volumen. So wie die Module für Strömung (Groundwater Flow, GWF) und den Stofftransport (Groundwater Transport, GWT) unterstützt GWE regelmäßige und unregelmäßige Grids. Weiterhin nutzt es auch den Algorithmus „Newton flow formulation“, XT3D (Provost et al., 2017), um die Fehler bei der Abbildung von Strömungen, die nicht den drei räumlichen Hauptachsen in x-, y-, z-Richtung entsprechen, zu minimieren. Mehrere GWF-, GWT- und GWE-Modelle können in einer Simulation zusammen laufen. GWE berechnet die Änderungen der Grundwasser-Temperatur in Raum und Zeit. Dabei bildet es dieses Prozesse ab:

- den konvektiven und advektiven Transport von Wärme mit dem strömenden Grundwasser
- die kombinierten hydrodynamischen Dispersionsprozesse der geschwindigkeitsabhängigen mechanischen Dispersion und Leitung analog zur Diffusion
- das thermische Gleichgewicht mit der Matrix des Grundwasserleiters
- die Mischung des Grundwassers mit Wässern aus Quellen und Senken
- die direkte Zugabe von thermischer Energie

Im GWE gilt die Modellannahme des sofortigen Temperaturausgleichs zwischen flüssiger und fester Phase. Das führt zu Retardationseffekten von Wärme- oder Kältefronten. In GWE gibt es, im Unterschied zum GWT, keine immobile Phase. Der Energieaustausch zwischen Grundwasser und Festphase erscheint als ein Term im Energie-Budget. GWE ist, so wie GWF und GWT, einheitenlos. Der Modell-Anwender muss daher konsistente Einheiten für alle Eingaben nutzen. Eine typische Einheit für Energieflüsse ist Joule. Ein wichtiger Vorteil von GWE gegenüber GWT mit Parameter-Übersetzung ist die Wärmeleitung in der Feststoffmatrix, die auch in trockenen Zellen aktiv ist und damit Wärmeeffekte in der ungesättigten Zone abbilden kann.

3.1.2. Softwarearchitektur

Bisherige programmtechnische Umsetzung

In Phase 1 dieses Forschungsprojektes (Müller et al., 2020) erfolgte der Python-Laufzeit-Zugriff auf die Fortran-Variablen von MODFLOW 6 (MF6) über den *memory manager* von Python aus. Die in Phase 1 entstandene Bibliothek `pymf6` bindet den Fortran-Quelltext von MF6 als eine sogenannte Python-Erweiterung ein. Dies erlaubt, von Python aus alle Variablen des *memory manager* zu jedem Zeitschritt auszulesen und zu modifizieren.

Die programmtechnische Umsetzung nutzt das Werkzeug *f2py*, um den Fortran-Quelltext von MF6 in eine Python-Erweiterung umzuwandeln. Der Fortran-Teil läuft in einem Thread und ruft bei jedem Zeitschritt eine sogenannte Callback-Funktion, die in Python läuft. Der Nutzer hat mit diesem komplexen Ansatz nichts direkt zu tun, sondern kann interaktiv, also Schritt für Schritt, durch eine MF6-Rechnung gehen und bei jedem Zeitschritt auf alle Variablen zugreifen.

Für die Nutzung einer neuen Version von MF6 war die Neu-Kompilierung der Python-Erweiterung nötig. Für jede MF6-Version ist daher eine separate Erweiterung nötig.

Programmtechnische Weiterentwicklung

Die Lösung über den *memory manager* funktioniert. Sie verlangt aber tiefes Wissen über die internen Strukturen von MF6. Der Anwender muss die Bedeutung der Fortran-Variablen-Namen kennen, um mit der Schnittstelle sinnvoll arbeiten zu können. Es gab aber in der Zwischenzeit wichtige Neuerungen in MF6, die eine bessere Lösung erlauben.

Die aktuelle Version von MF6 implementiert das Basic Model Interface (BMI, (CSDMS, 2023a)). Das BMI ist ein Standard für die Kopplung numerischer Simulationsmodelle. Das Community Surface Dynamics Modeling System (CSDMS, (CSDMS, 2023c)) hat den BMI-Standard mit diesen Hauptzielen definiert:

- Ein Modell ist eine wiederverwendbare Komponente (plug and play).
- Ein Modell ist selbstbeschreibend und vollständig durch das Modellier-System oder die Anwendung kontrollierbar.
- Es ist von der Programmiersprache unabhängig.
- Das BMI ist nicht-invasiv. Es gibt keine Änderungen im Verhalten zwischen der BMI- und Nicht-BMI-Version.
- Ein Modell kann sowohl eigenständig als auch als Framework genutzt werden.

Das BMI spezifiziert 42 Funktionen. Die Tabelle 1 in Anhang A zeigt eine Auswahl davon.

`bmi-python` (CSDMS, 2023b) ist die Spezifikation des BMI in Python. Aus technischer Sicht nutzt es dafür abstrakte Basisklassen. Eine Implementierung muss daher alle abstrakten Methoden überschreiben. `xmipy` (Deltares, 2023) ist eine Erweiterung des `bmi-python`, das alle abstrakten Methoden implementiert. `xmipy` greift auf die MF6-DLL mit `ctypes` (Foundation, 2023) zu. Weiterhin fügt `xmipy` zusätzliche Methoden hinzu, die eine tiefere Integration in den Gleichungs-Lösungs-Algorithmus von MF6 erlauben. Damit ist eine Kopplung mit anderen Modellen auf der Ebene der Lösungs-Zeitschritte möglich.

`pymf6` wurde auf die Nutzung von `xmipy` umgestellt. Damit wird die Handhabung einfacher, da die DLL zur Laufzeit wählbar ist und damit mehrere MF6-Versionen mit der gleichen Version von `pymf6` ausführbar sind. Die Nutzerschnittstelle ändert sich damit nur unwesentlich.

Die Nutzung von `xmipy` erfordert weiterhin tiefes Wissen über die Interna von MF6. Die Bibliothek `modflowapi` (Hughes et al., 2022) nutzt `xmipy` und fügt eine wesentlich einfachere Nutzerschnittstelle hinzu. `modflowapi` wird noch aktiv entwickelt. Die erste lauffähige, aber noch unvollständige Version 0.1 wurde im April 2023 veröffentlicht. Die verbesserte Version 0.2 folgte im Februar 2024. Der Wärmetransport in MF6, also das Groundwater Energy Transport (GWE) Model, wird seit Juli 2024 unterstützt.

`modflowapi` ist in `pymf6` integriert. Es gibt regelmäßigen Austausch mit den `modflowapi`-Entwicklern. Die Tabelle 3.1 gibt einen Überblick über den persönlichen Austausch mit Mitgliedern des `modflowapi`-Entwickler-Teams.

Tabelle 3.1.: Persönliche Kommunikation mit den MF-Entwicklern

Termin	**Art der Kommunikation**
18.09.2023	Online-Treffen mit dem MF6-Entwickler-Team
30.11. / 01.12.2023	persönliches Treffen mit dem MF6-Entwickler-Team auf Hydrology Software Days in Delft, Niederlande
31.05. – 05.06.2024	persönliches Treffen mit dem MF6-Entwickler-Team auf der MODFLOW and More Conference, Princeton USA Teilnahme am Short Course zu MF6
21./22.11.2024	persönliches Treffen mit dem MF6-Entwickler-Team auf Hydrology Software Days in Delft, Niederlande

Aus der Entwicklung von `pymf6` sind einige Verbesserungsvorschläge in Form von Issues und einem Pull Request [1] in `modflowapi` eingeflossen. So haben Testläufe von `pymf6` gezeigt, dass die derzeit aktuelle Version 0.3.0dev von `modflowapi` die Zeitschritt-Eingaben (time series) nicht berücksichtigt. Diese bilden die Grundlage für ein entsprechendes Issue im GitHub-Repository, das dieses Problem detailliert und nachvollziehbar beschreibt. Das `modflowapi`-Entwickler-Team arbeitet an einer Lösung. In `pymf6` ist eine Übergangslösung integriert, die optional auf einen Austausch auf Lösungs-Zeitschritten verzichtet und nur einen Austausch auf Modell-Zeitschritt-Ebene realisiert. Mit dieser Methode kann es Zeitschritt-Werte berücksichtigen. Für viele Zwecke ist diese weniger tiefe Integration in die Zeitsteuerung ausreichend.

[1] Issues und Pull Requests sind Werkzeuge der Softwareentwicklung, die es erlauben, in strukturierter Form Fragen zu klären und zur Quelltext-Basis beizutragen.

Ein anderes Problem war eine Diskrepanz in der Anzahl der von `modflowapi` gesendeten Ereignisse `stress_period_start` und `stress_period_end` bei MF6-Simulationen mit mehreren (Teil-)Modellen. Für Simulationen mit einem Modell hat die Anzahl der Anfangs- und End-Ereignisse übereingestimmt. Bei Simulationen mit mehreren Modellen fehlten aber die End-Ereignisse für die weiteren Teilmodelle. Dieses Problem ist beim Testen von `pymf6` aufgefallen. Die Ursache war die Position der Auslösung im Quelltext von `stress_period_end` in `modflowapi`. Das Problem konnte in Abstimmung mit den Entwicklern von `modflowapi` gelöst werden. Die Lösung besteht aus der nötigen Quelltext-Änderung in `modflowapi` und der substanziellen Erweiterung von Tests, um dieses Problem zu identifizieren. Die Änderungen sind mittels eines Pull Request mit dem Titel „Fix number of stress period end states for multi-model simulations"(Müller, 2025) in das GitHub-Repository eingegangen. Am 8. Mai 2025 hat das `modflowapi`-Team diesen Pull Request angenommen und in den Entwicklungsstrang (`branch develop`) von `modflowapi` übernommen (Larsen, 2025). Damit ist der darin enthaltene Quelltext Bestandteil von `modflowapi`.

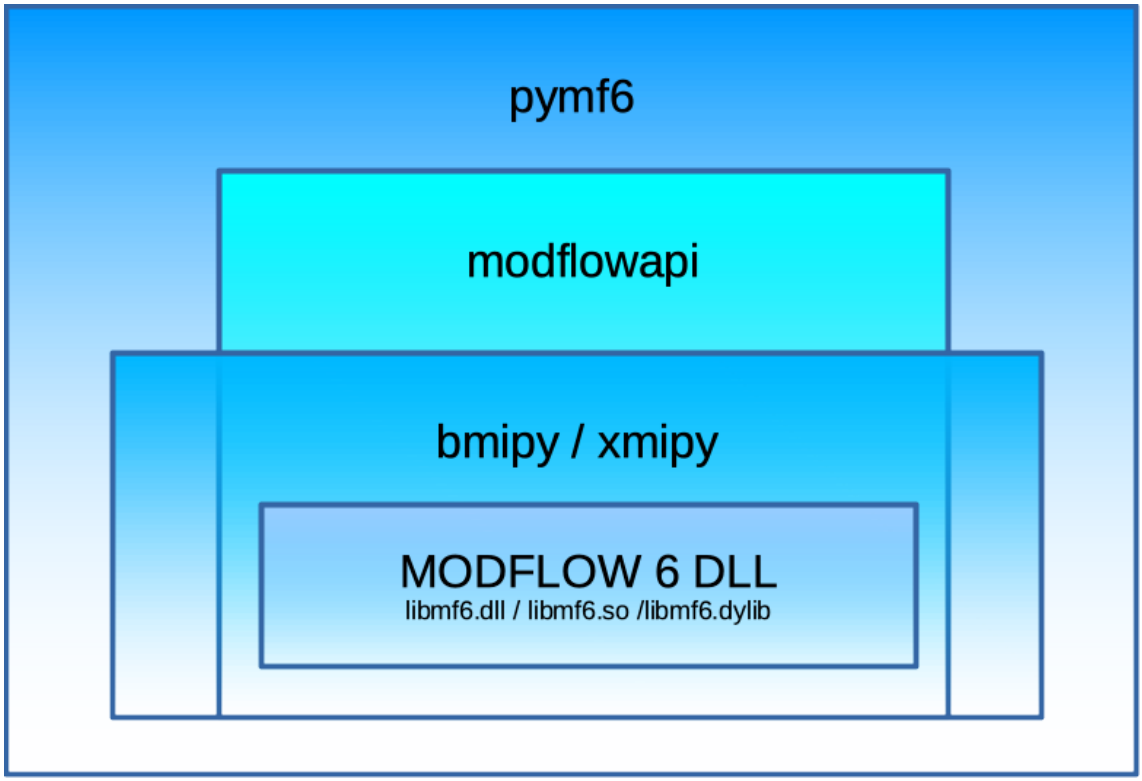

Abbildung 3.1.: pymf6 – Softwarearchitektur

Abbildung 3.1 zeigt die aktuelle Architektur von `pymf6`. Der Laufzeit-Zugriff auf MF6 ist sowohl über `modflowapi` als auch über `xmipy` möglich. In den meisten Fällen ist der Zugriff über `modflowapi` zu empfehlen, da die Schnittstelle weniger Wissen über die MF6-Interna erfordert und viele Modell-Informationen mit Nutzer-Terminologie anreichert. Der Zugriff über `xmipy` kann aber für erfahrene Anwender nützlich sein, um komplexere Abläufe umzusetzen oder Fehler zu suchen.

3.2. AP 1.2 – Validierung der Implementierung anhand synthetischer Testszenarien

3.2.1. Methodik

Die Validierung der Abbildung der Strömung sowie Stoff- und Wärme-Transport von `pymf6` erfolgte mit MODFLOW 6 Examples. Der United States Geological Survey (USGS) stellt ca. 155 Beispielmodelle für MF6 bereit (USGS, 2025) (im Weiteren MF6-Examples genannt). Diese decken eine breite Palette von Anwendungsfeldern für die Modellierung von Strömung, Transport und Wärmeausbreitung ab. Das MF6-Entwickler-Team hält diese Modelle aktuell, so dass sie mit der aktuellen Version von MF6 funktionieren. Daher bieten diese Modelle eine sehr gute Grundlage, die Funktionsfähigkeit von `pymf6` zu validieren.

Alle Modelle wurden mit MF6, mit `modflowapi` und mit `pymf6` gerechnet. MF6 arbeitet mit Modell-Zeitschritten. In jedem Zeitschritt wird das Gleichungssystem gelöst. Innerhalb dieser Lösung gibt es Lösungsschritte. In der Standard-Variante von `pymf6`, die auf `modflowapi` beruht, bietet jeder Lösungsschritt die Möglichkeit des Austausches von MF6-Modelllaufzeit-Daten. Dies erlaubt eine sehr feine Abstimmung der Modellsteuerung über `pymf6`. Es gab aber einige Modelle, bei denen es zu Abweichungen zwischen den Ergebnissen der MF6-Rechnungen und der Rechnungen mit `pymf6` und `modflowapi` kam. Daher erfolgte eine zusätzliche Rechnung mit `pymf6` mit einer modifizierten Zeitschritt-Steuerung (Hier `pymf6_ns` genannt. Der Zusatz `ns` steht für „no step", also kein Lösungsschritt). Anstatt für jeden Lösungsschritt erlaubt diese Variante den Austausch nur nach jedem Modell-Zeitschritt. Für viele Anwendungsfelder ist dieser Ansatz ausreichend. Da die Steuerungseingriffe in größeren Abständen erfolgen, ist die Reaktion des Modells etwas verzögert. Das kann zu Schwingungen von Werten führen. Wie stark diese Schwingungen sind und ob diese die Lösung der Aufgabe signifikant beeinflussen, hängt stark von der Art der Anwendung ab. Für viele Probleme ist dieser Ansatz durchaus ausreichend.

Für alle Lösungs-Varianten (MF6, `modflowapi`, `pymf6` und `pymf6_ns`) wurden die MF6-Examples gerechnet. Einige Modelle ließen sich nicht automatisiert rechnen, da eine manuelle Interaktion nötig war, in der eine Ergebnisdatei eines Modells als Eingabedatei für ein anderes Modell zur Verfügung gestellt werden muss. Da es sich dabei um Modelle handelt, die Partikel-Tracking abbilden, welches für die Anwendungsfälle Stoff- und Wärmetransport nicht relevant ist, wurden diese Modelle nicht gerechnet. Das MF6-Entwickler-Team ist sich dieser Situation bewusst und wird das Verfahren in der Zukunft anpassen, so dass eine automatisierte Rechnung möglich wird. Alle Ergebnisse der Rechnungen `modflowapi`, `pymf6` und `pymf6_ns` wurden mit denen von MF6 verglichen. Zum Vergleich wurden die List-Dateien der Modelle, die unter anderem die Bilanzen aller Quellen- und Senken beinhalten, herangezogen. Da für alle Berechnungen die gleiche Version von MF6 zum Einsatz kam, müssen alle Ergebnisse vollständig gleich sein.

3.2.2. Programmtechnische Umsetzung

Die Berechnungen wurden mit Python automatisiert. Das Modul `pymf6-tools` stellt dafür entsprechende Werkzeuge zur Verfügung. Alle Berechnungen lassen sich mit einer Steuerungsdatei anpassen (siehe Listing 3.1). Darin verweist `mf6_examples_path` auf den Pfad mit den Eingabedateien aller Modelle.

```
[paths]
base_path = models/2025_02_06
mf6_examples_path = ${base_path}/mf6examples
tests_path = ${base_path}/tests
out_path = out/all
pickle_file_name = ${out_path}/results.pcl
# taken from pymf6 config if empty
mf6_exe_path =
mf6_dll_path =
controlled_in_path = controlled
```

Listing 3.1: Steuerungsdatei für Validierungsrechnungen

Die Eingabedateien aller MF6-Examples-Modelle liegen in einem Verzeichnis. Für jeden Modell-Lauf für die Varianten MF6, `modflowapi`, `pymf6` und `pymf6_ns` erstellt das Programm ein Unter-Verzeichnis, in das es die Eingabedateien kopiert. Die Berechnungen einer Variante laufen dann in den jeweiligen Modellverzeichnissen. Für jede Modell-Variante existiert im Verzeichnis `runners` eine Python-Datei mit einem Namen nach dem Muster `run_<varianten-name>.py`, also `run_mf6.py`, `run_modflowapi.py`, `run_pymf6.py` und `run_pymf6_ns.py`. Das Programm geht durch diese Verzeichnisse und ruft die jeweilige Datei `run_<varianten-name>.py` für alle MF6-Examples in einem neuen, externen Prozess auf. Diese Prozesse startet das Programm mit `subprocess.run()`. Die Ausgaben auf den Standard-Ausgabe- und den Standard-Fehler-Kanal (stdout und stderr) fängt das Programm ab und nutzt diese, nach dem Ausfiltern irrelevanter Inhalte, für die Auswertung. Alle Ergebnisse werden schrittweise in einem Python-Dictionary je Variante gesammelt und in der Datei `results.pcl` als serialisierte Python-Objekte abgelegt. In einem Nach-Verarbeitungs-Schritt werden die darin enthaltenen Informationen in pandas DataFrames umgewandelt und in einer HDF5-Datei je Variante abgelegt. Diese Sammel- und Umwandlungs-Schritte sind nötig, weil die Daten in einer anderen Reihenfolge anfallen, als es für die Auswertung nötig ist.

Nach den Modell-Läufen nutzt `pymf6-tools` ein Post-Processing-Modul, das den Vergleich der Ergebnisse von Läufen der Modell-Varianten gegen die von MF6 erleichtert.

3.2.3. Ergebnisse

Die Ergebnisse liegen als pandas-DataFrames vor und lassen sich damit relativ einfach auswerten. Abbildung 3.2 zeigt einen Ausschnitt der Ergebnisse der Rechenläufe mit `pymf6_ns`.

Mit Hilfe von Filtern lassen sich alle fehlgeschlagenen Berechnungen mit spezifischen Merkmalen finden. Listing 3.2 zeigt einen Filter für fehlgeschlagene Rechnungen.

```
pymf6_ns = pd.read_hdf('out/pymf6_ns.h5')
err = pymf6_ns[~pymf6_ns.success]
err
```

Listing 3.2: Filtern der Ergebnisse nach nicht erfolgreichen Rechnungen

Das Ergebnis zeigt Abbildung 3.3. Die Spalte **error** zeigt nur einen Hinweis auf den Fehler. Die Fehlerursache geht aus dem Inhalt der Datei `*.lst` hervor. Die gezeigten, fehlgeschlagenen Modellläufe sind die in Abschnitt 3.2.1 erwähnten Modelle für die

		success	error	run_time
ex-gwf-sfr-pindersauera		True		1.169423
ex-gwf-maw-p01b		True		0.763219
ex-gwt-henry-a		True		6.109497
ex-gwf-hanir		True		0.042840
ex-gwt-mt3dms-p03		True		1.573474
...	...	...	...	...
ex-gwf-disvmesh		True		0.192921
ex-gwe-geotherm	mf6gwf	True		1.063666
	mf6gwe	True		23.477739
ex-gwt-moc3d-p02	mf6gwf	True		0.127740
	mf6gwt	True		7.909328

Abbildung 3.2.: Ergebnisse der Rechnungen mit `pymf6_ns` (Ausschnitt)

Nutzung des Partikel-Trackings. Nach dem Entfernen dieser Modelle aus den Varianten-Läufen sind alle Berechnungen fehlerfrei durchgelaufen. Das sagt allerdings nichts über die Richtigkeit der Ergebnisse aus.

		success	error	run_time
ex-prt-mp7-p03	prt	False	'prt6'\n\n	0.048662
ex-prt-mp7-p04	gwf	False	process did not run\n	0.000000
	prt	False	process did not run\n	0.000000
ex-prt-mp7-p02	gwf	False	process did not run\n	0.000000
	prt	False	process did not run\n	0.000000
ex-prt-mp7-p01	prt	False	'prt6'\n\n	0.305568

Abbildung 3.3.: Fehlgeschlagene Modell-Läufe der Rechnungen mit `pymf6_ns`

Für die Berechnungen mit `modflowapi` und `pymf6` ergeben sich für diese Modelle signifikante Abweichungen in den Ergebnissen:

- ex-gwf-csub-p03a
- ex-gwf-fhb
- ex-gwf-advtidal
- ex-gwf-csub-p03b

Abbildung 3.4 zeigt Ausschnitte aus den Abweichungen zwischen den Ergebnissen der Rechnung mit `modflowapi` und MF6 für das Modell `ex-gwf-fhb`. Ursache dieser Unterschiede ist nach jetzigem Kenntnisstand die in Abschnitt 3.1.2 beschriebene fehlende Berücksichtigung der Zeitschritt-Eingaben (time series), die über ein GitHub-Issue an das MF6-Entwickler-Team übermittelt wurde.

3.2.4. Nutzungsmöglichkeiten von `pymf6-tools`

Das Werkzeug `pymf6-tools` ermöglicht es relativ schnell nachzuprüfen, ob ein Modell mit `pymf6` mit oder ohne Lösungsschritt-Steuerung einsetzbar ist. Die Modell-Eingabe-Daten kann ein Modellanwender mit einem Programm mit grafischer Nutzeroberfläche wie ModelMuse oder Groundwater Vista oder programmatisch mit flopy erstellen. Danach

lineno	mf6	modflowapi
ex-gwf-fhb.lst_351	STO-SS = 0.0000 STO-SS = 0.0000 STORAGE	STO-SS = 3.6826E-12 STO-SS = 0.0000 STORAGE
ex-gwf-fhb.lst_352	WEL = 1850572.8592 WEL = 5807.8512 WEL	WEL = 800000.0000 WEL = 2000.0000 WEL
ex-gwf-fhb.lst_355	TOTAL IN = 1850572.8592 TOTAL IN = 5807.8512	TOTAL IN = 800000.0000 TOTAL IN = 2000.0000
ex-gwf-fhb.lst_359	STO-SS = 100.0215 STO-SS = 0.2045 STORAGE	STO-SS = 18.0000 STO-SS = 3.2399E-12 STORAGE
ex-gwf-fhb.lst_361	CHD = 1850472.8377 CHD = 5807.6467 CHD	CHD = 799981.9994 CHD = 2000.0000 CHD
ex-gwf-fhb.lst_363	TOTAL OUT = 1850572.8592 TOTAL OUT = 5807.8512	TOTAL OUT = 799999.9994 TOTAL OUT = 2000.0000
ex-gwf-fhb.lst_365	IN - OUT = -2.6643E-06 IN - OUT = -2.3920E-10	IN - OUT = 5.9481E-04 IN - OUT = 1.5929E-06
ex-gwf-fhb.lst_367	PERCENT DISCREPANCY = -0.00 PERCENT DISCREPANCY = -0.00	PERCENT DISCREPANCY = 0.00 PERCENT DISCREPANCY = 0.00
ex-gwf-fhb.lst_447	STO-SS = 0.0000 STO-SS = 0.0000 STORAGE	STO-SS = 3.6826E-12 STO-SS = 0.0000 STORAGE
ex-gwf-fhb.lst_448	WEL = 2960490.2146 WEL = 5394.6281 WEL	WEL = 1200000.0000 WEL = 2000.0000 WEL

Abbildung 3.4.: Signifikante Abweichungen zwischen den Ergebnissen der Rechnung mit `modflowapi` und MF6 für das Modell `ex-gwf-fhb`

kann die Prüfung mit den in vorangegangenen Abschnitten gezeigten Arbeitsschritten automatisch erfolgen. Das ist insbesondere nützlich, wenn der Aufbau eines Modells inkrementell erfolgt, das heißt ein Modell erhält nach und nach komplexere Merkmale. Nach jeder Änderung kann die Prüfung mit einem Befehlsaufruf erneut erfolgen.

3.3. AP 2.1 – Kopplung von analytischen Randbedingungen des Stoff- und Wärmetransports mit numerischen Stoff-Wärmetransportmodulen in ueflow

3.3.1. Ansatz der Kopplung analytischer Randbedingungen

Die Phase 1 dieses Projektes hat analytische Lösungen von Grundwasserprozessen für die Kopplung mit `pymf6` analysiert und für die Kopplung aufgearbeitet. Die Lösungen wurden weiter hinsichtlich ihrer Kopplungsfähigkeit untersucht. Dabei hat sich gezeigt, dass sich zahlreiche Ansätze auch mit dem analytischen Elementverfahren (Analytic Element Method – AEM) (Wikipedia, 2025) umsetzen lassen. Mit dem stationären TimML (Bakker und Strack, 2003; Bakker, 2006) und dem instationären TTim (Bakker, 2013a; Bakker, 2013b) stehen leistungsfähige AEM-Python-Bibliotheken zur Verfügung. Somit können viele der analytischen Lösungen mit Hilfe der Kopplung mit TTim eingebunden werden. Tabelle 3.2 gibt einen Überblick über die in Phase 1 analysierten Lösungen und die Möglichkeit der Abbildung in TTim.

Abschnitt 3.3.2 stellt die Kopplung von TTim an `pymf6` für einen Anwendungsfall zur Bestimmung der Brunnenabsenkung dar. Damit lassen sich mit einer wesentlich geringeren Anzahl von Modellzellen realitätsnähere Brunnenwasserstände berechnen. Zusätzlich sind mehrere Brunnen pro Modellzelle möglich. Jeder Brunnen kann hierbei einen individuellen Wasserstand haben.

Es gibt aber analytische Lösungen, die TTim nicht abdecken kann. Daher zeigt Abschnitt 3.3.3 eine Kopplung mit einer analytischen Lösung für den Wärmeübergang aus der Atmosphäre in den Untergrund ohne die Nutzung von TTim.

Weitere Kopplungen sind mit den in den Abschnitten 3.3.2 und 3.3.3 dargestellten Prinzipien umsetzbar. Damit ist es möglich, eine große Palette von Kopplungen analytischer Modelle mit relativ geringem Aufwand umzusetzen.

Tabelle 3.2.: Übersicht der analytische Lösungen und ihrer möglichen Abbildung mit TTim

Randbedingung	Referenz(en)	TTim-Ansatz	Genutzte Funktion
Wechselwirkung zwischen Bodenoberflächentemperatur und Grundwasser	Händel et al., 2013, Pannike et al., 2006	nicht möglich	-
Bewertung der Leistung von Erdwärmepumpen (Zwei-Brunnen-Variante)	Banks, 2009	nur indirekte Approximation möglich	-
Wärmetransport ins Grundwasser auf Aquiferskala	Anderson, 2005	Kreisrunde Neubildungsfläche	`ttim.CircAreaSink`
Fluss-Grundwasser-Interaktion	Anibas et al., 2011, Vogt et al., 2012, Hatch et al., 2006, Constantz, 2008, Molina-Giraldo et al., 2011, Rau et al., 2010, Keery et al., 2007	Mäandrierender oder gerader Fluss	`ttim.HeadLineSinkString`
Jährlicher periodischer Einfluss der Oberflächentemperatur auf das Grundwasser	Bundschuh, 1993	nicht möglich	-
Einfluss eines Erdwärmepumpenfeldes auf die Grundwassertemperatur	Diao et al., 2004	Brunnen	`ttim.Well`
Wechselwirkung zwischen Bodenoberflächentemperatur und Erdwärmepumpe	Goto et al., 2005	nicht möglich	-
Wärmeentzug aus einer geklüfteten Gesteinsmatrix	Gringarten et al., 1975	nicht möglich	-
Wärmetransport bei Erdwärmesonden unter stationären Bedingungen	Hähnlein et al., 2010b	Brunnen (TimML)	`tml.Well`
Transiente radiale Wärmekonduktion (Linienquellen-Approximation)	Hecht-Méndez et al., 2010	Brunnen	`tttim.Well`
Einfluss eines Erdwärmesondenfeldes	Claesson und Javed, 2011	Mehrere Brunnen	`tttim.Well`
Wärmetransport im Karst	Luhmann et al., 2015	nicht möglich	-
Temperaturermittlung für eine Erdwärmesonde (abgeleitet aus Slug-Test)	Shook, 2001	Brunnen (TimML)	`tml.Well`

3.3.2. Kopplung mit Analytic Element Models

Das analytische Elementverfahren (Analytic Element Method – AEM)

Das analytische Elementverfahren (AEM) ist ein semi-analytisches Verfahren zur Lösung von partiellen Differentialgleichungen (Wikipedia, 2025). Durch die Kombination numerischer und analytischer Ansätze ergeben sich für bestimmte Fragestellungen Vorteile. Nur die inneren und äußeren Grenzen werden diskretisiert. Flächen und Volumen sind in AEM aber nicht diskretisiert und liegen kontinuierlich vor, sodass Lösungswerte, wie der Grundwasserstand, an beliebigen Stellen vorliegen. Im Gegensatz dazu liegen bei rein numerischen Verfahren die Ergebnisse nur an den Diskretisierungsstellen wie Modellzell-Mittelpunkten oder Modellknoten vor. Die AEM bestimmt die Randintegrale analytisch.

Das mathematische Prinzip der AEM besteht in der Überlagerung von linearen Differentialgleichungen elementarer Lösungen, um komplexere Lösungen abzubilden. Unterschiedliche analytische Lösungen bilden die Elemente, die geometrische Grenzen wie Punkte, Linien und Kreise mit unterschiedlichen Randbedingungen belegen können. Die analytischen Lösungen enthalten meist Freiheitsgrade in Form von Koeffizienten.

Die Berechnung dieser Koeffizienten erfolgt durch ein System von Gleichungen, sodass die Randbedingungen an allen Elementen erfüllt sind. Die Lösung bietet eine räumlich kontinuierliche Beschreibung der abhängigen Variable, also beispielsweise des Grundwasserstandes. Durch die Überlagerung zahlreicher Elemente sind auch komplexe Geometrien möglich. Damit kann die AEM viele praktische Anwendungsfälle abbilden.

Die Kopplung von TTim mit numerischen Modellen wie MODFLOW bietet gegenüber herkömmlichen semi-analytischen Ansätzen erhebliche Vorteile, da sie rechnerische Effizienz mit verbessertem physikalischen Realismus verbindet. Während analytische Lösungen schnelle Ergebnisse für vereinfachte Fälle (Einzelbrunnen, homogene Grundwasserleiter, gerade Flüsse) liefern, können sie die Komplexität der realen Welt, wie zum Beispiel mäandrierende Flüsse, mehrschichtige Systeme oder transiente Randbedingungen, nicht erfassen.

TTim ist aufgrund seiner Fähigkeit, realistische Feldbedingungen abzubilden rein analytischen Ansätzen für viele praktische Anwendungen oft überlegen. So kann es zum Beispiel variable Flusskolmationen, transiente Pumpvorgänge und interagierende Grundwasserleiter abbilden. TTim benötigt typischerweise mehr Rechenzeit als analytische Lösungen, ist aber meist wesentlich schneller als vergleichbare numerische Lösungen. So schließt TTim die kritische Lücke zwischen stark vereinfachten analytischen Modellen und rechenintensiven vollständig numerischen Simulationen. Das ist insbesondere für die kontrollierte Grundwasseranreicherung, geothermische Systeme und integrierte Oberflächenwasser-/Grundwasserstudien relevant, bei denen sowohl Genauigkeit als auch Effizienz von größter Bedeutung sind.

Technische Übersicht über TTim

TTim wurde für die transiente Simulation der Grundwasserströmung in gespannten und halbgespannten Grundwassersystemen entwickelt, insbesondere für radiale, achsensymmetrische Simulationsgebiete. Es unterstützt die Simulation zeitabhängiger Randbedingungen, Brunnenwechselwirkungen und Grundwasserreaktionen, ohne dass eine räumliche Diskretisierung erforderlich ist. Damit eignet es sich ideal für konzeptionelle Modellierungen, Pumpversuchsanalysen und Studien mit dynamischen Belastungsszenarien.

Eine der leistungsstarken Funktionen von TTim ist die Klasse `ttim.HeadLineSinkString`, die eine detaillierte Modellierung von mäandernden Flüssen und Wechselwirkungen zwischen Flüssen und Grundwasserleitern ermöglicht. Diese Funktion ermöglicht eine abschnittsweise Darstellung, bei der Flüsse mit segmentierten Linien (Polylinien) modelliert werden können, die gekrümmte oder verzweigte Flussläufe genau wiedergeben. Räumlich variable Pegelstände entlang des Flusses sind möglich, wodurch Ereignisse wie saisonale Schwankungen und variable Leckagen simuliert werden können. Die Kolmation des Flussbettes kann unter Berücksichtigung heterogener Flussbettmaterialien und -bedingungen variabel dargestellt werden.

Die Klasse `ttim.DischargeWell` bietet einen hochgradig anpassungsfähigen Rahmen für die Simulation von Pump- und Injektionsbrunnen mit einer Vielzahl von Betriebs- und Randbedingungen. Sie ist in der Lage, vollkommene sowie unvollkommene Brunnen (Letztere sind Brunnen, welche nicht die komplette Mächtigkeit erfassen.) mit dynamisch variierenden Abstromraten auszustatten. Die Klasse unterstützt hier die funktionsbasierte Änderung von Entnahmeraten (oder Injektionsraten) und / oder der Wasserstände, mit

anderen Worten: es werden dynamische Dirichlet- oder Neumann-Randbedingungen ermöglicht.

Während sich TTim auf die transiente Analyse in Einschichtsystemen konzentriert, fokussiert TTimML auf stationärer Strömungen in mehrschichtigen Systemen. Im Gegensatz zu traditionellen analytischen Modellen, die oft auf sehr stringente Annahmen beschränkt sind, kann TTimML u.a. Grundwasserstockwerke einschließlich gespannter, undichter und halbgespannter Systeme abbilden. Die Einbeziehung vertikaler Strömungen und Leckage erlaubt es, auch die Strömung zwischen den Schichten abzubilden. Hierbei sind schichtspezifische Randbedingungen möglich, sodass unterschiedliche Druckhöhen oder Flussraten jeder Schicht zugewiesen werden können. Diese Funktionalität ist für die Modellierung geologisch komplexer Systeme, in denen vertikale Heterogenität eine wichtige Rolle in der Strömungsdynamik spielt, von entscheidender Bedeutung.

TTim bietet einen leistungsstarken Rahmen für die Kopplung analytischer und numerischer Lösungen. TTim hat hierbei große Stärken im Bereich der Modellierung von Oberflächenwasser-Grundwasser-Wechselwirkungen (siehe Anderson, 2005; Anibas et al., 2011; Keery et al., 2007; Vogt et al., 2012; Rau et al., 2010; Hatch et al., 2006; Constantz, 2008) und der Abbildung von Brunnensystemen (siehe Banks, 2009; Diao et al., 2004; Gringarten et al., 1975; Hähnlein et al., 2010a; Hecht-Méndez et al., 2010; Claesson und Javed, 2011; Pannike et al., 2006; Shook, 2001) TTim kann jedoch derzeit bestimmte Randbedingungen noch nicht abdecken, wie zum Beispiel atmosphärische Interaktionen (siehe Händel et al., 2013; Bundschuh, 1993; Goto et al., 2005; Molina-Giraldo et al., 2011) oder Karstaquifere (Luhmann et al., 2015).

Implementierung der Kopplung TTim und `pymf6`

Die Kopplung von `pymf6` und TTim ist derzeit nur für regelmäßige Diskretisierungen, also rechteckige Modellzellen, umgesetzt. Andere Modell-Zell-Geometrien wie Dreiecke oder Voronoi-Elemente sind mit der hier dargestellten Methode auch möglich, erfordern aber andere geometrische Interpolations-Algorithmen für die Randbedingungswerte des AEM-Modells.

Zu Beginn des gekoppelten Modelllaufs extrahiert das Kopplungsprogramm alle relevanten, zeitlich nicht veränderlichen Daten aus dem MF6-Modell für die zu koppelnde Modellzelle wie Zell-Geometrie, Mächtigkeit, hydraulische Durchlässigkeit und Porosität. Bei jedem Kopplungs-Zeitschritt erfolgen diese Schritte:

- Extraktion der Grundwasserstände von MF6 aus der zu koppelnden Modellzelle und allen Nachbarzellen.
- Interpolation der MF6-Wasserstände auf die Modellränder (Abbildung 3.5) des TimML-Modells
- Aufbau eines TimML-Modells für die zu koppelnde Modellzelle als Anfangszustand für ein TTim-Modell
- Lauf des TimML-Modells
- Aufbau eines TTim-Modells für die zu koppelnde Modellzelle unter Nutzung der Ergebnisse des TimML-Modells als Anfangszustand
- Lauf des TTim-Modells
- Speicherung der Ergebnisse der TTim-Rechnung für den Zeitschritt

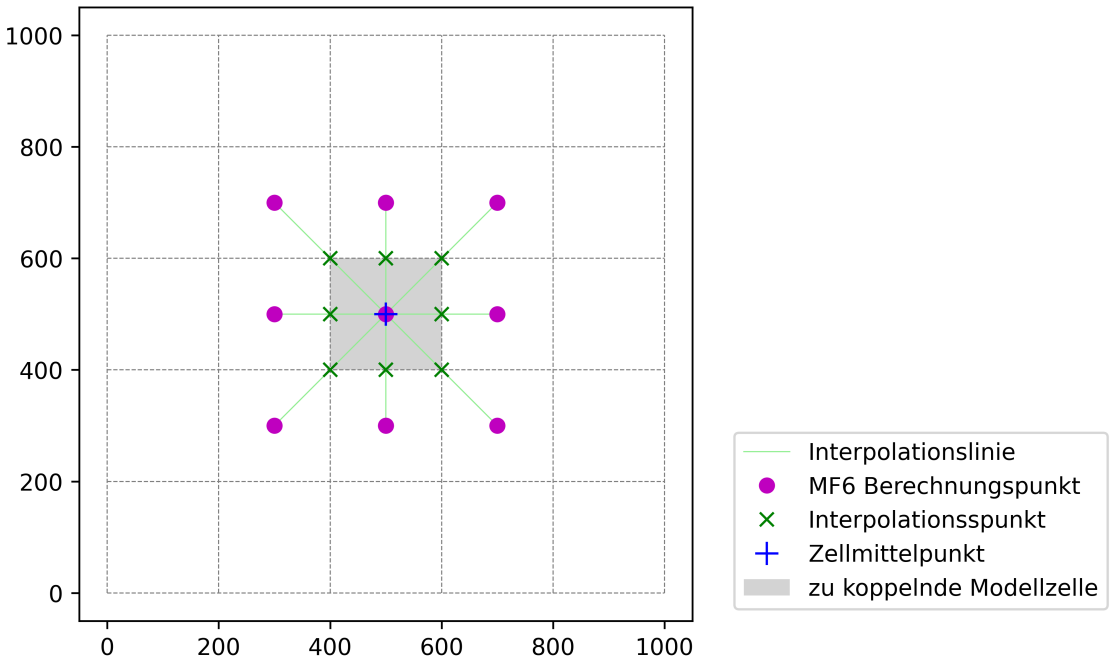

Abbildung 3.5.: Interpolation der Wasserstände aus MF6 zur Nutzung als TTim-Randbedingungen

Abbildung 3.5 zeigt die geometrische Anordnung der MF6-Modell-Zelle, die für die Interpolation der Grundwasserstände auf den Rand des TimML-Modells zum Einsatz kommt. Insgesamt gehen neun MF6-Berechnungspunkte ein. Die Interpolation erfolgt linear auf acht Interpolationspunkte, indem jeweils linear zwischen dem Zellmittelpunkt und einem der äußeren MF6-Berechnungspunkte interpoliert wird. Derzeit geht der Interpolations-Algorithmus davon aus, dass alle Nachbarzellen die gleichen Durchlässigkeitswerte wie die zu koppelnde Modellzelle haben. Für viele Anwendungsfälle trifft das zu, da häufig ein Durchlässigkeitswert einer Modell-Schicht zugewiesen ist. Weiterhin werden nur die Modellwasserstände der horizontalen Nachbarzellen berücksichtigt. Das ist oftmals eine sinnvolle Modellannahme für typische anisotropische Verhältnisse mit zehnfach höherer horizontaler als vertikaler Durchlässigkeit.

Dabei gelten diese Grundregeln:

- Die Rückwirkung der TTim-Rechnung auf die MF6-Rechnung ist optional. Für viele Anwendungen ist die unbeschränkte räumliche Auflösung ausreichend und eine Rückwirkung der TTim-Ergebnisse auf die MF6-Rechnung ist für die aktuelle Aufgabe nicht nötig. Die Ergebnisse der TTim-Rechnung können aber auch dazu dienen, MF6-Variablen-Werte zu modifizieren. So könnte der genauere Brunnenwasserstand aus der TTim-Rechnung als Steuerbedingung für die Brunnenentnahme-Menge in MF6 zum Einsatz kommen.
- Die Grundwasserstände werden auf die Modellränder des TTim-Modells aus den MF6-Modell-Zell-Wasserständen der gekoppelten Zelle und aller acht Nachbarzellen distanz-abhängig interpoliert (Abbildung 3.5). Ein anderer Interpolationsalgorithmus, der beispielsweise die Durchlässigkeitswerte der Zellen einbezieht und/oder die Werte aus anderen Schichten einbezieht, ist möglich. Auch andere Modellgeometrien sind grundsätzlich umsetzbar. Das Kopplungsprogramm ist nach den Grundsätzen der objekt-orientierten Programmierung erstellt. Ein Aus-

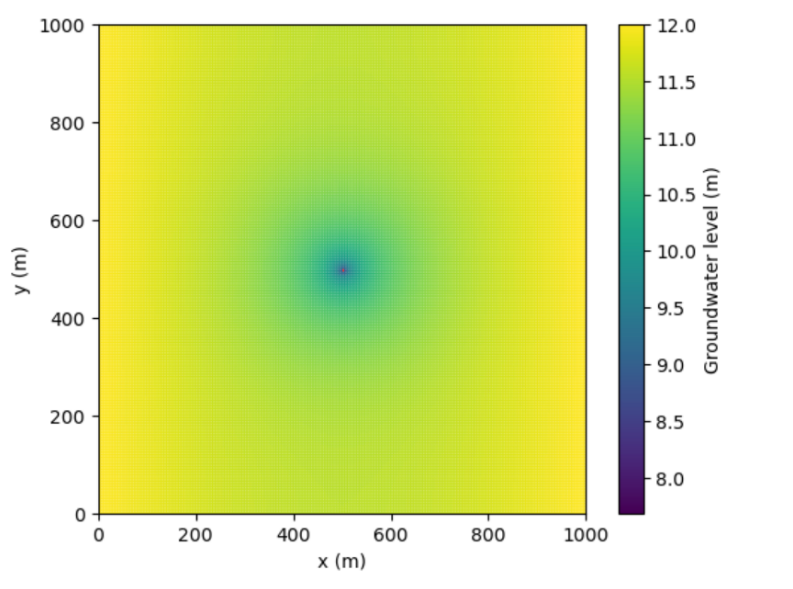

Abbildung 3.6.: Modellergebnisse – Absenkung um den Brunnen in der Modellmitte (Modellauflösung 5 x 5 m)

tausch der aktuellen Klasse für die Interpolation mit einer anderen Klasse, die andere Interpolationsmethoden unterstützt, ist bereits vorgesehen.

Anwendungsfall: Reduzierung der Anzahl der Modellzellen um einen Brunnen

Um Details in bestimmten Gebieten besser abzubilden, ist es normalerweise nötig, die Modell-Diskretisierung zu verfeinern und mehr Zellen zu nutzen. Bei regelmäßiger Diskretisierung kann das aber zu vielen Modellzellen führen, von denen viele häufig für die Modellaussage nicht nötig sind. Diese sind aber aus technischen Gründen nicht vermeidbar und erhöhen die Rechenzeiten und den Speicherbedarf gegebenenfalls beträchtlich. Auch die Nutzung von Dreiecks- oder Voronoi-Elementen ermöglicht es oft, die Anzahl der Modellzellen für derartige Probleme beträchtlich zu verringern. Der Umbau von einem Modell mit regelmäßiger Diskretisierung in ein Modell mit flexibler Diskretisierung kann aber sehr aufwändig sein.

Mit der hier beschriebenen Modellkopplung mit TTim ist es möglich, für bestimmte Aufgabenstellungen lokal sehr hochaufgelöste Modellergebnisse zu erhalten, obwohl die Modellzellen groß sind. Hier kommt ein Testmodell mit einem Entnahme-Brunnen im Modellzentrum zum Einsatz. Abbildung 3.6 zeigt den Grundwasserstand am Ende der Rechnung. Aus dieser Abbildung wird auch der Modellaufbau mit dem Brunnen in der Mitte und den konstanten Randbedingungen am linken und rechten Modellrand deutlich. Tabelle 3.3 fasst die Modell-Parameter zusammen.

Mit dem hier genutzten MF6-Paket `WEL` berechnet MF6 den Brunnenwasserstand als Grundwasserstand in der Mitte der Modellzelle mit der spezifizierten Brunnenentnahme. In Abhängigkeit von der Fragestellung kann diese räumliche Auflösung ausreichend sein. Für eine Aussage, ob die Pumpe im Brunnen gegebenenfalls trocken fällt, muss die Modellauflösung aber relativ hoch sein. Um diesen Auflösungs-Effekt zu untersuchen,

Tabelle 3.3.: Aufbau des Testmodells für die AEM-Kopplung

Modelleigenschaft	**Wert**
Ausdehnung x-Richtung	1 km
Ausdehnung y-Richtung	1 km
Randbedingung konstanter Wasserstand links und rechts	12 m
Mächtigkeit des Grundwasserleiters	10 m
Anfangswasserstand	12 m
hydraulische Leitfähigkeit k_f	10 m/d $8{,}64^{-4}$ m/s
Modell-Zeit	100 d
Brunnenentnahme q	500 m^3/d

Tabelle 3.4.: Diskretisierung der Testmodelle für die AEM-Kopplung

Modellzellgröße	**Anzahl der Modellzellen**
100 m × 100 m	100
10 m × 10 m	10.000
5 m × 5 m	40.000
1 m × 1 m	1.000.000
0,5 m × 0,5 m	4.000.000
0,25 m × 0,25 m	16.000.000

kamen Modelle mit unterschiedlichen Diskretisierungen von Modellzellgrößen von 100 x 100 m bis 0,25 x 0,25 m zum Einsatz. Tabelle 3.4 zeigt die Diskretisierungs-Details.

Abbildung 3.7 zeigt das alternativ dazu erstellte TTim-Modell, das die MF6-Modell-Zelle mit dem Brunnen repräsentiert nach einer Berechnung (unter Verwendung einer sehr großen Zellgröße von 100 × 100 m). Der Absenkungstrichter in der Modell-Mitte ist symmetrisch. Die Ballung der Isohypsen am Rand ist ein Artefakt des Modellaufbaus mit konstanten Wasserständen (RB 1. Art).

Die Erzeugung der MF6-Eingabedateien und der Lauf der Berechnungen wurden mit (Bakker et al., 2016) automatisiert. Tabelle 3.5 zeigt die Modelllaufzeiten und die berechneten Brunnenwasserstände. Es zeigt sich, dass der Brunnenwasserstand stark von der Diskretisierung abhängt. Der Brunnenwasserstand für die Diskretisierung von 0,25 m × 0,25 m ist höher als der für die Auflösung mit Zellen der Größe 0,5 m × 0,5 m. Dieser Effekt ist unerwartet und könnte auf numerische Fehler hinweisen.

Der mit der mit TTim gekoppelten Rechnung bestimmte Brunnenwasserstand ist noch niedriger. Das ist unerwartet, da der Brunnenradius im TTim einen Wert von 0,3 m hat. Die Ergebnisse liegen aber im Bereich von ca. 5,3 bis 5,7 m hinsichtlich typischer Unsicherheiten bei Modellparametern wie hydraulischen Durchlässigkeiten nahe genug zusammen.

Die Rechenzeit lässt sich mit dem gekoppelten Ansatz signifikant verringern. Gegenüber der MF6-Rechnung mit einer Diskretisierung mit Zellen der Größe 0,5 m × 0,5 m ergibt sich eine ca. 30-fach kürzere Rechenzeit. Das ist eine erhebliche Beschleunigung. Auch die Einsparung des Speicherbedarfs ist erheblich. So lassen sich insbesondere bei der Speicherung der Ergebnisse für viele Zeitschritte viele GB Speicherplatz einsparen.

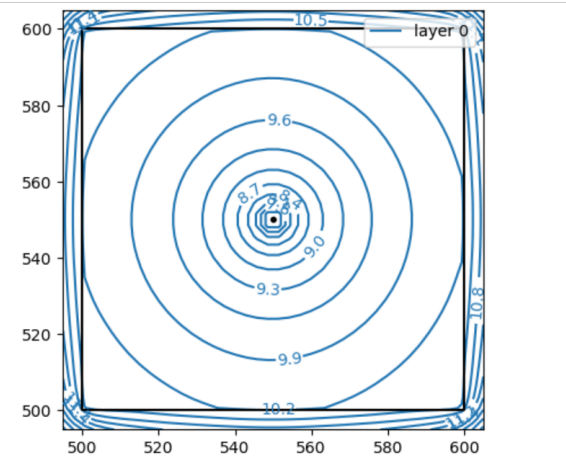

Abbildung 3.7.: Abbildung eines Brunnens mit TTim in der Mitte einer MF6-Zelle

Tabelle 3.5.: Laufzeiten der Testmodelle für die AEM-Kopplung

Zellengröße	Zellenanzahl	Laufzeit (s)	Brunnenwasserstand (m)
100 m × 100 m	100	0,15	9,23
10 m × 10 m	10.000	0,78	7,30
5 m × 5 m	40.000	2,81	6,78
1 m × 1 m	1.000.000	68,22	5,98
0,5 m × 0,5 m	4.000.000	271,31	5,69
0,25 m × 0,25 m	16.000.000	1111,33	5,88
100 m × 100 m	100	8,75	5,27

MODFLOW 6
MODFLOW 6 + analytische Lösung

```
BEGIN options
    TIME_UNITS  days
END options

BEGIN dimensions
    NPER  4
END dimensions

BEGIN perioddata
    1.00000000   1        1.00000000
   10.00000000  10        1.00000000
   10.00000000  10        1.00000000
   10.00000000  10        1.00000000
END perioddata
```

Listing 3.3: Zeitsteuerung Testmodell AEM-Kopplung

Anwendungsfall: Mehrere Brunnen pro Modellzelle

In einer MF6-Zelle wird die Brunnenentnahme mit dem MF6-Paket `WEL` als ein Wert abgebildet. Die Pumpraten mehrerer, räumlich in einer Modellzelle liegender Brunnen erfasst das Modell also nur summarisch. Eine Differenzierung der Einflüsse der einzelnen Brunnen ist daher nicht möglich. Mit der Kopplung mit TTim ist es aber möglich, beliebig viele Brunnen pro MF6-Modellzelle abzubilden. Ein Testmodell soll dies verdeutlichen. Das genutzte Modell hat eine stationäre Stress-Periode zur Etablierung von definierten Anfangsbedingungen gefolgt von drei instationären Stress-Perioden. Die Anzahl und Entnahme-Raten der Brunnen variiert zwischen diesen Stress-Perioden. Listing 3.3 zeigt die MF6-Eingabedatei für die Zeitsteuerung, also für das Paket `TDIS`.

Einige Brunnen sind in mehreren gekoppelten TTim-Modellen erfasst. Ein TTim-Modell enthält dabei ein oder zwei Brunnen. Die Anzahl der Brunnen pro MF6-Zelle ist grundsätzlich nicht begrenzt. Listing 3.4 zeigt die Zuordnung der analytischen Brunnen zu den MF6-Zellen. Mit Hilfe des MF6-Schlüsselwortes `AUXILIARY` lassen sich zusätzliche Spalten für Randbedingungswerte einfügen. Die Spalte `pymf6_analytic` enthält einen Schalter, ob der Brunnen auch analytisch berechnet werden soll (`1`) oder nicht (`0`). Die Spalte `pymf6_well_name` speichert MF6 als `BOUNDNAMES`. Alle Informationen sind zur Laufzeit für die Kopplung verfügbar.

Für die Eingabe der Entnahmeraten mehrerer analytischer Brunnen pro MF6-Modellzelle kam eine zusätzliche Datei im JSON-Format zum Einsatz. Listing 3.5 zeigt diese Datei. MF6-Brunnen wie `mf_pump1` für die zweite Stress-Periode (`"1"`, da Python-Zählung bei Null beginnt) modelliert das gekoppelte TTim-Modell mit zwei Brunnen. Die Summe aller Einträge `"rate_fraction"` pro MF6-Brunnen muss immer eins sein. Das Kopplungsprogramm prüft diese Bedingung und bricht bei Nichterfüllung mit einer entsprechenden Fehlermeldung ab. Für die MF6-Brunnen `mf_pump3` und `mf_pump4` gibt es für die Stress-Periode `"1"` keinen Eintrag in der JSON-Datei, obwohl diese Brunnen in der MF6-Eingabe mit `pymf6_analytic` mit dem Wert `1` als aktive analytische Brunnen gekennzeichnet sind. Diese MF6-Brunnen haben nur einen entsprechenden analytischen Brunnen. Die Auflistung in der JSON-Datei ist also nur bei mehreren analytischen Brunnen je MF6-Brunnen nötig.

Die Ergebnisse für die instationären Stress-Perioden sind in den Abbildungen 3.8, 3.9 und 3.10 zusammengestellt.

```
1
2   BEGIN options
3     AUXILIARY pymf6_analytic
4     BOUNDNAMES
5   END options
6
7   BEGIN dimensions
8     MAXBOUND 4
9   END dimensions
10
11
12  BEGIN period  2
13  #layer row col    Q pymf6_analytic pymf6_well_name
14       1   2   3 -111              1         mf_pump1
15       1   6   7 -250              0         mf_pump2
16       1   3   5 -150              1         mf_pump3
17       1   7   3 -300              1         mf_pump4
18  END period  2
19
20  BEGIN period  3
21  #layer row col    Q pymf6_analytic pymf6_well_name
22       1   2   3 -217              1         mf_pump1
23       1   6   7 -250              1         mf_pump6
24  END period  3
25
26  BEGIN period  4
27  #layer row col    Q pymf6_analytic pymf6_well_name
28       1   2   3 -150              1         mf_pump1
29       1   7   7 -300              0         mf_pump8
30  END period  4
```

Listing 3.4: Brunnenentnahme mit modifizierter WEL-Datei Testmodell AEM-Kopplung

```
1  {
2   "1":
3    {
4     "mf_pump1":
5      [
6       {"name": "pump1", "coords": [220, 120], "rate_fraction": 0.3},
7       {"name": "pump2", "coords": [270, 170], "rate_fraction": 0.7}
8      ]
9    },
10  "2":
11   {
12    "mf_pump1":
13     [
14      {"name": "pump1", "coords": [220, 120], "rate_fraction": 0.5},
15      {"name": "pump2", "coords": [270, 170], "rate_fraction": 0.5}
16     ],
17     "mf_pump6":
18     [
19      {"name": "pump1", "coords": [620, 520], "rate_fraction": 0.1},
20      {"name": "pump2", "coords": [670, 570], "rate_fraction": 0.9}
21     ]
22   },
23  "3":
24   {
25    "mf_pump1":
26     [
27      {"name": "pump1", "coords": [220, 120], "rate_fraction": 0.2},
28      {"name": "pump2", "coords": [270, 170], "rate_fraction": 0.8}
29     ]
30   }
31 }
```

Listing 3.5: Eingabedatei für mehrere Brunnen in einer MF6-Zelle

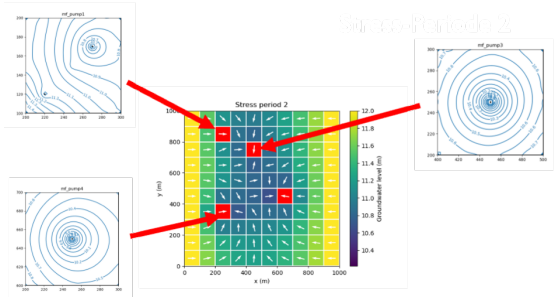

Abbildung 3.8.: Absenkungen der AEM-Brunnen in Stress-Periode 2

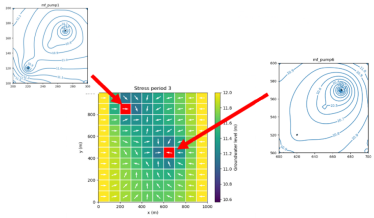

Abbildung 3.9.: Absenkungen der AEM-Brunnen in Stress-Periode 3

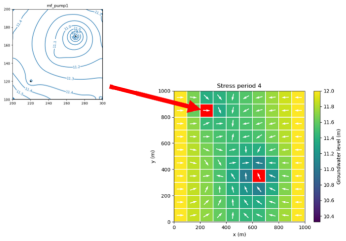

Abbildung 3.10.: Absenkungen der AEM-Brunnen in Stress-Periode 4

3.3.3. Analytische Lösung für Wärmeübergang aus der Atmosphäre

Die in Phase 1 dieses Projektes untersuchte analytische Lösung für den Wärmeübergang aus der Atmosphäre nach Gleichung 2.3 in Händel et al., 2013 wurde in Python implementiert. Abbildung 3.11 zeigt eine schematische Darstellung der eindimensionalen Lösung. Die Lösung ermöglicht die Abbildung der Ausbreitung von Wärme in der ungesättigten Zone von der Geländeoberfläche bis zum Grundwasser. Die Bestandteile sind Wärmekonvektion mit dem Sickerwasser und Wärmeleitung (Konduktion). Das Listing 1 in Anhang B zeigt den vollständigen Quelltext.

In die Rechnung gehen die in Listing 3.6 gezeigten Variablen ein. Es gibt einige wichtige Einschränkungen:

- Die Bewegung des Sickerwassers ist eine Vorgabegröße und wird vom Modell nicht berechnet.
- Der horizontale Wärmetransport wird vernachlässigt.
- Die Transport-Parameter sind homogen über das gesamte, modellierte Sediment.
- Die räumliche und zeitliche Diskretisierung sind geeignet zu wählen, um die Stabilität der Rechnung zu gewährleisten.

Listing 3.7 zeigt die Implementierung des Algorithmus. Um die Übereinstimmung mit Gleichung 2.3 aus Händel et al., 2013 zu zeigen, kam eine Python-Schleife zum Einsatz. Diese Implementierung dient als Vergleichsgrundlage für die weiteren Arbeiten.

Eine derartige Schleife ist für einen Kopplungseinsatz aus Laufzeitgründen nicht geeignet. Bei einer Kopplung mit der Oberkante eines Grundwassermodells muss für jede Modell-Zelle der obersten, aktiven Schicht des Grundwassermodells für jeden Grundwasser-Modell-Zeitschritt ein eindimensionales Modell erstellt und gerechnet werden. Dafür

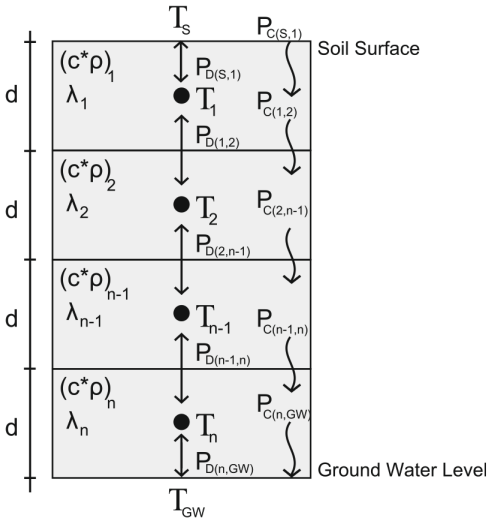

Abbildung 3.11.: Eindimensionale Abbildung der Wärmeausbreitung aus Händel et al., 2013, (Abbildung 3), P_C – Wärme-Konvektion mit dem Sickerwasser, P_D – Wärme-Leitung,

ist ein vektorisierter Ansatz mit NumPy wesentlich besser geeignet, da die Rechenzeiten damit wesentlich geringer sind. Die Vektorisierung des Algorithmus ist derzeit in Bearbeitung.

Das Ergebnis der Berechnung ist in Abbildung 3.12 dargestellt. Zu Rechnungsbeginn hat nur die oberste Schicht die Lufttemperatur von 20°C. Alle anderen Schichten haben eine Temperatur von 10°C. Im Laufe der Rechnung schreitet die Wärmefront langsam nach unten fort. Sensitivitätsanalysen zeigen, dass der konvektive Term, also der Wärmetransport über das Sickerwasser, schon bei relativ geringen Sickerraten bestimmend ist. Die Erhöhung der Sickerraten muss mit einer entsprechenden Verringerung des Modell-Zeitschritts einhergehen, um eine stabile Rechnung zu gewährleisten. Derzeit laufen Arbeiten zur Umsetzung einer automatischen Anpassung des Modell-Zeitschritts an diese Stabilitätsbedingungen.

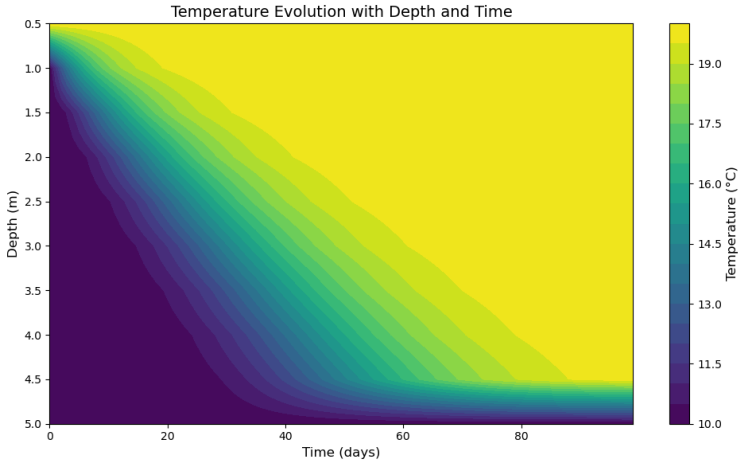

Abbildung 3.12.: Ergebnisse der analytischen Lösung für den Wärmeübergang aus der Atmosphäre nach Händel et al., 2013

```
    """Solves 1D heat transport through multiple soil layers.

    Parameters:
    -----------
    T : ndarray
        Initial temperature matrix (n_time x n_layer) [K]
    n_time : int
        Number of time steps
    n_layer : int
        Number of soil layers
    d : ndarray
        Layer thicknesses [m]
    q : ndarray
        Percolation water flux [m/s]
    Lamda : float
        Thermal conductivity [W/m K]
    c : float
        Volumetric heat capacity [J/m^3 K]
    r : float
        Bulk density [kg/m^3]
    rw : float
        Water density [kg/m^3]
    cw : float
        Water heat capacity [J/kg K]
    dt : float
        Time step [s]
    max_iter : int
        Maximum iterations for convergence
    max_error : float
        Convergence threshold [K]

    Returns:
    --------
    T : ndarray
        Temperature distribution [K]

    Assumptions:
    Transport parameters of the unsaturated zone are vertically and
    horizontally homogeneously distributed there is no significant
    horizontal transport in the unsaturated zone soil surface
    temperature is uniformly distributed for the complete model area
    """
```

Listing 3.6: Implementierung der analytischen Lösung für den Wärmeübergang aus der Atmosphäre nach Händel et al., 2013 – Beschreibung der Variablen

```
    Phi = np.zeros((n_layer, n_layer))
    for i in range(1, n_time):
        for j in range(1, n_layer - 1):
            # Calculate conductance between layers
            Phi[j, j + 1] = Lamda / (d[j] / 2 + d[j + 1] / 2)
            Phi[j - 1, j] = Lamda / (d[j - 1] / 2 + d[j] / 2)

            # Heat flux components
            Pc_01 = q[i] * rw * cw * (T[i - 1, j - 1] - T[i - 1, j])
            PD_01 = Phi[j - 1, j] * (T[i - 1, j - 1] - T[i - 1, j])
            PD_12 = Phi[j, j + 1] * (T[i - 1, j] - T[i - 1, j + 1])

            # Temperature update
            dT_ij = (Pc_01 + PD_01 - PD_12) * dt / (c * r * d[j])
            T[i, j] = T[i - 1, j] + dT_ij
    return T
```

Listing 3.7: Implementierung der analytischen Lösung für den Wärmeübergang aus der Atmosphäre nach Händel et al., 2013 – Algorithmus

3.4. AP 2.2 – Praxisnaher Anwendungsfall zur Evaluierung – Die Fallstudie Kleinbasel

3.4.1. Einleitung und Standortbeschreibung

Als Anwendungsfall wurde ein urban geprägter Teil des Schweizer Kantons Basel-Stadt (ca. 32 km²) ausgewählt; Grund hierfür ist das Vorhandensein eines gekoppelten Grundwasserströmungs- und Wärmetransportmodells (Epting et al., 2013, Mueller et al., 2018) basierend auf Feflow© (Diersch, 2014). Neben einer Vielzahl an natürlichen thermischen Randbedingungen (Fluss, Atmosphäre, geothermischer Gradient) und offensichtlichen anthropogenen Einflüssen (also geothermische Anlagen) enthält das Modellgebiet auch weitere thermisch relevante Strukturen, wie beispielsweise Tiefgaragen, Tunnel und Keller. Daher eignet es sich sehr gut für die Validierung der Entwicklungen in ModSimple – Phase 2. Das Modell wurde durch die Arbeitsgruppe „Angewandte und Umweltgeologie“ der Universität Basel, vertreten durch den wissenschaftlichen Beirat PD Dr. Jannis Epting, zur Verfügung gestellt.

Im Rahmen der Arbeiten erfolgte auch eine Ertüchtigung des Modells (Binder et al., 2025), um eine optimale Nutzung sicherzustellen. Nachfolgend sei der Standort stichpunktartig beschrieben: Der oberflächennahe Kiesgrundwasserleiter ist bis zu 38 m mächtig und besteht hauptsächlich aus fluvialen Lockersedimenten, die von den lokalen Flüssen (Rhein, Birs, Wiese und Birsig) abgelagert wurden. Dieser hoch durchlässige Kiesgrundwasserleiter wird von schluff- und tonreichen Sedimenten unterlagert (Aquiclude), die als hydraulisch irrelevante, aber thermisch relevante Systeme fungieren (Huggenberger & Epting, 2011). Die Modelloberkante wurde direkt aus digitalen Höhenmodellen (2 × 2 m² Auflösung) und Echoloten des Rheins abgeleitet, während das Übergangsniveau zwischen dem Aquifer und der Aquiclude auf einem bestehenden geologischen Modell (Dresmann et al., 2013) basiert. Die Daten zum Grundwasserspiegel und zur Grundwassertemperatur stammen aus dem Langzeitüberwachungsprogramm der Kantone Basel-Stadt und Basel-Landschaft. Die Flusspegel und -temperaturen des Rheins wurden vom Bundesamt für Umwelt (BAFU) und teilweise vom Tiefbauamt Basel-Landschaft (TBA BL) beigesteuert. Basel-Stadt hat ein gemäßigtes ozeanisches Klima (Mueller et al., 2018) mit monatlichen Lufttemperaturen zwischen -2,2 und 23,2 °C (Mittelwert: 11,3 °C) und gilt als städtische Wärmeinselregion (unter anderem Mueller et al., 2018). Der mittlere Jahresniederschlag beträgt rund 820 mm.

3.4.2. Vorbereitung des Transfers von FEFLOW nach MODFLOW

Das eigentliche Modellgebiet umfasst den Großteil des Regionalmodells für den Bereich „Kleinbasel", welches rechtsseitig des Rheins liegt. Die Abbildung 3.14 zeigt den zugehörigen Kartenausschnitt. Basierend auf dieser Begrenzung wurden Modellinformationen aus dem FEFLOW-Modell ausgeschnitten. Dies umfasste insbesondere:

- grundlegende Modellgeometrie, das heißt Modell-Oberfläche, Modell-böden, Anzahl und Lage der Berechnungslayer
- Form des Berechnungsgitters (Lage der Nodes und FE-Elemente)
- hydraulische Leitfähigkeiten in allen Raumrichtungen (Kx, Ky, Kz)
- grundlegende Transportparameter wie Porositäts- und Dispersionswert
- thermisch relevante Transportparameter wie Wärmeleitfähigkeit und Wärmekapazität

Abbildung 3.13.: Fallstudie – Lage der Stadt Basel in der Schweiz (Kartenquelle: Medienarchiv Wikimedia Commons, Basel nachträglich rot hervorgehoben)

- thermische und hydraulische Randbedingungen inklusive regionaler Zufluss, geothermischer Gradient, Austausch mit der Atmosphäre (Lufttemperaturen, Austauschraten); Fluss-Grundwasser-Interaktionen (Flusstemperaturen und Wasserstände; Austauschraten), punktuelle Nutzerdaten (unter anderem Entnahmeraten an geothermischen Anlagen inklusive deren Raumkoordinaten)
- Beobachtungsdaten an Messpunkten sowie deren Positionen (offizielle Grundwasser-beobachtungsmessstellen im Stadtgebiet und Umland)

Das ausgewählte MODFLOW-Modellgebiet ist hierbei etwas kleiner als das bestehende FEFLOW-Modellgebiet, sodass Randbedingungsinformationen zum Teil neu definiert werden mussten. Hierzu wurden Modellsimulationsdaten von FEFLOW im Randbereich des Ausschnittes ausgelesen und dann als neue Randbedingungen des regionalen Zustroms (Dirichlet-Randbedingung) interpretiert. Das verkleinerte Modellgebiet enthält alle für das hiesige Projekt interessanten Randbedingungsarten. Technisch erfolgte die Übertragung zum einen über standardisierte Geoobjekte (GIS-Shapefiles), zum anderen über tabellarisch strukturierte ASCII-Textdateien. Hierfür wurde die in FEFLOW implementierte Export-Funktion genutzt. Im Anschluss erfolgte dann eine Umwandlung in MODFLOW-kompatible Informationen mittels einer Reihe an Python-Skripten. Diese zusätzliche Umwandlung wurde notwendig, da FEFLOW auf einer kombinierten Node- und Element-basierten Informationsspeicherung basiert, MODFLOW hingegen auf einer rein Element-basierten Variante.

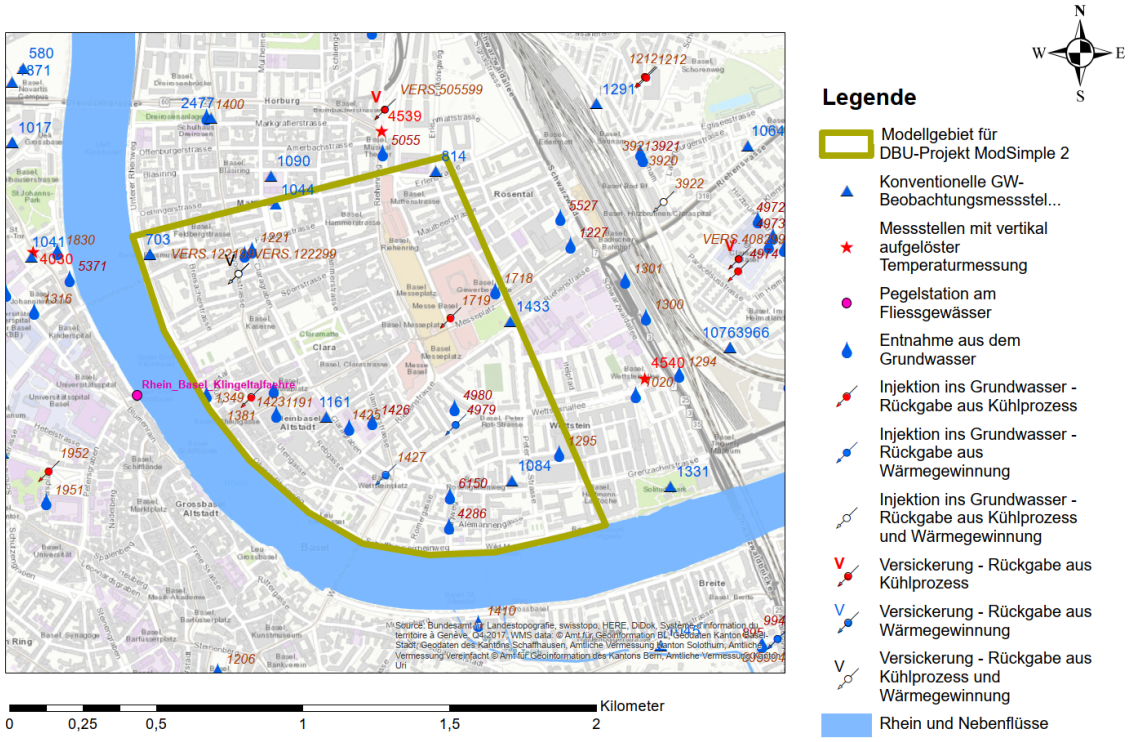

Abbildung 3.14.: Fallstudie Kleinbasel – Modellausschnitt (Quelle Geodaten: Forschungsgruppe „Angewandte und Umweltgeologie – AUG“ der Universität Basel)

3.4.3. Automatisierter Aufbau des MF6-Modells mit `feflowtomf6`

Die Umwandlung des FEFLOW-Modells in ein MF6-Modell ist aufgrund der teilweise sehr unterschiedlichen Modell-Konzepte und Datenstrukturen komplex. Daher ist eine manuelle Umwandlung der Informationen aus FEFLOW in MF6-Eingabe-Dateien nicht praktikabel. Das hier neu entstandene Python-Paket `feflowtomf6` automatisiert das Einlesen der FEFLOW-Daten, die Umwandlung und den Abgleich der Datenstrukturen und das Erzeugen der MF6-Eingabe-Dateien. Derzeit enthält `feflowtomf6` 33 Python-Dateien mit insgesamt circa 4500 Zeilen objekt-orientiertem Code. Die Steuerung der Erstellung der MF6-Eingabe-Dateien läuft über eine umfangreiche Datei im `ini`-Format. Die Angaben sind in Gruppen eingeteilt, die Pfade zu Eingabe- und Ausgabe-Dateien sowie viele andere Steuerinformationen enthalten.

Es hat sich gezeigt, dass für die Entwicklung des Konvertierungswerkzeugs viele Modell-Läufe nötig sind. Die Entwicklung erfolgte mit schrittweiser Hinzunahme von Modelldaten. Die meisten der Steuervariablen, wie die Pfade der Datenquellen und die relativen Pfade der MF6-Eingabe-Dateien, ändern sich zwischen den Modellläufen nicht. Daher liegen die Steuerdaten in zwei Dateien vor: eine Basis-Steuerdatei hat circa 225 Zeilen mit allen Angaben und eine Projekt-Steuer-Datei mit den für den jeweiligen Modelllauf spezifischen Informationen.

Listing 3.8 zeigt ein Beispiel für eine Projekt-Steuer-Datei. Bei einem Modelllauf vereinigt `feflowtomf6` beide Dateien, wobei die Angaben in der Projekt-Steuer-Datei die in der Basis-Steuer-Datei überschreiben. Durch den Einsatz von Interpolation lassen sich Pfade flexibel deklarativ angeben. Listing 3.9 zeigt einen Ausschnitt aus einer Basis-Steuer-Datei. Die Variable `root_base_path` enthält einen absoluten Pfad, den der Eintrag

```
# Model without river control
# River SSM is off
# No Buildings CTP

[names]
model_name = esl_narea

[options]
riv_ssm = False
ctp = False
set_river_temps = False
river_heat_transfer_area = False

[files]
explanation_file = esl_control_ctp_off_no_area.md
```

Listing 3.8: Steuerungsdatei für einen MF6-Lauf mit Umwandlung von FEFLOW-Daten

`model_path = ${root_base_path}/models/${names:model_name}` nutzt. Hier kommt `names:model_name` aus dem Abschnitt `[names]` aus der Projekt-Steuer-Datei. Dies funktioniert, weil die Interpolation erst nach dem Zusammenfügen beider Steuerdateien erfolgt.

```
[paths]
root_base_path = /absolute/path/to/Modellgebiet_Basel/
...
model_path = ${root_base_path}/models/${names:model_name}
```

Listing 3.9: Ausschnitt aus der Basis-Steuerungsdatei für die Umwandlung von FEFLOW-Daten

Listing 3.10 zeigt die programmtechnische Umsetzung der Vereinigung der Steuerdateien. Dieses sich an der Vererbung orientierende Vorgehen macht die Nutzung der Steuerdateien für viele, nur leicht voneinander abweichende Szenarien wesentlich praktikabler als das Kopieren aller Steuerdaten für ein Szenario.

```
config = Config(
    config_file_name='config_esl_control_ctp_off_no_area.ini',
    base_config_file_name='base_config.ini')
```

Listing 3.10: Vereinigung der Konfigurationsinformationen

Während der schrittweisen Entwicklung von `feflowtomf6` war es nötig, die Daten sehr häufig einzulesen. Die Umwandlung der großen Datenmengen nimmt Zeit in Anspruch. Um die Laufzeiten zu reduzieren, kam Caching zum Einsatz. Durch Zwischenspeichern von teil-verarbeiteten Daten in effizienten Formaten wie HDF5 und in einer Sqlite3-Datenbank hat sich die Laufzeit für viele Aufgaben von 30 und mehr Minuten auf einige Sekunden reduziert.

Die Entwicklung von `feflowtomf6` erfolgte schrittweise. Wichtige Entwicklungsphasen für einen Schritt waren:

- Programmierung von Leseroutinen für FEFLOW-Daten-Formate
- Umwandlung der gemischt knoten- und element-basierten in rein element-basierte Daten

- Erzeugung der MF6-Modell-Geometrie
- Export der Randbedingungen für das MF6-Strömungsmodell
- Export der Randbedingungen für das MF6-Wärmetransportmodell

Unterstützung der Modellauswertung

Im Lauf der Umwandlung des FEFLOW-Modells in ein MF6-Modell waren viele Modellläufe nötig. Durch das Ein- oder Ausschalten von einzelnen Randbedingungen, das Belegen von Zeitreihen mit konstanten Werten und anderen Variationen ist es möglich, die entscheidenden Einflüsse auf die Modellergebnisse zu untersuchen. Dieses Vorgehen führt zu vielen Modell-Szenarien, deren Ergebnisse ausgewertet werden müssen. Dabei ist immer ein Vergleich mit den Ergebnissen der FEFLOW-Rechnung nötig. Als Zielwerte kommen vor allem die Grundwasserstände und -temperaturen in Frage.

Abbildung 3.15 zeigt die mit `feflowtomf6` erzeugte HTML-Datei mit der Übersicht von Ergebnissen für ein Modell-Szenario. Die MF6-Rechnung und die Erzeugung der Datei erfolgt mit einem Befehl ohne weitere Nutzer-Interaktion. Damit lassen sich mehrere Rechnungen gleichzeitig starten, deren Ergebnisse dann sofort aufbereitet zur Auswertung zur Verfügung stehen.

Der erklärende Text stammt aus der Markdown-Datei, die im Eintrag `explanation_file` in der Sektion `[files]` der in Listing 3.8 gezeigten Projekt-Steuer-Datei steht. Das Markdown-Format erlaubt es, die Erläuterungen mit Tabellen und Abbildungen anzureichern. Die acht aufgelisteten Ergebnisse sind Links zu Animationen.

Abbildung 3.15.: Ergebnisübersicht für ein Modell-Szenario, Links führen zu Animationen mit räumlich verteilten Werten

Abbildung 3.16 zeigt ein Beispiel für eine dieser Animationen für eine der 15 Schichten. Jede Animation besteht aus 21 Zeitschritten und zeigt die Werte in Intervallen von 90 Tagen für den gesamten Modellzeitraum von 1825 Tagen. Diese Intervalle wurden gewählt, weil die Ergebnisse des FEFLOW-Modells in 90-Tage-Schritten zur Verfügung standen. Die Animationen sind für die Auswertung äußerst nützlich, da sie viele Informationen kondensiert, aber trotzdem verständlich darstellen können. Um die gleichen Informationen mit statischer Darstellung auszuwerten, müsste sich ein Modellierer für ein Szenario

2520 Abbildungen ansehen (8 Ergebnis-Arten × 15 Schichten × 21 Zeitschritte = 2520 Abbildungen). Das ist nicht praktikabel.

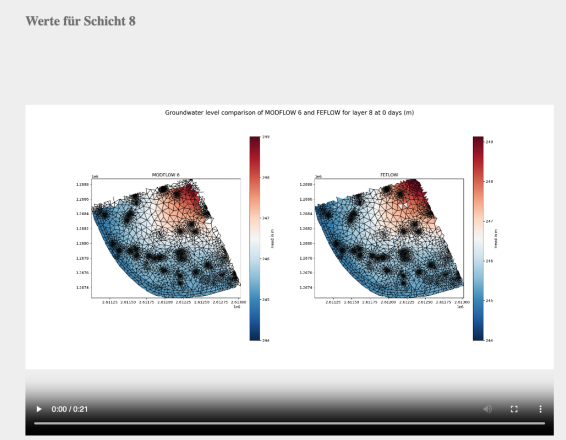

Abbildung 3.16.: Beispiel für eine Animation, hier Gegenüberstellung des zeitlichen Ablaufs der Grundwasserstände im FEFLOW- und MF6-Modell

3.4.4. Gegenüberstellung der in FEFLOW und MODFLOW realisierten Grundwasserströmungsmodelle

Eigenschaften des Strömungsmodells in MODFLOW 6

Tabelle 3.6 enthält alle MF6-Pakete, die für das Strömungsmodell nötig sind. Tabelle 2 in Anhang C.1 stellt alle für das erstellte Modell verwendeten MF6-Pakete und die konkrete Nutzung zusammen. MF6 arbeitet mit sogenannten Stress-Perioden. Für jede Änderung von Randbedingungswerten erforderten ältere MODFLOW-Versionen eine neue Stress-Periode. MF6 unterstützt dieses Konzept weiterhin. Alternativ ist es aber möglich, mit Zeitreihen (time series) zu arbeiten. Dieser Ansatz kam hier zum Einsatz. Das Modell hat eine Stress-Periode mit 1825 Tagen. Die Werte für die Randbedingungen für vorgegebene Grundwasserstände (CHD), Flusswasserstände (RIV), Grundwasser-Neubildung (RCH) und die Brunnenentnahmen (WEL) sind als Zeitreihen mit täglichen Werten in das Modell eingegangen. Für das Wärmetransport-Modell haben alle diese Randbedingungen tägliche Temperaturwerte.

Abbildung 3.17 zeigt die räumliche Verteilung der Randbedingungen. Die Diskretisierung ist ebenfalls dargestellt. Das Modell hat am nördlichen und östlichen Rand zahlreiche inaktive Zellen. Insgesamt sind von den 143.865 Modell-Zellen, die beim Ausschneiden des Teilmodells entstanden sind, 28.920 inaktiv. Das entspricht circa 20 Prozent. Allerdings sind viele der inaktiven Zellen klein. Die inaktiven Zellen waren nötig, da die Export-Routinen für verschiedene Modell-Eigenschaften leicht unterschiedliche Begrenzungen anwenden. Daher war es nötig, die Umhüllende des Gebietes zu nutzen, in dem für alle Zellen alle Eigenschaften vorliegen. Die äußeren Randbedingungen mussten entsprechend innerhalb dieser Umhüllenden liegen. Daraus resultiert die Lage der CHD-Randbedingungen. Die Brunnen liegen in sehr kleinen Zellen. Daher ist um jeden Brunnen ein Kreissymbol mit Beschriftung angebracht, das wesentlich größer als die Modellzelle der Randbedingung WEL ist. Das Modell hat 15 Schichten mit jeweils 9591 Zellen, von denen jeweils 7663 aktiv sind.

Tabelle 3.6.: MF6-Pakete für das Strömungsmodell

Akronym	Package	Nutzung
CHD	Constant-Head	räumliche Zuordnung der Grundwasserstands-Zeitreihen am Modellrand
DIS	Structured Discretization	räumliche Diskretisierung in Dreieckselemente, gleich für alle Modell-Schichten
IMS	Specified Complexity	Modell-Komplexität „complex“ (lineare und nicht-lineare Variablen für die Gleichungslösung)
NAM	Simulation Name File	zentrale Steuerdatei für Simulation
NPF	Node Property Flow	räumliche Verteilung de hydraulischen Leitfähigkeit des Grundwassers
OC	Output Control	Spezifikation der Speicherzeitpunkte von Wasserstand, Temperatur und Volumenbilanzen
RCH	Recharge	räumliche Zuordnung der Grundwasser-Neubildungs-Zeitreihen am oberen Modellrand
RIV	River	räumliche Zuordnung der Wasserstands-Zeitreihen des Rheins sowie Höhe und Durchlässigkeit der Flusssohle
STO	Storage	konstante Werte für den spezifischen Speicherkoeffizienten und die effektive Porosität
TDIS	Temporal Discretization	eine Stress-Periode, zeitliche Variabilität durch Zeitreihen von Randbedingungswerten
WEL	Well	räumliche Zuordnung der Entnahme-Raten-Zeitreihen der Brunnen (negative Werte bedeuten Entnahme, positive Werte Infiltration)

Vergleich der Simulationsergebnisse

Abbildung 3.18 zeigt einen Vergleich der mit MF6 und FEFLOW berechneten Grundwasserstände für die Modellschicht 8 zum Ende des Berechnungszeitraums. Die Abweichungen sind gering. MF6 kann die Berechnungsergebnisse von FEFLOW innerhalb zu erwartender Abweichungen reproduzieren.

Abbildung 3.19 zeigt die Differenz der berechneten Grundwasserstände beider Modelle für die gleiche Schicht und den gleichen Zeitpunkt wie Abbildung 3.18. Es gibt einige Zellen mit größeren Abweichungen. Diese sind wahrscheinlich auf numerische Abweichungen in der FEFLOW-Rechnung zurückzuführen, da die Wasserstände nicht physikalisch begründet sind.

3.4.5. Gegenüberstellung des Wärmetransportmodells zwischen FEFLOW und MODFLOW

Eigenschaften des Wärmetransportmodells in MODFLOW 6

Das Wärmetransportmodell baut auf dem Strömungsmodell auf und nutzt die gleiche räumliche und zeitliche Diskretisierung.

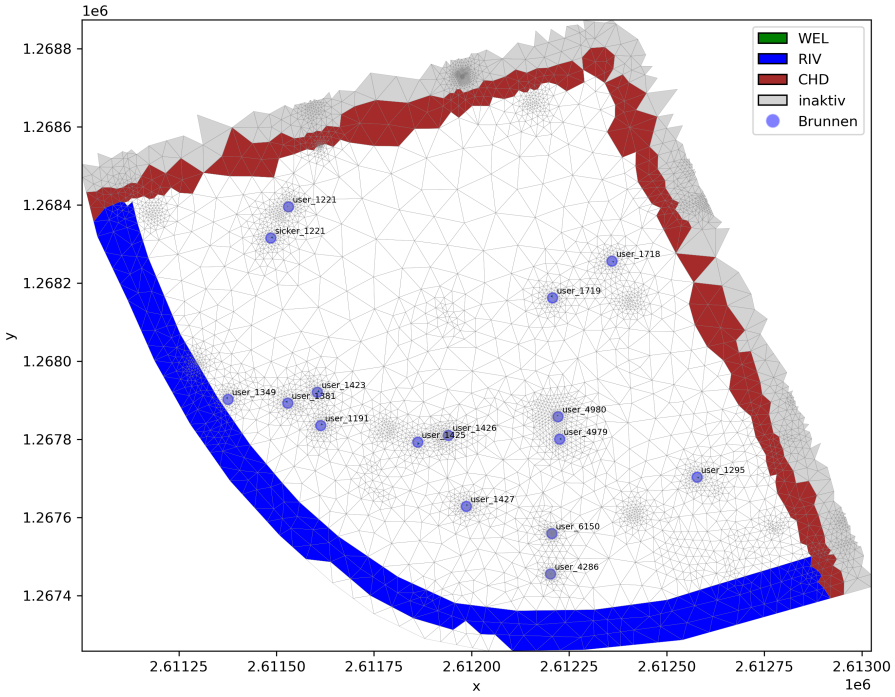

Abbildung 3.17.: Diskretisierung und Randbedingungen des MF6-Modells für Kleinbasel

Tabelle 3.7 enthält alle MF6-Pakete, die für das Wärmetransportmodell nötig sind. Tabelle 2 stellt alle für das erstellte Modell verwendeten MF6-Pakete und die konkrete Nutzung zusammen.

Vergleich der Simulationsergebnisse

Abbildung 3.20 zeigt einen Vergleich der mit MF6 und FEFLOW berechneten Temperaturverteilungen des Grundwassers für die Modellschicht 8 zum Ende des Berechnungszeitraums. Die Abweichungen sind beträchtlich. MF6 kann die Berechnungsergebnisse von FEFLOW nicht in akzeptablen Grenzen reproduzieren.

Abbildung 3.21 zeigt die Differenz der berechneten Temperaturverteilungen des Grundwassers beider Modelle für die gleiche Schicht und den gleichen Zeitpunkt wie Abbildung 3.20. Hier zeigen sich ebenfalls beträchtliche Abweichungen. Die Abweichungen in den Ergebnissen sind auf Unterschiede in der Abbildung der Prozesse in beiden Modellen zurückzuführen.

Es sind also weitere Anpassungen nötig. Der folgende Abschnitt beschreibt diese und geht auf die Besonderheiten der Randbedingungen ein.

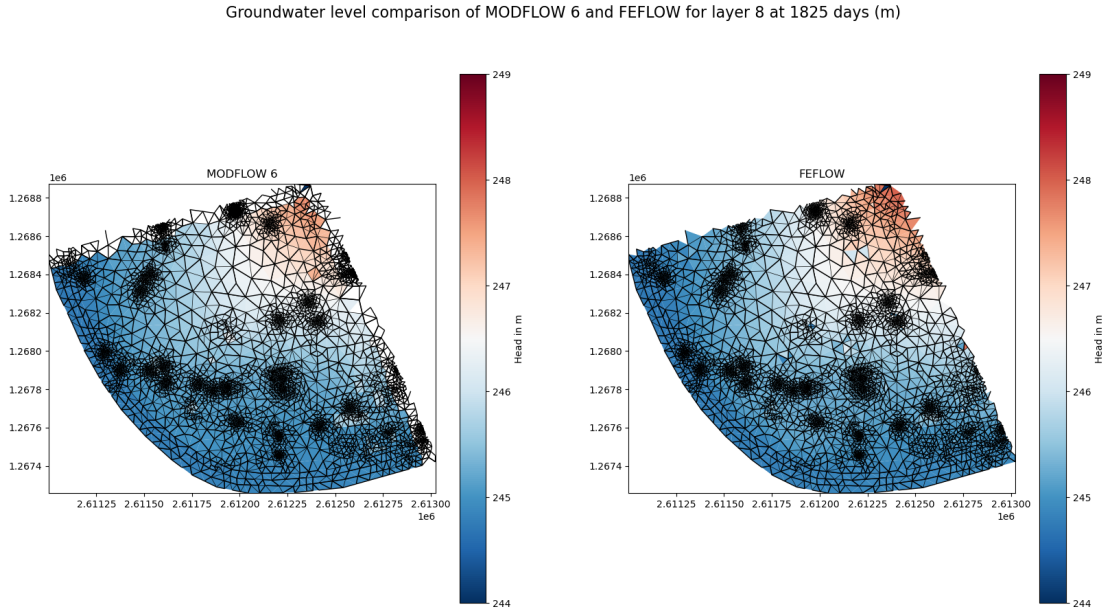

Abbildung 3.18.: Vergleich der Grundwasserstände der Rechnungen mit FEFLOW und MF6-Modell, Modellschicht 8, Zeitschritt Tag 1825

3.4.6. Technische Erläuterungen zur Implementierung einer thermischen Cauchy-Randbedingung im Groundwater Energy Module von MODFLOW 6

Arten von Randbedingungen

Randbedingungen (RB) sind spezielle Merkmale eines numerischen Modells, die für die Lösung der Randwertprobleme eben dieser Modelle erforderlich sind. Ein Randwertproblem ist im Allgemeinen ein System von Differentialgleichungen, das in einem Bereich gelöst werden muss, an dessen Rand eine Reihe von Bedingungen bekannt ist. Eine Simulation von Grundwasserströmung und/oder Transport im Grundwasser ohne Definition von mindestens einer RB ist zwar technisch unter Umständen möglich, liefert aber uneindeutige Ergebnisse. Das korrekte Setzen von RB unter Berücksichtigung des realen Verhaltens ist daher essenziell. An der Grenze der Modellregion können verschiedene Arten von RB definiert werden (Hinweis: innerhalb der Modellregion spricht man von Quellen und Senken). Im Grundwasserbereich werden insbesondere die Dirichlet-, die Neumann- und die Cauchy-RB angewendet. Die Dirichlet-RB (auch als RB 1. Art bekannt) gibt den Wert einer Prozessvariablen entlang der Grenze des Modellbereichs an, beispielsweise den Wert der Grundwassertemperatur und/oder des Grundwasserspiegels. Die Neumann-RB (RB 2. Art) gibt die Werte an, die die Ableitung der Prozessvariablen an der Grenze des Modells annimmt. Beispiele hierfür sind der Wasser- und/oder Stofffluss in den/aus dem Bereich. Ein Sonderfall ist die No-Flow-RB (also die normale Modellgrenze ohne besondere Definition einer RB), die ebenfalls eine Neumann-RB ist. Die Cauchy-RB (RB 3. Art) ist schließlich eine mathematische Bedingung sowohl für die obigen Prozessvariablen als auch für deren Ableitungen. Sie impliziert dort die Auferlegung von zwei RB gleichzeitig (Cauchy-RB = Dirichlet-RB + Neumann-RB). Es sei darauf hingewiesen, dass beispielsweise Brunnen-Objekte, obwohl innerhalb der Modellregion platziert, trotzdem als RB angesehen werden können, da diese durch die beispielsweise Wasserentnahme oder -injektion einen Übergang zwischen „innerhalb des Modells“ und

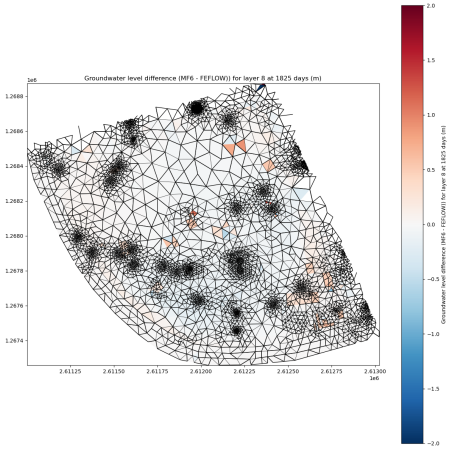

Abbildung 3.19.: Unterschiede der Grundwasserstände der Rechnungen mit FEFLOW und MF6-Modell, Modellschicht 8, Zeitschritt Tag 1825

„außerhalb des Modells“ erzeugen.

Bisherige Einschränkungen seitens MODFLOW 6 im Bereich thermischer Randbedingungen

Die Fallstudie Basel nutzt eine Vielzahl an Randbedingungen sowohl für die Strömungssimulation als auch für die Darstellung des Wärmetransportes. Hierzu zählt beispielsweise der thermische Austausch zwischen den Oberflächengewässern (insbesondere der Rhein als systemkontrollierendes hydrologisches Objekt, aber auch dessen Nebenflüsse Wiese, Birs und Birsig) und dem Untergrund. Der Austausch erfolgt sehr ähnlich zum Prozess des hydraulischen Austausches (also Wasserfluss) durch die geringdurchlässige Kolmationsschicht, wie dieser in MODFLOW 6 auch integriert ist. Der Unterschied liegt in der Prozessvariablen sowie den entsprechend anderen thermischen statt hydraulischen Materialeigenschaften. MODFLOW 6 inklusive GWE unterstützt den thermischen Austausch nur indirekt (zum Zeitpunkt des Berichtes) – sowohl thermische Dirichlet-RB als auch Neumann-RB sind möglich und können auch allgemein eingesetzt werden. Es können also feste Grundwassertemperaturen (auch zeitlich variabel) definiert werden als auch bekannte Wärmeströme. Darüber hinaus sind auch eine Art von Cauchy-RB möglich, jedoch nur für ausgewählte Teilpackages von GWE; eine allgemeine Nutzung hingegen nicht. Für einen vollständigen Transfer des FEFLOW-Modells nach MODFLOW-6 ist diese Art von Randbedingung unabdingbar. Somit ist es notwendig, hierfür einen technischen Workaround zu schaffen. Dieser Workaround muss es ermöglichen, analog zu den Möglichkeiten von FEFLOW:

1. eine Zeitreihe an Oberflächenwasser-Temperaturen eingeben zu können,
2. die im Modell vorliegende, aktuell berechnete Grundwassertemperatur während des Modelllaufs auszulesen,
3. darauf basierend den Wärmestrom zwischen Oberflächenwasser und Grundwasser zu berechnen, und
4. diesen Wärmestrom während des aktuellen Modelllaufs direkt anzuwenden.

Tabelle 3.7.: MF6-Pakete für das Wärmetransportmodell

Akronym	**Package**	**Nutzung**
ADV	Advection	Lösungsschema für Advektion „upstream“
CND	Conduction and Dispersion	konstante Werte für Dispersion und thermische Leitfähigkeit des Wassers, konstanter Wert pro Modellschicht für die thermische Leitfähigkeit des Grundwasserleitermaterials
CTP	Constant Temperature	räumliche Zuordnung der Grundwassertemperatur-Zeitreihen am Modellrand
ESL	Energy Source Loading	konstante Energiequelle am unteren Modellrand, bei einigenTest-Szenarien konstante Energiequelle aus dem Rhein
EST	Energy Storage and Transfer	Werte für das Grundwasserleitermaterials, konstante Porosität und Dichte, konstanter Wert für obersten 11 und unterste 4 Modellschicht für die Wärmekapazität
GWFGWE	Exchanges	Kopplung der Strömungs- und Wärmetransportmodelle
IC	Initial Conditions	räumliche Verteilung der Anfangstemperaturen des Grundwassers
IMS	Specified Complexity	Modell-Komplexität „complex“ (lineare und nicht-lineare Variablen für die Gleichungslösung)
OC	Output Control	Spezifikation der Speicherzeitpunkte von Wasserstand, Temperatur und Volumenbilanzen
SPC	Stress Pack. Component 1	räumliche Zuordnung der Temperatur-Zeitreihen der Infiltrations-Brunnen
SPC	Stress Pack. Component 2	räumliche Zuordnung der Temperatur-Zeitreihen der Grundwasserneubildung
SPC	Stress Pack. Component 3	räumliche Zuordnung der Temperatur-Zeitreihen des Rheins
SSM	Source and Sink Mixing	Zuordnung zu den SPC-Dateien zu den Randbedingungsnamen

Dies wurde im Projekt unter Zuhilfenahme des zuvor beschriebenen Kopplungsansatzes verwirklicht (nächster Abschnitt). Identisch ist zu verfahren bei der thermischen Implementierung von Gebäuden sowie des Austausches mit der Atmosphäre.

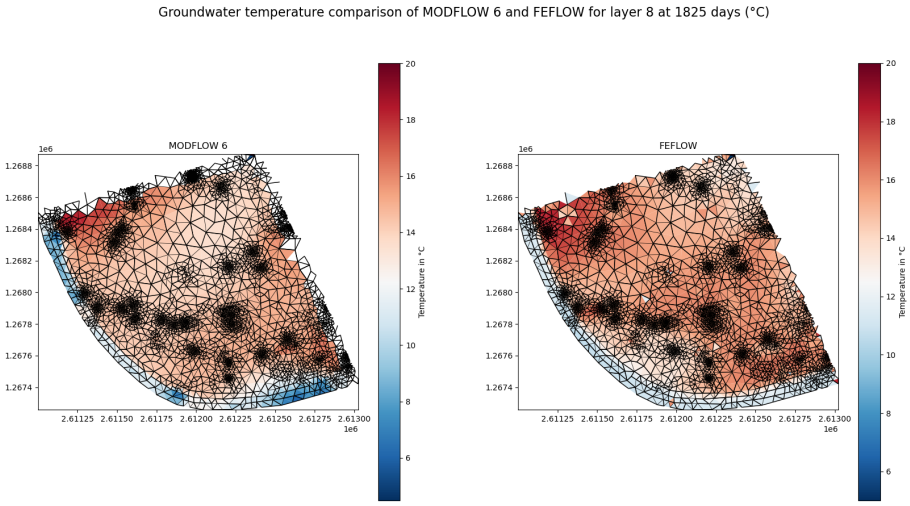

Abbildung 3.20.: Vergleich der Grundwassertemperaturen der Rechnungen mit FEFLOW und MF6-Modell, Modellschicht 8, Zeitschritt Tag 1825, ohne Implementierung / Anpassung der Cauchy-RB

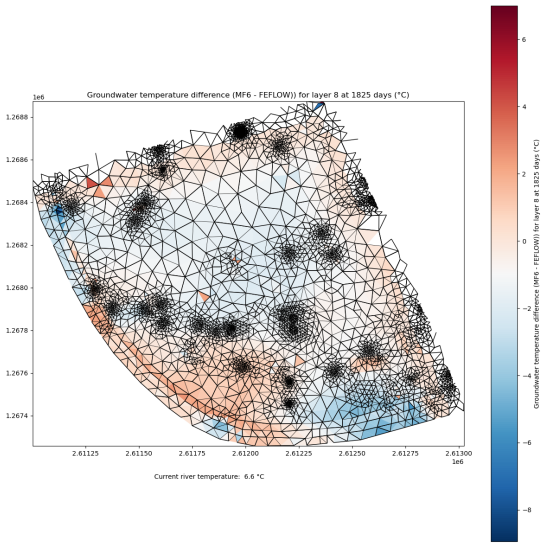

Abbildung 3.21.: Unterschiede der Grundwassertemperaturen der Rechnungen mit FEFLOW und MF6-Modell, Modellschicht 8, Zeitschritt Tag 1825, ohne Implementierung / Anpassung der Cauchy-RB

Implementierung einer thermischen Cauchy-Randbedingung mit `pymf6`

Listing 2 in Anhang C.3 zeigt die vollständige Implementierung der thermischen Cauchy-Randbedingung am Beispiel der thermischen Berücksichtigung des Flusses Rhein. Listing 3.11 zeigt den Beginn der Definition der Klasse `RiverController`. Wichtig ist hier die Übergabe des Namens der Randbedingung mit `bc_name='riv_0'`. Das zentrale Element ist die Instanz der Klasse `pymf6.mf6.MF6`, die als `self.mf6` gespeichert ist.

```
class RiverController:
    """River controller for temperature transfer with CAUCHY BC."""

    def __init__(self, config, mf6, bc_name='riv_0'):
        self.config = config
        self.mf6 = mf6  # instance of pymf6.mf6.MF6
        self.bc_name = bc_name
```

Listing 3.11: Implementierung der Cauchy-RB für den Wärmetransport mit `pymf6` – Nutzung von `pymf6.mf6.MF6`

Der Austausch der Wärme zwischen Fluss-Randbedingung und Grundwasser erfolgt in der Methode `set_heat_flux`, die in der Klasse `RiverController` definiert ist. Listing 3.12 zeigt diese Methode. Zum Testen ist es möglich, eine feste Rate `rate` zu nutzen. Wenn keine Rate übergeben wird, wird diese aus der Temperatur-Differenz zwischen Fluss und Grundwasser sowie der Wärme-Transfer-Rate berechnet. Die räumlich verteilten Werte für diese Rate stammen aus den Daten des FEFLOW-Modells. Diese sind von der Austauschrichtung abhängig. Je nachdem, ob das Wasser vom Grundwasser zum Fluss oder umgekehrt fließt, kommt ein anderer Wert zum Einsatz. Die Rate überschreibt dann die Werte des Pakets ESL (Energy Source Loading) für die Flusszellen.

```
    def set_heat_flux(self, rate=None, rate_factor=1):
        """
        Set new heat flux value for river cells.

        rate = phi x (T_riv - T_gw)
        rate can be supplied as constant for testing purposes.
        """
        if rate is None:
            t_gw = self.gwe.X[self.riv_layers, self.river_cell_ids]
            rate = (
                self.current_heat_transfer
                * (self.river_temperature - t_gw)
            )
            if self.muliply_area:
                rate *= self.areas
        values = self.gwe.esl.get_advanced_var('BOUND')
        values[self.river_coords_mask, 0] = rate * rate_factor
        self.gwe.esl.set_advanced_var('BOUND', values)
```

Listing 3.12: Implementierung der Cauchy-RB für den Wärmetransport mit `pymf6` – Wärmeaustausch

Listing 3 in Anhang C.3 zeigt das vollständige Programm zur Nutzung der Implementierung der thermischen Cauchy-Randbedingung. Der entscheidende Teil ist in Listing 3.13 nochmals dargestellt. Bei der Instanziierung von `MF6` schaltet `do_solution_loop=False` den Durchlauf der Lösungsschleife aus. Dies ist derzeit noch nötig, um das in Abschnitt 3.1.2 beschriebene Problem der Nicht-Berücksichtigung von Zeitschritt-Eingaben

zu vermeiden. Nach der Instanziierung von `RiverController` erfolgt der Austausch in der Zeitschritt-Schleife bei jedem Zeitschritt-Beginn `model.state == states.timestep_start`.

```
def run_controlled(
    model_name,
    base_config_file_name=BASE_CONFIG_FILENAME,
    control_transfer=True,
    constant_rate=None,
    rate_factor=1,
):
    """Run MF6 controlled."""
    config_file_name = _make_config_file_name(model_name)
    config = Config(
        config_file_name, base_config_file_name=base_config_file_name
    )
    start = default_timer()
    mf6 = MF6(sim_path=config.paths.model_path, do_solution_loop=False)
    loop = mf6.model_loop()
    riv_controller = RiverController(config, mf6)

    for model in loop:
        if model.state == states.timestep_start:
            if control_transfer:
                riv_controller.set_heat_flux(
                    constant_rate, rate_factor=rate_factor
                )
```

Listing 3.13: Nutzung der Cauchy-RB für den Wärmetransport mit `pymf6`

Einfluss der neuen thermischen Cauchy-Randbedingung

Die Berechnungen mit der thermischen Cauchy-Randbedingung ergaben Änderungen in den Ergebnissen. Die Implementierung dieser neuen Randbedingung konnte also die Funktionalität von MF6 erweitern. Abbildung 3.22 zeigt die Temperaturverteilung im Vergleich mit denen der FEFLOW-Simulation. In Abbildung 3.23 sind die gleichen Informationen als Differenz dargestellt. Abweichungen sind immer noch signifikant. Die Nutzung der thermischen Cauchy-Randbedingung nur für den Fluss reicht nicht aus, um mit der MF6-Wärmetransport-Rechnung die Ergebnisse von FEFLOW in für die praktische Anwendung sinnvollen Grenzen zu reproduzieren.

Sowohl die Gebäudestrukturen als auch der Kontaktbereich zwischen Atmosphäre und Untergrund sind ebenfalls mit einer Cauchy-Randbedingung auszustatten – dies ist in Vorbereitung. Die Implementierung kann in Anlehnung an die obige Definition der Fluss-Randbedingung erfolgen.

Weitere Verbesserung der Abbildung des Wärmetransports

Ein weiterer Unterschied zwischen FEFLOW und MF6 ist die Abbildung der ungesättigten Zone in Bezug auf den Wärmetransport. Die obersten Modellschichten sind ungesättigt. FEFLOW rechnet diese mit der Richards-Gleichung. In MF6 werden ungesättigte Schichten nicht berechnet. Die Grundwasser-Neubildung wird der obersten gesättigten Modellschicht zugewiesen. Damit wird die Temperatur des Bodenwassers sofort für das Grundwasser wirksam. Die zeitliche Verzögerung, die die ungesättigte Zone erzeugt, entfällt.

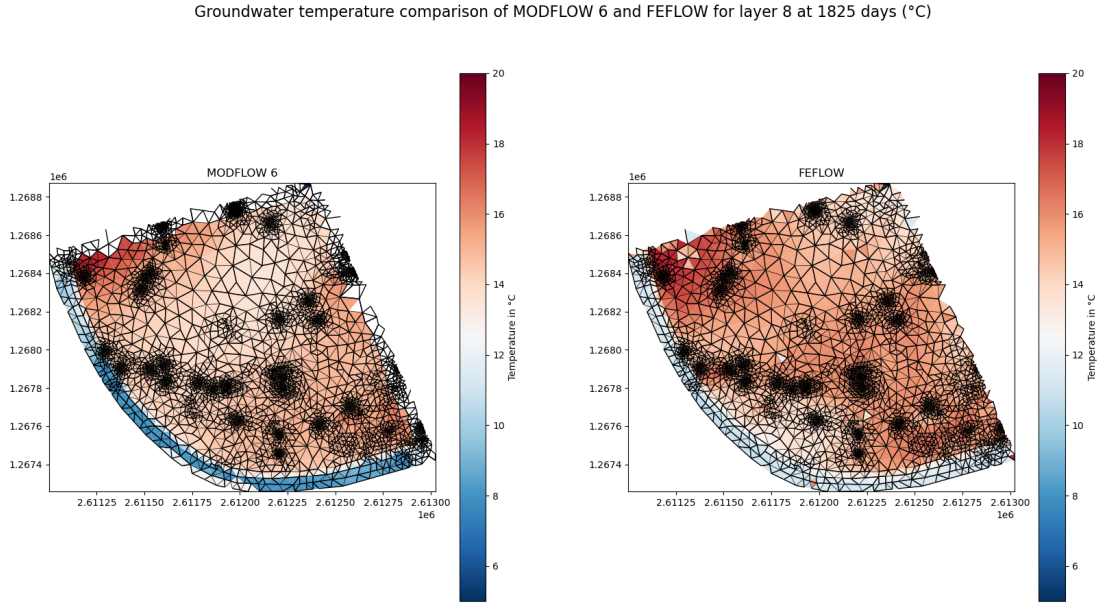

Abbildung 3.22.: Vergleich der Grundwassertemperaturen der Rechnungen mit FEFLOW und MF6-Modell mit thermischer Cauchy-Randbedingung, Modellschicht 8, Zeitschritt Tag 1825, mit Implementierung des Flusses als thermische Cauchy-RB.

Auf die Berechnung der Grundwasser-Strömung hat dieser Unterschied keinen signifikanten Einfluss, da die ungesättigte Zone die Neubildung zeitlich vergleichmäßigt. Beim Wärmetransport spielt die zeitliche Verteilung eine größere Rolle, da die Schwankungen der Lufttemperatur zwischen Sommer und Winter relativ hoch sind.

Hier kann die Kopplung der in Abschnitt 3.3.3 erläuterten analytischen Lösung zum Einsatz kommen. Die Kopplung ist derzeit in Vorbereitung. Der hier verfolgte Ansatz baut für jede MF6-Modellzelle, die Grundwasser-Neubildung erhält, ein eindimensionales, analytisches Modell auf, das den in Listing 3.7 gezeigten Algorithmus umsetzt. Prinzipiell ist dieser Ansatz mit `pymf6` gut umsetzbar. Da der zeitliche Aufwand der Umwandlung des FEFLOW-Modells in ein MF6-Modell wesentlich höher war als erwartet, konnte diese Kopplung der analytischen Lösung für den Wärmeübergang aus der Atmosphäre noch nicht fertiggestellt werden.

Darüber hinaus ist auch anzumerken, dass im Rahmen dieses Projektes nur ein Teilbereich von Kleinbasel modelliert wurde. Entsprechend andere Randbedingungspositionen werden im Vergleich zum größeren FEFLOW-Modell verwendet. Das heißt, es erfolgte ein Vergleich von Simulationsergebnissen von geometrisch verschiedenen Modellen. Dies kann durchaus zu leichten Unterschieden in den Ergebnissen führen. Noch bestehende Abweichungen im Bereich bis zu ca. 0,5 Kelvin in beide Richtungen lassen sich so erklären (Hinweis: geschätzter Wert basierend auf bisherigen Erfahrungen im Umgang mit Wärmetransportmodellen). Eine nochmalige Neuerstellung eines im Umfang zum FEFLOW-Modell identischen MF6-Modells war aus Kapazitätsgründen im Projekt sowie aus Gründen der Datenverfügbarkeit nicht möglich. MF6 und FEFLOW verwenden zudem verschiedene Gleichungslöser, welche ebenfalls zu (vergleichsweise kleinen) Diskrepanzen führen können. Insgesamt besteht also noch technischer Verbesserungsbedarf, dennoch kann der Transfer in Grundzügen als erfolgreich gewertet werden.

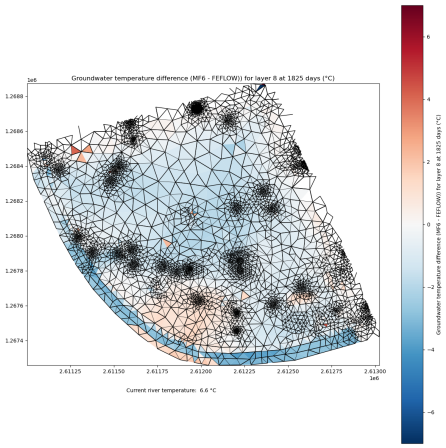

Abbildung 3.23.: Unterschiede der Grundwassertemperaturen der Rechnungen mit FEFLOW und MF6-Modell mit thermischer Cauchy-Randbedingung, Modellschicht 8, Zeitschritt Tag 1825, mit Implementierung des Flusses als thermische Cauchy-RB.

3.5. AP 3 – Erstellung einer zugehörigen Programmdokumentation und Schulungen

3.5.1. Dokumentation

Die Dokumentation führt den Nutzer mit zahlreichen Beispielen durch die Anwendungsmöglichkeiten von `pymf6`. Abbildung 3.24 zeigt einen Ausschnitt aus der Dokumentation zur interaktiven Nutzung von `pymf6`. Der Nutzer kann alle Modell-Variablen von MF6 interaktiv in einem Jupyter-Notebook analysieren. In jeder Notebook-Zelle können ein oder mehrere Modell-Zeitschritte ablaufen. Bei jedem Schritt hat der Nutzer lesenden Zugriff auf alle über das BMI exponierte Variablen. Der größte Teil ist auch modifizierbar, sodass MF6 beim nächsten Zeitschritt mit den aktualisierten Werten arbeitet. Es ist auch möglich, bei jedem Lösungszeitschritt auf Variablen lesend und schreibend zuzugreifen. Innerhalb eines Modell-Zeitschritts arbeitet MF6 Lösungszeitschritte ab, um die Lösung iterativ zu erreichen. Durch den Steuerungseingriff innerhalb der Lösungsschleife ist eine feinere Regelung von Variablen möglich.

`pymf6` fügt vielen Modell-Variablen grafische Darstellungen im HTML-Format hinzu, das Jupyter-Notebook anzeigt. Damit erhält der Nutzer oft weitere, nützliche Informationen über die jeweilige Variable, wie Dimensionen und eine Beschreibung. Letztere hat `pymf6` aus den Quelltexten von MF6 extrahiert.

Die Interaktivität und die Zusatz-Informationen sind insbesondere in der ersten Phase der Entwicklung eines Steuermoduls, das mit MF6 interagiert, äußerst nützlich. Je nach Komplexität des Modells und der Anzahl der genutzten MF6-Pakete kann ein Modell mehrere tausend über das BMI ansprechbare Variablen haben. Für eine Steuerung sind meist nur wenige davon nötig. Bei der Identifizierung der geeigneten Variablen hilft die Interaktivität immens. Es ist beispielsweise möglich, programmatisch durch die Attribute einer Variablen zu gehen und nach Kriterien zu suchen.

```
mf6.simulation.TDIS.PERLEN
```

Variable PERLEN

value:	array([1., 10., 10., 10.])
shape:	(4,)
docstring:	length of each stress period
docstring source:	src/Timing/tdis.f90 line 38

Abbildung 3.24.: Beispiel für interaktives Arbeiten mit `pymf6` aus der Dokumentation

Abbildung 3.25 zeigt einen Ausschnitt aus der Dokumentation zur Steuerung der Strömungs-Cauchy-Randbedingungen. In Abhängigkeit der Strömungsrichtung des Grundwassers vom Fluss zum Grundwasser oder umgekehrt verändert das Steuerprogramm die Durchlässigkeit der Flusssohle dynamisch. Wenn der Grundwasserstand in der Modell-Randbedingungszelle höher ist als der Flusswasserstand, nutzt das Steuermodul den vorgegebenen Referenz-Wert für die Durchlässigkeit des Flusses `condref`. Wenn der Flusswasserstand über dem Grundwasserstand liegt, reduziert das Steuermodul die Durchlässigkeit auf 10 % des Wertes. Hydraulischer Hintergrund für dieses Verhalten ist das Einspülen von feinen, an der Flusssohle abgelagerten Partikeln, der Kolmation, die die Durchlässigkeit signifikant verringert. Die Reduzierung auf 10 % ist ein sehr einfacher Algorithmus. Komplexere, nicht lineare Abminderungen oder die Nutzung einer analytischen Lösung für die zeitliche Variation von Kolmationsschichtdicken in Abhängigkeit des Wasserstands-Unterschieds sind möglich.

```
for model in loop:
    if model.state == states.iteration_start:
        if gwf.X[0, 0, 0] > h_mean:
            riv.cond = condref
        else:
            riv.cond = condref * 0.10
    elif model.state == states.timestep_end:
        flux.append(mf6.sim_values.river[0])
        chd.append(mf6.sim_values.chd_0[0])
```

The results are the same:

```
plot(flux);
```

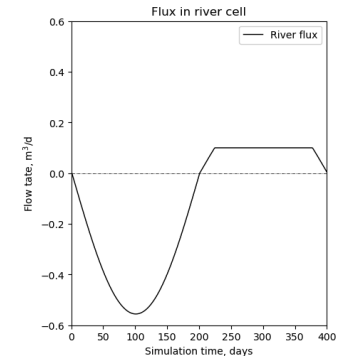

Abbildung 3.25.: Beispiel für Steuerung einer MF-6-Rechnung mit `pymf6` aus der Dokumentation, richtungsabhängige Leitfähigkeit der Flusssohle

Die Dokumentation bietet weitere Beispiele aus den Bereichen Strömungs-, Stofftransport- und Wärmetransport-Modellierung. Diese können Nutzer als Vorlagen für eigene Anwendungen nutzen. Für die Umsetzung eigener Steuermodule bietet hydrocomputing

GmbH & Co. KG Nutzern Beratungs- und Programmier-Dienstleistungen an.

3.5.2. Schulungen

Ein wichtiger Teil der Verwertungsstrategie sind Schulungen. Die Zielgruppe von Schulungen sind Grundwasser-Modellierende, die mit MF6 vertraut sind. Alle Schulungen richten sich an ein internationales Publikum und werden daher in englischer Sprache angeboten. Grundsätzlich sind Schulungen in Person und online geplant. Die Online-Schulungen werden vorzugsweise mit dem System BigBlueButton (BBB) umgesetzt (»Virtual Classroom Software | BigBlueButton — bigbluebutton.org«, n. d.). BBB ist Open Source und kommt vor allem im akademischen Bereich zum Einsatz und bietet ein „virtuelles Klassenzimmer". Für alle Schulungen kommt ein selbst gehostetes JupyterHub-System (»Project Jupyter — jupyter.org«, n. d.) zum Einsatz. [2] Abbildung 3.26 zeigt wichtige Teile des genutzten JupyterHub-Systems. Die Teilnehmenden müssen nichts installieren. Sie erhalten Login-Informationen und können damit einfach im Browser mitarbeiten. Jeder Teilnehmende bekommt eine eigene JupyterLab-Instanz (»Project Jupyter — jupyter.org«, n. d.), die aus Nutzersicht wie ein auf dem eigenen Computer installiertes System fungiert. Damit sind alle Arbeitsschritte, die auch auf einem lokalen System möglich sind, umsetzbar. JupyterHub hat sich in Schulungen vielfach bewährt. So bietet die hydrocomputing GmbH & Co. KG regelmäßig Schulungen und Workshops zu PITLAKQ an. PITLAKQ ist ein gekoppeltes Modell für die Abbildung von Wasser-Beschaffenheits-Prozessen in vom Grundwasser beeinflussten Bergbau-Restseen.

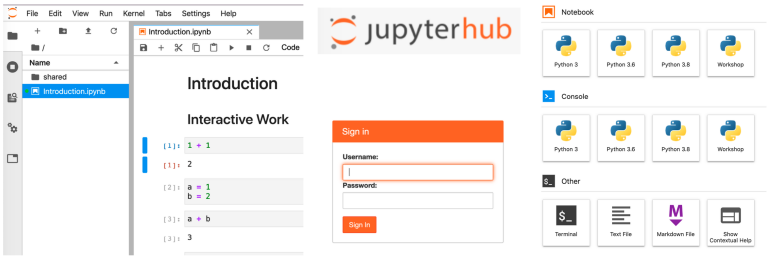

Abbildung 3.26.: Internationaler Workshop zur Nutzung von `pymf6`

Die erste Schulung wird ein Workshop am 9./10. Oktober an der TU Bergakademie Freiberg sein. Abbildung 3.27 zeigt die beschreibende Website zum Workshop. Der Workshop ist ein Gemeinschafts-Projekt der hydrocomputing GmbH & Co. KG, der TU Bergakademie Freiberg, der Deutschen Bundesstiftung Umwelt und der Fachsektion Hydrogeologie e.V. in der Deutschen Geologischen Vereinigung e.V. (FH-DGGV). Dieser Workshop findet hybrid in englischer Sprache statt. Neben der Teilnahme vor Ort ist die Online-Teilnahme möglich. Damit erweitert sich der potenzielle Teilnehmenden-Kreis international. Die Erfahrung mit PITLAKQ-Schulungen zeigt, dass durchaus Modellierende aus den Amerikas, Asien und Australien trotz zeitlicher Verschiebung der Kurszeiten bereit sind, online dabei zu sein.

[2] Das Hosting von BBB und JupyterHub übernimmt die Python Academy GmbH & Co. KG. Sie ist das Partner-Unternehmen der hydrocomputing GmbH & Co. KG.

MODFLOW 6 Lab: Customizing for Geothermal, Flow, and Transport Modeling

Empowering Modelers to Extend the Functionality of MODFLOW 6

Infos

Date	October 9 - 10, 2025
Location	• TU Bergakademie Freiberg, Germany (Library, room UBH-0208) • remote
Fees	• Full: onsite 50 €, remote 10 € • Students: onsite 25 €, remote 5 €
Format	Hand-on workshop. All participants work on a computer.
Registration	Get your ticket

Abbildung 3.27.: Internationaler Workshop zur Nutzung von `pymf6`

4. Fazit

4.1. Gesamteinschätzung

Das Projekt ist überaus erfolgreich verlaufen, die Projektziele wurden erreicht. Im Ergebnis ist mit `pymf6` ein leistungsfähiges Werkzeug entstanden, das die Modellierung geothermischer Prozesse entscheidend verbessern kann. Da ein geothermisches Modell immer auf einem Strömungsmodell aufbaut und die Stofftransport-Modellierung auf den gleichen Grundlagen wie die Wärmetransport-Modellierung basiert, ist `pymf6` in gleicher Weise für Strömungs- und Stofftransport-Modelle einsetzbar.

Die Entwicklungsstrategie hat sich in Phase 1 und Phase 2 dieses Projektes mehrfach geändert. Durch die Anpassung an neue Entwicklungen konnte die Qualität und Funktionalität von `pymf6` kontinuierlich verbessert werden. Durch konsequente Anwendung des Open-Source-Ansatzes wurden die neuesten Arbeiten des USGS aufgenommen und Code-Verbesserungen an die USGS-Entwickler zurückgegeben.

Nach umfassender Validierung konnten analytische Lösungen in die numerischen Rechnungen eingebunden werden, die sich für viele Anwendungsfälle eignen. Der Anwendungsfall Grundwasser-Wärmetransport-Modell Kleinbasel hat gezeigt, dass sich auch sehr komplexe Modelle mit dem neuen System umsetzen lassen. Die Erweiterung der Modell-Funktionalität um eine thermische Cauchy-Randbedingung hat die Modellabbildung verbessert.

Schulungen, die auf der auf der interaktiven Dokumentation aufbauen, sind geplant. Der erste Workshop-Termin steht fest. Der Fokus auf internationale Teilnehmende erweitert den potenziellen Anwenderkreis beträchtlich.

Die im Projekt entstandenen Produkte und Konzepte werden auch in Zukunft weiterentwickelt. Durch die direkte Kommunikation mit den Entwicklern des USGS ist eine gute Grundlage für eine kontinuierliche Aktualisierung gegeben.

4.2. Arbeitspakete und Meilensteinerreichung

4.2.1. AP 1.1 – Entwicklung der ueflow-Module zur numerischen Simulation von Stoff- und Wärmetransport

Das ueflow-Modul `pymf6` unterstützt jetzt den Stoff- und Wärmetransport. Grundlage sind das Groundwater Transport Model (GWT) und das Groundwater Energy Transport Model (GWE) von MF6. Das neue GWE machte den ursprünglichen Ansatz der Abbildung der Wärmetransport-Prozesse mit dem GWT mittels Parameter-Übersetzung überflüssig.

Die Programmierung des Laufzeit-Zugriffs auf MF6-Rechnungen hat große Fortschritte gemacht. Die im Laufe des Projektes veröffentlichte Bibliothek `modflowapi` (Hughes et al., 2022) bietet wesentlich bessere Interaktions-Möglichkeiten mit MF6. Diese wurde in `pymf6` integriert. Einsatz und Test von `pymf6` haben Probleme in `modflowapi` aufgezeigt. Durch intensive Kommunikation mit den Entwicklern von `modflowapi` und Beiträgen

zur Code-Basis von `modflowapi` über Issues und Pull Requests konnten diese Probleme für `pymf6` gelöst werden.

Dieser Meilenstein ist damit erreicht. Die derzeitige Funktionalität von `pymf6` übertrifft die geplanten Ziele beträchtlich. Der von Beginn an gewählte Open-Source-Ansatz hat sich bewährt. Das „Upstreaming“ von Verbesserungen in `modflowapi` hat sich bewährt, um eine solide Code-Basis zu erhalten. Damit konnten sich die Arbeiten an `pymf6` auf die projektspezifischen Merkmale konzentrieren.

4.2.2. AP 1.2 – Validierung der Simulation von Stoff- und Wärmetransport mit ueflow anhand synthetischer Testszenarien

Die Validierung von `pymf6` hat gezeigt, dass es auch für komplexe Modelle geeignet ist. Der Ansatz, die vom USGS gewarteten MF6-Examples zu nutzen, hat sich bewährt. Während der Validierung sind Probleme aufgetreten, die sich auf die Umsetzung in `modflowapi` zurückführen ließen. Alle relevanten Probleme konnten während der Validierung gelöst werden. Code-Beiträge zu `modflowapi` haben fehlerhafte Zeitsteuerungsroutinen für Simulationen mit mehreren Modellen korrigiert. Das Problem mit nicht genutzten Werten von Zeitreihen-Randbedingungen löst `pymf6` mit einem Schalter zum Überspringen des Austauschs dieser Werte in der Lösungsschleife. Für viele Anwendungen ist dieser vereinfachende Ansatz ausreichend.

Dieser Meilenstein ist damit erreicht. Die automatisierte Ausführung und Auswertung vieler Modelle hat sich bewährt. Der Stoff- und Wärmetransport ist mit `pymf6` zuverlässig abbildbar.

4.2.3. AP 2.1 – Kopplung von analytischen Randbedingungen des Stoff- und Wärmetransports mit numerischen Stoff-Wärmetransportmodulen in ueflow

Die Kopplung von `pymf6` mit TTim, das die Methode der analytischen Elemente umsetzt, deckt viele der in Phase 1 analysierten analytischen Lösungen ab. Damit ist die Funktionalität beträchtlich erweitert. Die Beispiele der detaillierten Abbildung eines Brunnens ohne Modellverfeinerung und die Modellierung mehrerer Brunnen in einer Modellzeile geben einen Eindruck der Möglichkeiten dieses Ansatzes.

Die analytische Lösung für den Wärmeübergang aus der Atmosphäre in den Untergrund wurde in ein Python-Programm umgesetzt. Dieser Programmteil ist für die Kopplung vorbereitet.

Dieser Meilenstein ist damit weitestgehend erreicht. Die Kopplung wird derzeit weitergeführt und wird im praxisnahen Anwendungsfall (AP 2.2) zum Einsatz kommen.

4.2.4. AP 2.2 – Praxisnaher Anwendungsfall zur Evaluierung

Die Nutzung des geothermischen Grundwassermodells für die Stadt Basel als Anwendungsfall hat gezeigt, dass auch sehr komplexe Modelle mit dem hier entwickelten System rechenbar sind. Die Umsetzung des FEFLOW-basierten Modells in ein MF6-Modell erfolgte mit dem, in diesem Projekt neu entwickelten, Konvertierungswerkzeug `feflowtomf6`. Der Konvertierungsprozess hat sich komplexer als erwartet herausgestellt.

Die Auskopplung eines Teilgebietes führt zur Reduzierung der Rechenzeiten. Allerdings entstehen komplexere geometrische Modellränder, die die Modellhandhabung erschweren.

Das MF6-basierte Strömungsmodell kann die Ergebnisse des FEFLOW-Modells gut reproduzieren. Das Wärmetransportmodell kann die mit dem FEFLOW-Modell berechnete Temperaturverteilung aber noch nicht ausreichend gut berechnen. Die Implementierung einer thermischen Cauchy-Randbedingung mit `pymf6` hat die Modellabbildung verbessert. Die Unterschiede zum FEFLOW-Modell sind aber immer noch zu groß. Derzeit läuft die Kopplung mit der in AP 2.1 programmierten analytischen Lösung für den Wärmeübergang aus der Atmosphäre und den Gebäuden. Damit sollten sich die Modellergebnisse verbessern.

Dieser Meilenstein ist damit weitestgehend erreicht. Die Anwendung von `pymf6` auf ein sehr komplexes Praxis-Modell hat bereits gute Ergebnisse erbracht. Die Erweiterungsmöglichkeiten der Modellfunktionalität helfen, die Modell-Abbildung zu verbessern. Aufgrund der Komplexität des Modells sind weitere Arbeiten nötig. Der grundsätzliche Ansatz hat sich wegen seiner Flexibilität, die eine Anpassung an neu auftretende Probleme erlaubt, bewährt.

4.2.5. AP 3 – Erstellung einer zugehörigen Programmdokumentation und Schulungen

Eine umfangreiche Programmdokumentation mit zahlreichen Beispielen ermöglicht es Anwendern, schnell eigene Modelle mit `pymf6` zu erstellen. Die Interaktion mit laufenden Modellen zu jedem Zeitschritt mit einer Kombination aus Python-Code und grafischen Elementen ermöglicht ein schnelles Verständnis der Funktionsweise von `pymf6`.

Alle Schulungen sind grundsätzlich sowohl für die Vor-Ort- als auch die Online-Durchführung ausgelegt. Die Kurs-Sprache ist vorwiegend Englisch, um ein internationales Publikum anzusprechen. Durch die Nutzung eines Online-Programmiersystems sind nur ein Internet-Browser und ein Internet-Zugang nötig. Die lokale Installation von Software ist nicht nötig. Der erste hybride Workshop in Zusammenarbeit mit der DBU, TUBAF und FH-DGGV wird im Oktober 2025 stattfinden.

Dieser Meilenstein ist damit erreicht. Die Dokumentation wird kontinuierlich erweitert. Weitere Schulungen und Workshops sind in Planung.

Literatur

Anderson, M. P. (2005). Heat as a ground water tracer. *Ground Water*, *43*(6), 951–968 (siehe S. 13, 15).

Anibas, C., Buis, K., Verhoeven, R., Meire, P., & Batelaan, O. (2011). A simple thermal mapping method for seasonal spatial patterns of groundwater–surface water interaction. *J. Hydrol. (Amst.)*, *397*(1-2), 93–104 (siehe S. 13, 15).

Bakker, M. (2006). An analytic element approach for modeling polygonal inhomogeneities in multi-aquifer systems. *Advances in Water Resources*, *29*(10), 1546–1555 (siehe S. 12).

Bakker, M. (2013a). Semi-analytic modeling of transient multi-layer flow with TTim. *Journal of Hydrology*, *21*(4), 935–943 (siehe S. 12).

Bakker, M., Post, V., Langevin, C. D., Hughes, J. D., W., J. T., S., J., J., & Fienen, M. N. (2016). Scripting MODFLOW model development using Python and FloPy. *Groundwater*. https://doi.org/doi:10.1111/gwat.12413 (siehe S. 18).

Bakker, M., & Strack, O. (2003). Analytic Elements for Multiaquifer Flow. *Journal of Hydrology*, *271*(1-4), 119–129 (siehe S. 12).

Bakker, M. (2013b). *Analytic modeling of transient multilayer flow*. (Siehe S. 12).

Banks, D. (2009). Thermogeological assessment of open-loop well-doublet schemes: a review and synthesis of analytical approaches. *Hydrogeol. J.*, *17*(5), 1149–1155 (siehe S. 13, 15).

Bedekar, V., Morway, E. D., Langevin, C. D., & Tonkin, M. J. (2016). MT3D-USGS version 1: A U.S. Geological Survey release of MT3DMS updated with new and expanded transport capabilities for use with MODFLOW. *Techniques and Methods*, (6-A53). https://doi.org/10.3133/tm6A53 (siehe S. 3).

Binder, M., Händel, F., Engelmann, C., Steiner, B., Hasler, A., García Gil, A., & Epting, J. (2025). The Subsurface Urban Heat Island of Basel-City – More than a decade of spatiotemporal high-resolution monitoring and modelling. *Philosophical Transactions of the Royal Society A*, (accepted). https://doi.org/10.1098/rsta.2020.0568 (siehe S. 25).

Bundschuh, J. (1993). Modelling heat transport from the earth's surface through aquifers to springs: theoretical examples and case studies (siehe S. 13, 15).

Claesson, J., & Javed, S. (2011). An Analytical Method to Calculate Borehole Fluid Temperatures for Time-scales from Minutes to Decades (ML-11-C034). *ASHRAE Transactions*, *117*, 279–288 (siehe S. 13, 15).

Constantz, J. (2008). Heat as a tracer to determine streambed water exchanges. *Water Resour. Res.*, *44*(4) (siehe S. 13, 15).

CSDMS. (2023a). The Basic Model Interface (BMI). Verfügbar 9. Mai 2023 unter https://bmi-spec.readthedocs.io/en/latest (siehe S. 6).

CSDMS. (2023b). bmi-python. Verfügbar 9. Mai 2023 unter https://github.com/csdms/bmi-python/blob/master/bmipy/bmi.py (siehe S. 7).

CSDMS. (2023c). CSDMS. Verfügbar 9. Mai 2023 unter https://csdms.colorado.edu/wiki/About_CSDMS (siehe S. 6).

Deltares. (2023). xmipy. Verfügbar 9. Mai 2023 unter https://github.com/Deltares/xmipy (siehe S. 7).

Diao, N., Li, Q., & Fang, Z. (2004). Heat transfer in ground heat exchangers with groundwater advection. *Int. J. Therm. Sci.*, *43*(12), 1203–1211 (siehe S. 13, 15).

Diersch, H.-J. G. (2014). *FEFLOW: Finite Element Modeling of Flow, Mass and Heat Transport in Porous and Fractured Media*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-38739-5 (siehe S. 25).

Dresmann, H., Huggenberger, P., Epting, J., Wiesmeier, S., & Scheidler, S. (2013). A 3D spatial planning tool-application examples from the Basel region. (Siehe S. 25).

Epting, J., Händel, F., & Huggenberger, P. (2013). Thermal management of an unconsolidated shallow urban groundwater body. *Hydrology and Earth System Sciences*, *17*(5), 1851–1869. https://doi.org/10.5194/hess-17-1851-2013 (siehe S. 25).

Foundation, P. S. (2023). ctypes. Verfügbar 9. Mai 2023 unter https://docs.python.org/3.11/library/ctypes.html (siehe S. 7).

Goto, S., Kim, H. C., Uchida, Y., & Okubo, Y. (2005). Reconstruction of the ground surface temperature history from the borehole temperature data in the southeastern part of the Republic of Korea. *J. Geophys. Eng.*, *2*(4), 312–319 (siehe S. 13, 15).

Gringarten, A. C., Witherspoon, P. C., & Ohnsishi, Y. (1975). Theory of Heat Extraction From Fractured Hot Dry Rock. *Journal of Geophysical Research*, *80*(8), 1120–1124 (siehe S. 13, 15).

Hähnlein, S., Molina-Giraldo, N., Blum, P., Bayer, P., & Grathwohl, P. (2010a). Ausbreitung von Kältefahnen im Grundwasser bei Erdwärmesonden. *Grundwasser*, *15*(2), 123–133. https://doi.org/10.1007/s00767-009-0125-x (siehe S. 15).

Hähnlein, S., Molina-Giraldo, N., Blum, P., Bayer, P., & Grathwohl, P. (2010b). Ausbreitung von Kältefahnen im Grundwasser bei Erdwärmesonden. *Grundwasser (Berl.)*, *15*(2), 123–133 (siehe S. 13).

Händel, F., Liedl, R., Fank, J., & Rock, G. (2013). Regional modeling of geothermal energy systems in shallow aquifers: the Leibnitzer Feld case study (Austria). *Environ. Earth Sci.*, *70*(8), 3433–3446 (siehe S. 13, 15, 22–24, 55).

Hatch, C. E., Fisher, A. T., Revenaugh, J. S., Constantz, J., & Ruehl, C. (2006). Quantifying surface water–groundwater interactions using time series analysis of streambed thermal records: Method development. *Water Resour. Res.*, *42*(10) (siehe S. 13, 15).

Hecht-Méndez, J., Molina-Giraldo, N., Blum, P., & Bayer, P. (2010). Evaluating MT3DMS for Heat Transport Simulation of Closed Geothermal Systems. *Groundwater*, *48*(5), 741–756. https://doi.org/10.1111/j.1745-6584.2010.00678.x (siehe S. 5, 13, 15).

Huggenberger, P., & Epting, J. (2011). *Urban geology: Process-oriented concepts for adaptive and integrated resource management*. Springer. https://doi.org/10.1007/978-3-0348-0185-0 (siehe S. 25).

Hughes, J. D., Russcher, M. J., Langevin, C. D., Morway, E. D., & McDonald, R. R. (2022). The MODFLOW Application Programming Interface for simulationcontrol and software interoperability. *Environmental Modelling & Software*, *148*(105257). https://doi.org/doi:10.1016/j.envsoft.2021.105257. (siehe S. 7, 44).

Keery, J., Binley, A., Crook, N., & Smith, J. W. N. (2007). Temporal and spatial variability of groundwater–surface water fluxes: Development and application of an analytical method using temperature time series. *J. Hydrol. (Amst.)*, *336*(1-2), 1–16 (siehe S. 13, 15).

Köhler, M., Händel, F., Epting, J., Binder, M., Mueller, M. H., Huggenberger, P., & Liedl, R. (2015). Numerical Evaluation and Optimization of Depth-oriented Temperature Measurement for the Investigation of Thermal Influences on Groundwater Resources [European Geosciences Union General Assembly 2015 – Division Energy, Resources and Environment, EGU 2015]. *Energy Procedia*, *76*, 371–380. https://doi.org/https://doi.org/10.1016/j.egypro.2015.07.844 (siehe S. vi, 2).

Larsen, J. (2025). Merge of Pull Request: Fix number of stress period end states for multi-model simulations. Verfügbar 8. Mai 2025 unter https://github.com/MODFLOW-ORG/modflowapi/commit/5679349789b2ae7dfb70ec10c6c0caa2399c5d80 (siehe S. 8).

Luhmann, A. J., Covington, M. D., Myre, J. M., Perne, M., Jones, S. W., Alexander, E. C., Jr, & Saar, M. O. (2015). Thermal damping and retardation in karst conduits. *Hydrol. Earth Syst. Sci.*, *19*(1), 137–157 (siehe S. 13, 15).

Ma, R., & Zheng, C. (2010). Effects of density and viscosity in modeling heat as a groundwater tracer. *Groundwater*, *48*(3), 380–389. https://doi.org/doi:10.1111/j.1745-6584.2009.00660.x (siehe S. 5).

Molina-Giraldo, N., Bayer, P., & Blum, P. (2011). Evaluating the influence of thermal dispersion on temperature plumes from geothermal systems using analytical solutions. *Int. J. Therm. Sci.*, *50*(7), 1223–1231 (siehe S. 13, 15).

Mueller, M. H., Huggenberger, P., & Epting, J. (2018). Combining monitoring and modelling tools as a basis for city-scale concepts for a sustainable thermal management of urban groundwater resources. *The Science of the total environment*, *627*, 1121–1136. https://doi.org/10.1016/j.scitotenv.2018.01.250 (siehe S. 25).

Müller, M. (2025). Pull Request: Fix number of stress period end states for multi-model simulations. Verfügbar 8. Mai 2025 unter https://github.com/MODFLOW-ORG/modflowapi/pull/73 (siehe S. 8).

Müller, M., Händel, F., Engelmann, C., Binder, M., Huggenberger, P., Börke, P., & Epting, J. (2020, September). *Entwicklung und Validierung eines modularen Simulationswerkzeuges für hydrogeologische und geothermische Fragestellungen mit einer interaktiven Schnittstelle zur direkten Implementierung dynamischer und rückkoppelnder Randbedingungen ModSimple – Phase 1 : Abschlussbericht über ein Entwicklungsprojekt, gefördert unter dem Az.: DBU-AZ 32961/01 von der Deutschen Bundesstiftung Umwelt* (Abschlussbericht) (Förderkennzeichen: DBU3296101, PPN: 1758326816). hydrocomputing GmbH & Co. KG. https://lbsvz4.gbv.de/DB=3/SET=3/TTL=231/CMD?ACT=SRCHA&IKT=1016&SRT=YOP&TRM=Modsimple (siehe S. 3, 6).

Müller, M., Pedrosa, L., Engelmann, C., & Binder, M. (2023). Towards Complex Geothermal Simulations – Controlling MODFLOW 6 Groundwater Flow Models at Runtime to Enable for Dynamic Boundary Conditions. *Simulation Umwelt- und Geowissenschaften – Workshop Bayreuth 2023* (siehe S. 62).

Pannike, S., Kölling, M., Schulz, H. D., Panteleit, B., Reichling, J., & Scheps, V. (2006). Auswirkung hydrogeologischer Kenngrößen auf die Kältefahnen von Erdwärmesondenanlagen in Lockersedimenten. *Grundwasser (Berl.)*, *11*(1), 6–18 (siehe S. 13, 15).

Project Jupyter — jupyter.org [Accessed 2025-06-23]. (n. d.). (Siehe S. 42).

Provost, A. M., Langevin, C. D., & Hughes, J. D. (2017). Documentation for the "XT3D" option in the Node Property Flow (NPF) Package of MODFLOW 6. https://doi.org/10.3133/TM6A56 (siehe S. 5).

Rau, G. C., Andersen, M. S., McCallum, A. M., & Acworth, R. I. (2010). Analytical methods that use natural heat as a tracer to quantify surface water–groundwater exchange, evaluated using field temperature records. *Hydrogeol. J.*, *18*(5), 1093–1110 (siehe S. 13, 15).

Shook, G. (2001). Predicting thermal breakthrough in heterogeneous media from tracer tests. *Geothermics*, *30*, 573–589. https://doi.org/10.1016/S0375-6505(01)00015-3 (siehe S. 13, 15).

USGS. (2020). MODFLOW and Related Programs. https://www.usgs.gov/mission-areas/water-resources/science/modflow-and-related-programs?qt-science_center_objects=0#qt-science_center_objects (siehe S. 3).

USGS. (2025). MODFLOW 6 Examples. Verfügbar 21. Mai 2025 unter https://github.com/MODFLOW-ORG/modflow6-examples (siehe S. 9).

Virtual Classroom Software | BigBlueButton — bigbluebutton.org [Accessed 2025-06-23]. (n. d.). (Siehe S. 42).

Vogt, T., Schirmer, M., & Cirpka, O. A. (2012). Investigating riparian groundwater flow close to a losing river using diurnal temperature oscillations at high vertical resolution. *Hydrol. Earth Syst. Sci.*, *16*(2), 473–487 (siehe S. 13, 15).

Wikipedia. (2025). Analytic element method — Wikipedia, The Free Encyclopedia [Accessed 2025-05-14]. (Siehe S. 12, 13).

Anhänge

A. Anhang: BMI

Tabelle 1.: BMI methods (selection)

Function	Description
`initialize`	Perform startup tasks for the model.
`update`	Advance model state by one time step.
`update_until`	Advance model state until the given time.
`finalize`	Perform tear-down tasks for the model.
`get_component_name`	Name of the model.
`get_input_item_count`	Count of a model's input variables.
`get_output_item_count`	Count of a model's output variables.
`get_input_var_names`	List of a model's input variables.
`get_output_var_names`	List of a model's output variables.
`get_var_grid`	Get the grid identifier for a variable.
`get_var_type`	Get the data type of a variable.
`get_var_units`	Get the units of a variable.
`get_var_itemsize`	Get the size (in bytes) of one element of a variable.
`get_var_nbytes`	Get the total size (in bytes) of a variable.

B. Anhang: Implementierung der analytischen Lösung für den Wärmeübergang aus der Atmosphäre

```python
"""Multi-layer unsaturated zone thermal model.

Reference:
Haendel et al. (2013)
Regional scale heat transport in unsaturated zones
"""

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

import numpy as np


def multi_layer_model(
    T,
    n_time,
    n_layer,
    d,
    q,
    Lamda,
    c,
    r,
    rw,
    cw,
    dt,
):
    """Solves 1D heat transport through multiple soil layers.

    Parameters:
    -----------
    T : ndarray
        Initial temperature matrix (n_time x n_layer) [K]
    n_time : int
        Number of time steps
    n_layer : int
        Number of soil layers
    d : ndarray
        Layer thicknesses [m]
    q : ndarray
        Percolation water flux [m/s]
    Lamda : float
        Thermal conductivity [W/m K]
    c : float
        Volumetric heat capacity [J/m^3 K]
    r : float
        Bulk density [kg/m^3]
    rw : float
        Water density [kg/m^3]
    cw : float
        Water heat capacity [J/kg K]
    dt : float
        Time step [s]
    max_iter : int
        Maximum iterations for convergence
    max_error : float
        Convergence threshold [K]

    Returns:
    --------
    T : ndarray
        Temperature distribution [K]

    Assumptions:
    Transport parameters of the unsaturated zone are vertically and
```

```
    horizontally homogeneously distributed there is no significant
    horizontal transport in the unsaturated zone soil surface
    temperature is uniformly distributed for the complete model area
    """
    # Initialize arrays
    Phi = np.zeros((n_layer, n_layer))
    for i in range(1, n_time):
        for j in range(1, n_layer - 1):
            # Calculate conductance between layers
            Phi[j, j + 1] = Lamda / (d[j] / 2 + d[j + 1] / 2)
            Phi[j - 1, j] = Lamda / (d[j - 1] / 2 + d[j] / 2)

            # Heat flux components
            Pc_01 = q[i] * rw * cw * (T[i - 1, j - 1] - T[i - 1, j])
            PD_01 = Phi[j - 1, j] * (T[i - 1, j - 1] - T[i - 1, j])
            PD_12 = Phi[j, j + 1] * (T[i - 1, j] - T[i - 1, j + 1])

            # Temperature update
            dT_ij = (Pc_01 + PD_01 - PD_12) * dt / (c * r * d[j])
            T[i, j] = T[i - 1, j] + dT_ij
    return T


def plot_temperature_distribution(time_steps, depths, T):
    """Visualizes temperature evolution through soil profile.

    Parameters:
    -----------
    time_steps : ndarray
        Time points [s]
    depths : ndarray
        Layer depths [m]
    T : ndarray
        Temperature matrix [C]
    """
    plt.figure(figsize=(10, 6))

    # Convert time to days for readability
    time_days = time_steps / 86400
    depth_cumulative = np.cumsum(depths)
    # Convert temperature for celcius
    T_celsius = T - 273.15  # Convert K to C

    # Create contour plot
    X, Y = np.meshgrid(depth_cumulative, time_days)
    cont = plt.contourf(Y, X, T_celsius, levels=20, cmap=cm.viridis)

    # Axis formatting
    plt.gca().invert_yaxis()  # Surface at top (0m), depth increasing downward

    plt.xlabel('With (m)', fontsize=12)
    plt.ylabel('Depth (m)', fontsize=12)
    cbar = plt.colorbar(cont)
    cbar.set_label('Temperature (C)', fontsize=12)
    plt.title('Temperature Evolution with Depth', fontsize=14)
    plt.tight_layout()
    plt.savefig('temperature_depth_haendel_eq_plot.png')
    plt.show()


if __name__ == '__main__':
    # Parameters
    n_time = 100
    n_layer = 10
    d = np.array([0.5] * n_layer)  # Uniform 0.5m layers
    q = np.zeros(n_time)
    q[:] = 1e-6  # Small initial percolation

    # Material properties (typical sandy soil)
    Lamda = 2.13  # Thermal conductivity [W/m K]
```

```
    c = 2.0e6  # Volumetric heat capacity [J/m^3 K]
    r = 2.5e3  # Bulk density [kg/m^3]
    rw = 1000.0  # Water density [kg/m^3]
    cw = 4.18e6  # Water heat capacity [J/m^3 K]
    dt = 86400  # Daily time steps [s]

    # Initial conditions
    T = np.ones((n_time, n_layer)) * 283.15  # 10 C uniform
    T[:, 0] = 293.15  # Surface at 20 C
    T[:, -1] = 283.15  # Deep layer at 10 C

    # Run model
    T_result, fluxes = multi_layer_model(
        T, n_time, n_layer, d, q, Lamda, c, r, rw, cw, dt
    )

    # Visualization
    time_axis = np.arange(n_time) * dt
    plot_temperature_distribution(time_axis, d, T_result)
```

Listing 1: Implementierung der analytischen Lösung für den Wärmeübergang aus der Atmosphäre nach (Händel et al., 2013)

C. Anhang: MF6-Modell Kleinbasel

C.1. MF6-Pakete für das Strömungsmodell

Tabelle 2.: MF6-Pakete für das Strömungsmodell

Akronym	Package	Nutzung
CHD	Constant-Head	räumliche Zuordnung der Grundwasserstands-Zeitreihen am Modellrand
DIS	Structured Discretization	räumliche Diskretisierung in Dreieckselemente, gleich für alle Modell-Schichten
IMS	Specified Complexity	Modell-Komplexität „complex“ (lineare und nicht-lineare Variablen für die Gleichungslösung)
NAM	Simulation Name File	zentrale Steuerdatei für die Simulation
NPF	Node Property Flow	räumliche Verteilung de hydraulischen Leitfähigkeit des Grundwassers
OC	Output Control	Spezifikation der Speicherzeitpunkte von Wasserstand, Temperatur und Volumenbilanzen
RCH	Recharge	räumliche Zuordnung der Grundwasser-Neubildungs-Zeitreihen am oberen Modellrand
RIV	River	räumliche Zuordnung der Wasserstands-Zeitreihen des Rheins sowie Höhe und Durchlässigkeit der Flusssohle
STO	Storage	konstante Werte für den spezifischen Speicherkoeffizienten und die effektive Porosität
TDIS	Temporal Discretization	eine Stress-Periode, zeitliche Variabilität durch Zeitreihen von Randbedingungswerten
WEL	Well	räumliche Zuordnung der Entnahme-Raten-Zeitreihen der Brunnen (negative Werte bedeuten Entnahme, positive Werte Infiltration)

C.2. Vergleich der Grundwasserstände

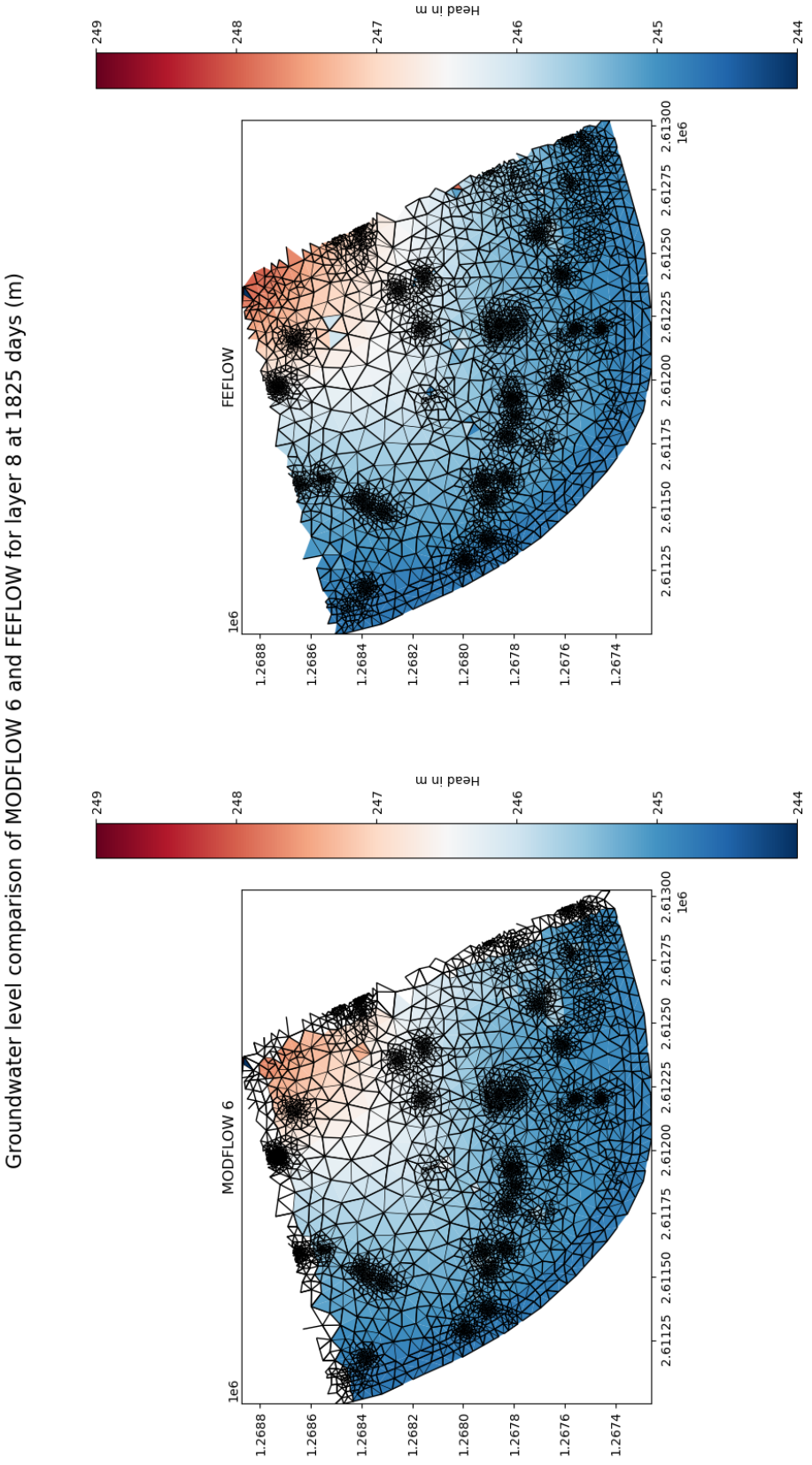

Abbildung 1.: Großdarstellung: Vergleich der Grundwasserstände der Rechnungen mit FEFLOW und MF6-Modell, Modellschicht 8, Zeitschritt Tag 1825

C.3. Cauchy-RB für den Wärmetransport mit pymf6

```python
import flopy
import numpy as np


from feflowtomf6.readers.read_river import read_river_heat_transfer
from feflowtomf6.utils.save import DB
from pymf6.api import create_mutable_bc


class RiverController:
    """River controller for temperature transfer with CAUCHY BC."""

    def __init__(self, config, mf6, bc_name='riv_0'):
        self.config = config
        self.mf6 = mf6  # instance of pymf6.mf6.MF6
        self.bc_name = bc_name
        if config.options.river_heat_transfer_area:
            self.muliply_area = True
        else:
            self.muliply_area = False
        print(f'{self.muliply_area=}')
        self.gwf = mf6.models['gwf6']
        self.gwe = mf6.models['gwe6']
        self.riv = create_mutable_bc(self.gwf.get_package(self.bc_name))
        self.riv_layers, self.river_cell_ids = self._get_river_indices()

        self._heat_transfer_in = None
        self._heat_transfer_out = None
        self._river_flux_coords_mask = None
        db = DB(config.files.db_file)
        if self.muliply_area:
            cell2d_df = db.get_data(key='cell2d_df', table_name='mfcoords')
            self.areas = cell2d_df['area'].loc[self.river_cell_ids].values
        self._river_temperature = {
            float(key): value
            for key, value in db.get_data(
                key='riv_temp',
                table_name='riv',
            ).items()
        }

    def _get_river_temperature_variabel_name(self, bc_name):
        """Find variable name."""
        model_name = f'gwe_{self.config.names.model_name}'
        riv_index = self._find_ssm_index(
            sim_path=self.config.paths.model_path,
            bc_name=bc_name,
            model_name=model_name,
        )
        return f'{model_name}/spc-{riv_index + 1}/dblvec'.upper()

    @staticmethod
    def _find_ssm_index(sim_path, bc_name, model_name):
        sim = flopy.mf6.MFSimulation.load(
            sim_ws=sim_path, lazy_io=True, load_only=['ssm'], verbosity_level=0
        )
        gwe_input = sim.get_model(model_name)
        bc_names = gwe_input.ssm.fileinput.array['pname']
        return list(bc_names).index(bc_name)

    @property
    def river_temperature(self):
        """Current river temperature."""
        current_time = self.mf6.get_current_time()
        return self._river_temperature[current_time]

    def _get_river_indices(self):
```

```
        riv_package = self.gwf.get_package(self.bc_name)
        riv_indices = riv_package.get_advanced_var('cellid') - 1
        riv_layers = riv_indices[:, 0]
        riv_ids = riv_indices[:, 1]
        return riv_layers, riv_ids

    def _set_heat_transfers(self):
        transfer_rates = read_river_heat_transfer(self.config)
        self._heat_transfer_in = transfer_rates['transfer_in'][
            self.river_cell_ids
        ].values
        self._heat_transfer_out = transfer_rates['transfer_out'][
            self.river_cell_ids
        ].values

    @property
    def heat_transfer_in(self):
        """Heat transfer direction "in"."""
        if self._heat_transfer_in is None:
            self._set_heat_transfers()
        return self._heat_transfer_in

    @property
    def heat_transfer_out(self):
        """Heat transfer direction "out"."""
        if self._heat_transfer_out:
            self._set_heat_transfers()
        return self._heat_transfer_out

    @property
    def river_coords_mask(self):
        """River cell coords in ESL."""
        if self._river_flux_coords_mask is None:
            arr = self.gwe.esl.get_advanced_var('BOUNDNAME_CST')
            self._river_flux_coords_mask = arr == self.bc_name.upper()
        return self._river_flux_coords_mask

    @property
    def current_heat_transfer(self):
        """Current in/out transfer rates."""
        gw_levels = self.gwf.X[self.riv_layers, self.river_cell_ids]
        stages = self.riv.stage
        diff = stages - gw_levels
        sign = np.sign(diff)
        in_mask = sign == 1
        out_mask = sign == -1
        heat_transfer = np.zeros_like(self.heat_transfer_in)
        heat_transfer[in_mask] = self.heat_transfer_in[in_mask]
        heat_transfer[out_mask] = self.heat_transfer_in[out_mask]
        return heat_transfer

    def set_heat_flux(self, rate=None, rate_factor=1):
        """
        Set new heat flux value for river cells.

        rate = phi x (T_riv - T_gw)
        rate can be supplied as constant for testing purposes.
        """
        if rate is None:
            t_gw = self.gwe.X[self.riv_layers, self.river_cell_ids]
            rate = (
                self.current_heat_transfer
                * (self.river_temperature - t_gw)
            )
            if self.muliply_area:
                rate *= self.areas
        values = self.gwe.esl.get_advanced_var('BOUND')
        values[self.river_coords_mask, 0] = rate * rate_factor
        self.gwe.esl.set_advanced_var('BOUND', values)
```

Listing 2: Implementierung der Cauchy-RB für den Wärmetransport mit pymf6

```python
"""Helpers to run scenarios.."""

from datetime import timedelta
from pathlib import Path
from timeit import default_timer


from modflowapi import Callbacks as states

from feflowtomf6.config import Config
from feflowtomf6 import main

from feflowtomf6.controllers.river_temperature_conductance import (
    RiverController,
)

from pymf6.mf6 import MF6

BASE_CONFIG_FILENAME = Path('configs/base_config.ini')


class TimeDelta:
    """Timedelta that displays without seconds decimals."""

    def __init__(self, seconds):
        self._dt = timedelta(seconds=seconds)

    def __str__(self):
        return f'{str(self._dt).rsplit('.', 1)[0]}'


def _make_config_file_name(model_name):
    if isinstance(model_name, Path):
        return model_name
    if model_name.endswith('.ini'):
        return Path(model_name)
    return Path(f'configs/config_{model_name}.ini')


def make_model_input(model_name, base_config_file_name=BASE_CONFIG_FILENAME):
    """Create model input."""
    config_file_name = _make_config_file_name(model_name)
    return main.main(
        config_file_name=config_file_name,
        base_config_file_name=base_config_file_name,
        use_cached=True,
        verbose=False,
    )


def run_controlled(
    model_name,
    base_config_file_name=BASE_CONFIG_FILENAME,
    control_transfer=True,
    constant_rate=None,
    rate_factor=1,
):
    """Run MF6 controlled."""
    config_file_name = _make_config_file_name(model_name)
    config = Config(
        config_file_name, base_config_file_name=base_config_file_name
    )
    start = default_timer()
    mf6 = MF6(sim_path=config.paths.model_path, do_solution_loop=False)
    loop = mf6.model_loop()
```

```
66      riv_controller = RiverController(config, mf6)
67
68      for model in loop:
69          if model.state == states.timestep_start:
70              if control_transfer:
71                  riv_controller.set_heat_flux(
72                      constant_rate, rate_factor=rate_factor
73                  )
74              duration = default_timer() - start
75              current_time = mf6.get_current_time()
76              print(
77                  current_time,
78                  TimeDelta(seconds=duration),
79                  TimeDelta(seconds=duration / current_time * 1825),
80                  end='\r',
81              )
82
83      duration = default_timer() - start
84      print('run time:', duration)
85      print(TimeDelta(seconds=duration))
```

Listing 3: Nutzung der Cauchy-RB für den Wärmetransport mit `pymf6`

D. Anhang: Veröffentlichungen

D.1. Paper

Die Veröffentlichung (Müller et al., 2023) beschreibt den Stand der programmtechnischen Umsetzung und erster Anwendungen.

Titel:	Towards Complex Geothermal Simulations – Controlling MODFLOW 6 Groundwater Flow Models at Runtime to Enable for Dynamic Boundary Conditions
Autoren:	Mike Müller; Lúcia Pedrosa; Christian Engelmann; Martin Binder
Buchtitel:	Simulation Umwelt- und Geowissenschaften – Workshop Bayreuth 2023
Verlag:	Shaker Verlag Aachen
ISBN:	978-3-8440-9250-9

Towards Complex Geothermal Simulations - Controlling MODFLOW 6 Groundwater Flow Models at Runtime to Enable for Dynamic Boundary Conditions

Mike Müller[1], Lúcia Pedrosa[2], Christian Engelmann[2], Martin Binder[2,3]

[1] hydrocomputing GmbH & Co. KG
[2] TU Bergakademie Freiberg
Faculty of Geosciences, Geoengineering and Mining (Faculty 3)
Institute for Geology, Chair of Hydrogeology and Hydrochemistry
[3] University of Basel
Department of Environmental Sciences, Hydrogeology
Applied and Environmental Geology

Abstract

Boundary conditions (BCs) need to be specified as input parameters for numerical models. Values for BCs are often not fully defined before model run, but rather depend on models results. While identifying adequate BCs with dependence on model results can be achieved through multiple forward runs of a model, i.e., with a step-by-step adjustment of BCs, the number of necessary iterations can be high. Especially for complex problems, the computational and time-wise efforts for finding appropriate values for BCs can exceed practical limits. The library `pymf6` presented here is based on a different approach. It enables dynamic boundary conditions for groundwater models that are built with MODFLOW 6 (MF6). Specifically, it allows for a direct access to all MF6 variables at each model time step. This enables for adaptively modifying values of BCs based on calculated results from previous time steps. This study uses a head-controlled groundwater abstraction well as an example and demonstrates how a groundwater flow model can be dynamically controlled by a user-written program. In this example, the groundwater level in the model grid cell of the pumping well is kept in a given range by dynamically adjusting the withdrawal rate at model run time. Subsequent steps of `pymf6` development will include the support of solute and heat transport.

1 Introduction

The use of shallow geothermal energy for heating and cooling purposes is widely spread. The proceedings of the world Geothermal Congress 2020 based on data of 88 countries about the direct use of geothermal energy point to an average annual growing percentage of 11.5 [1]. Moreover, as countries are seeking to mitigate climate change by investing in renewable energy sources, its usage will continue to increase in Europe [2], US [3], Asia [4], Africa [5] and Australia [6]. In order to accomplish the growing demand from governments to invest in such systems, it is necessary to make sure that proper tools are available for their sustainable design and correct implementation. Therefore, modeling open source tools must be developed to help scientists and engineers accurately sustain decision making processes.

D.2. Vorträge

Veranstaltung:	29. FH-DGGV-Tagung – Unsere wichtigste Georessource Grundwasser Analysieren – Prognostizieren – Gestalten
Titel:	Customizing Groundwater Models with Dynamic Boundary Conditions
Ort:	Aachen, Deutschland
Datum:	22. März 2024
Autoren:	Mike Müller; Lúcia Pedrosa; Christian Engelmann; Martin Binder

Auszug aus dem Tagungsband (nächste Seite):

patentierte Schadstoffdatenbank zur systematischen Erfassung und Verwaltung von Gebäudeschadstoffen entwickelt.

Ein weiterer wichtiger Part ist die Installation von Schnittstellen zwischen einzelnen Prozessen und zu externen Dienstleistern. Die Herausforderungen bei der täglichen Arbeit in einer organisch gewachsenen Firma, mit verschiedenen Fachbereichen, unterschiedlichen Auftraggebern und diverser Messtechnik sind vielfältig. Es ist es nicht immer möglich, Programme zu vereinheitlichen und vorzugeben, was die Anzahl an erforderlichen Schnittstellen anwachsen lässt.

Neben der Schaffung der technischen Voraussetzungen sind Motivation der Mitarbeiter sowie Schulung und Wissenstransfer wichtige Bausteine in der Umsetzung.

Die Digitale Transformation ist ein kontinuierlicher Prozess, Systeme müssen so gestaltet sein, dass wandelnde Anforderungen und Entwicklungen einbezogen werden können.

Vortrag 21.8 (ID 304)

Customizing Groundwater Models with Dynamic Boundary Conditions

Mike Müller[1], Lúcia Pedrosa[2], Christian Engelmann[2], Martin Binder[2,3]

[1]: hydrocomputing GmbH & Co. KG, 04158 Leipzig, Deutschland

[2]: Technische Universität Bergakademie Freiberg, Hydrogeologie und Hydrochemie, 09599 Freiberg (Sachsen), Deutschland

[3]: Universität Basel, Hydrogeologie / Angewandte und Umweltgeologie, 4056 Basel, Schweiz

Kontakt: mmueller@hydrocomputing.com

Numerical groundwater models are established and robust tools for the prediction of flow and transport processes in the subsurface. The typical workflow consists of preparing input data, running a simulation program such as MODFLOW, and eventually evaluating model results. An important part of this entire procedure are boundary conditions (BCs). For complex scenarios, these BCs may need to be adaptively adjusted, e.g. based on previous model results, to better reflect real-world conditions. Given complex mutual dependencies between BCs, this workaround may require multiple iterative model runs and can lead to unsolvable optimization strategies.

A solution to this problem can be found in the definition of dynamic BCs, i.e., BCs that can change their value based on pre-defined rules and depending on the values of other BCs and / or process variables. For example: An injection well can only infiltrate the amount of water which was previously pumped from an hydraulically connected extraction well. At the same time, the groundwater level in the extraction well should not fall below a certain level to protect the well and the installed pump from falling dry. This scenario,

although still comparably simple, already requires both an adaptive adjustment of the pumping rate of the extraction well (based on the process variable of hydraulic head) and of injection well's rate (based on the BC value of the coupled extraction well). This scenario can become even more complex if multiple extraction and / or injection wells are involved. From the technical point of view, dynamic behaviors such as the described one could be achieved by modifying the source code of the numerical model. While possible, this would require a considerable effort and understanding of the inner workings of the code (in case of MODFLOW: Fortran).

The latest version of MODFLOW 6 offers a novel approach by exposing a programming interface which can be externally accessed during runtime via scripting languages, e.g., Python. In our contribution, we introduce pymf6, a new open-source Python package that allows a simplified usage of the aforementioned programming interface. Using pymf6, a model user with basic Python knowledge can write a program in Python that can achieve a highly dynamic model, i.e., the implementation of dynamic BC behavior becomes now possible for MODFLOW-based models. Such programs can be short and are much simpler to implement than modifying the source code of the numerical model itself. Even very complex conditions can be represented with comparably short Python programs.

The contribution shows how pymf6 can enable users to solve complex groundwater modeling problems with reasonable effort. Among others, examples for flow and solute transport models involving dynamic BCs will be shown. The examples will demonstrate how pymf6 can be used to define problem-specific groundwater models that can help solve complex interactions between BCs. These examples will, among others, also involve advanced technical BCs such as a hydraulic barrier used for containing subsurface contaminations. In the presented example the hydraulic barrier is formed by multiple wells, each equipped with dynamic pumping rates which are adaptively regulated by defining not-to-be-exceeded threshold for the process variable of solute concentration.

Veranstaltung:	MODFLOW and More 2024
Titel:	Better Groundwater Models with MODFLOW 6 and Dynamic Boundary Conditions
Ort:	Princeton, USA
Datum:	Juni 2024
Autoren:	Mike Müller; Lúcia Pedrosa; Martin Binder; Christian Engelmann

Abstract

Realistic values of boundary conditions (BCs) are essential for robust results of groundwater models. Such values need to be defined before a model run. However, some BCs depend on the current state of a model not known a priori. For the example of a geothermal doublet system: The injection well can only re-infiltrate the amount of water which was previously pumped from the hydraulically connected extraction well. At the same time, the groundwater level in the extraction well should desirably not fall below a predefined level to protect the installed pump from falling dry. Such a scenario would likely require multiple model runs, adjusting pumping and infiltration rates based on results of a previous run. This manual workaround can become practically unfeasible for even more complex settings.

Our contribution, the open-source Python package `pymf6`, offers a simplified approach to define dynamic BCs by programmatically interacting with a model at runtime. This allows for finding a solution for the aforementioned scenario implementing a single model run. Hereby, dynamic BCs change their values during a model run according to rules defined by the modeler. While this dynamic approach is also possible by modifying the original MODFLOW Fortran code, this would require advanced programming skills. Here, MODFLOW 6 (MF6) exposes a well-defined programming interface that is used by our package.

Examples for flow and solute transport involving dynamic BCs will be presented to show how `pymf6` can enable users to solve complex groundwater modeling problems with reasonable effort. Our approach can be further extended to couple MF6 with other analytical or numerical models. The solution time steps in MF6 can be controlled on a very fine level, which will also allow a tight integration of time step-based models by interfacing solution loops of multiple models. Therefore, solutions can be smoother than applying an operator splitting approach.

Better Groundwater Models with MODFLOW 6 and Dynamic Boundary Conditions

MODFLOW and More 2024

Princeton, June 2024

Presenter: Dr.-Ing. Mike Müller[1], mmueller@hydrocomputing.com

Team: Lúcia Pedrosa[2], Dr.-Ing. Martin Binder[2, 3] Dr. Christian Engelmann[2],

Funded by Deutsche Bundesstiftung Umwelt (DBU) - grand number AZ 37733/01 - 33

[1] hydrocomputing GmbH & Co. KG, 04158 Leipzig, Germany

[2] Technische Universität Bergakademie Freiberg, Hydrogeologie und Hydrochemie, 09599 Freiberg (Sachsen), Deutschland

[3] Universität Basel, Hydrogeologie / Angewandte und Umweltgeologie, 4056 Basel, Schwitzerland

Veranstaltung:	Simulation in den Umwelt- und Geowissenschaften – Workshop (GI Fachgruppe 4.6.3, ASIM Fachgruppe SUG)
Titel:	Practical implementation of dynamic boundary conditions in flow and transport models: Three exemplary models
Ort:	Leipzig, Deutschland
Datum:	10. April 2024
Autoren:	Lúcia Pedrosa; Martin Binder; Christian Engelmann; Mike Müller

TU BERGAKADEMIE FREIBERG · TUBAF Die Ressourcenuniversität. Seit 1765. · hydrocomputing

Practical implementation of dynamic boundary conditions in flow and transport models:

Three exemplary models

DBU Deutsche Bundesstiftung Umwelt

(Leipzig, *April 10th, 2024*)

Lúcia Pedrosa [1], Christian Engelmann[1], Martin Binder[1,2] & Mike Müller[3]
[1] TU Bergakademie Freiberg, Institute for Geology, Chair of Hydrogeology and Hydrochemistry, Gustav-Zeuner-Straße 12, 09599 Freiberg, Germany
[2] University of Basel, Department of Environmental Sciences, Hydrogeology /Applied and Environmental Geology, Bernoullistrasse 32, 4056 Basel, Switzerland
[3] hydrocomputing GmbH & Co. KG, 04158 Leipzig, Germany

D.3. Poster

Titel:	Runtime Control of Numerical Groundwater Models
Ort:	Basel, Schweiz
Datum:	2023
Autoren:	Lúcia Pedrosa; Christian Engelmann; Martin Binder; Mike Müller

Runtime Control of Numerical Groundwater Models

Lúcia Pedrosa[1], Christian Engelmann[1] , Martin Binder[1,2] , Mike Müller[3]

[1] TU Bergakademie Freiberg, Chair of Hydrogeology and Hydrochemistry, Gustav-Zeuner-Straße 12, 09599 Freiberg, Germany
[2] University of Basel, Applied and Environmental Geology, Bernoullistrasse 32, 4056 Basel, Switzerland
[3] hydrocomputing GmbH & Co. KG, Zur Schule 20, 04158 Leipzig, Germany

Motivation

Dynamic adaptation of the boundary conditions at runtime to enable interactive model (compartments) and to reduce excessive runtimes.

Methods

- The current version of MODFLOW (MF6) incorporates the concept of Basic Model Interface (BMI) [1].
- *Pymf6* [2] is based on *xmipy* [3], an extension to *bmi-python* [4] that implements all abstract methods and can accesses the MF6 DLL with ctypes [5] (Fig. 1).
- In contrast to *xmipy*, *pymf6* provides a more pythonic interface and much less boilerplate code for typical usages.

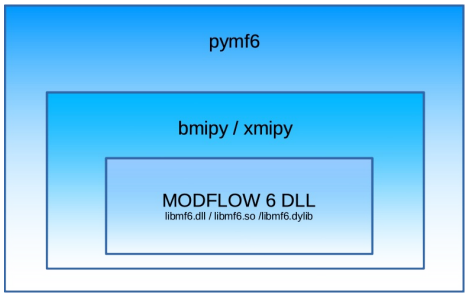

Fig.1 Architecture of *pymf6*

Conclusion

The tool performed accordingly to a simple set up. Other additions to the *pymf6*-aided model workaround may cover the coupling of numerically calculated model grid cell states to other (semi-)analytical or numerical model, such as rainfall-runoff, surface-groundwater interaction or even management tools.

Application of the tool

- Dynamic control of a well extraction or injection rate by setting a water level as threshold

Program 1:
HEAD-CONTROLLED WELL

```python
from pymf6.mf6 import MF6

def run_model(nam_file):
    mf6 = MF6(nam_file=nam_file)
    head = mf6.vars['SLN_1/X']
    wel_index = 44
    tolerance = 0.01
    head_limit = 0.5
    upper_limit = head_limit + tolerance
    lower_limit = head_limit - tolerance
    wel = mf6.vars['HEADCONWELL/WEL_0/BOUND']
    been_below = False
    for step in mf6.steps():
        if step < 21:
            if head[wel_index] <= lower_limit:
                wel[:] = wel[:] * 0.9
                been_below = True
            elif been_below and head[wel_index] >= upper_lir
                wel[:] = wel[:] * 1.1


if __name__ == '__main__':
    run_model(r'models/pymf6/mfsim.nam')
```

Program 1. User-written program to set up a well (at cell 44), specify a threshold (upper and lower limit +/- tolerance) to control the extraction when the water level is above or injection rate when it is bellow.

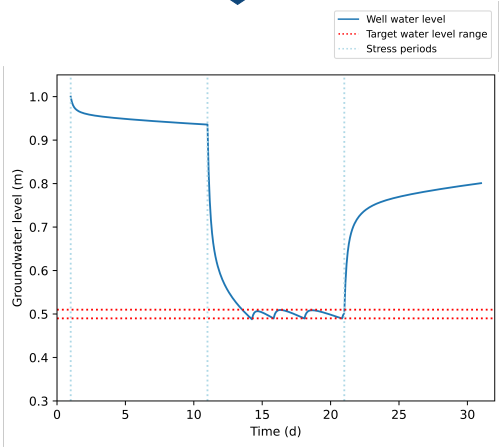

Fig.2 Calculated groundwater level at grid cell with pumping well for model run controlled by pymf6.

- Dynamic control of hydraulic barrier by setting a concentration threshold for the protected area

Program 2:
HYDRAULIC BARRIER

```python
from pymf6.mf6 import MF6

concs_rates = [
    (2.5, -0.6),
    (2.4, -0.5),
    (2.2, -0.4),
    (2.1, -0.3),
    (2.0, -0.2),
    (1.9, -0.1),

mf6 = MF6('models/pymf6')

conc_array = mf6.vars['SLN_2/X'].reshape(10, 10)
wel = mf6.vars['TRANSPORT/WEL-1/BOUND']

steps = mf6.steps()

for time_step in steps:
    modelled_conc = conc_array[1:-1, -1].max()
    for conc, rate in concs_rates:
        if modelled_conc > conc:
            wel[:] = rate
            break
    else:  # no `break` reached
        wel[:] = 0
```

Program 2. User-written program to set up a concentration threshold (1.9) to control hydraulic barrier's injection rate and avoid contamination to reach the protected area (right side of the domain).

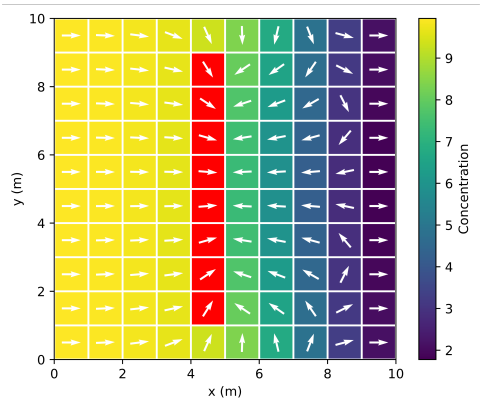

Fig.3 Calculated concentration with Hydraulic barrier (red cells) for model run controlled by pymf6.

References:

[1] BMI https://bmi-spec.readthedocs.io/en/latest
[2] pymf6 https://pymf6/readthedocs.io/en/latest/index.html
[3] xmipy https://github.com/Deltares/xmipy
[4] bmi-python https://github.com/csdms/bmi-python/blob/master/bmipy/bmi.py
[5] ctypes https://docs.python.org/3.11/ library/ctypes.html

Lúcia Pedrosa
TU Bergakademie Freiberg | Chair of Hydrogeology and Hydrochemistry I Address: Gustav-Zeuner-Straße 12 | 09599 Freiberg | Phone: +49 3731393118 | Lucia.Pedrosa@geo.tu-freiberg.de