

Hydroinformatik - SoSe 2026

UW-BHW-414-15: Finite-Differenzen-Methode

Prof. Dr.-Ing. habil. Olaf Kolditz

¹Helmholtz Centre for Environmental Research – UFZ, Leipzig

²Technische Universität Dresden – TUD, Dresden

³Center for Advanced Water Research – CAWR

⁴TUBAF-UFZ Center for Environmental Geosciences – C-EGS, Freiberg / Leipzig

Dresden, 26.06.2026

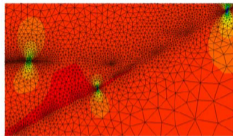
Zeitplan: Hydroinformatik I+II

Sommersemester 2026: Stand: 06.04.2026

Nr.	KW	Datum	ID	Thema
01+02	16	17.04.2026	UW-BHW-414-01/02	Einführung in die Vorlesung, Umweltinformatik
03	16	17.04.2026	UW-BHW-414-03	Werkzeuge, Hello World (in C++)
05	17	24.04.2026	UW-BHW-414-04	Selbststudium: Software-Installationen
07	19	08.05.2026	UW-BHW-414-05	Objekt-Orientierte Programmierung: C++, Klassen
09	20	15.05.2026	UW-BHW-414-06	Programmiersprache Python
11	21	22.05.2026	UW-BHW-414-07/08	Modellierung, Digitalisierung - Wasser 4.0
00	22	29.05.2026		Vorlesungsfreie Woche
13	23	05.06.2026	UW-BHW-414-09/10	KI, Maschinelles Lernen, Neuronale Netzwerke
15	24	12.06.2026	UW-BHW-414-11/12	Kontinuumsmechanik, Hydromechanik
17	25	19.06.2026	UW-BHW-414-13/14	Differentialgleichungen, Näherungsverfahren
19	26	26.06.2026	UW-BHW-414-15	Finite-Differenzen, explizite Verfahren
21	27	03.07.2026	UW-BHW-414-K	Finite-Differenzen, implizite Verfahren
23	28	10.07.2026	UW-BHW-414-L	Gerinnehydraulik, Grundwasserhydraulik
25	29	17.07.2026	UW-BHW-414-M	Grundwasserhydraulik
27	30	24.07.2026	UW-BHW-414-N	Zusammenfassung, Klausurvorbereitung

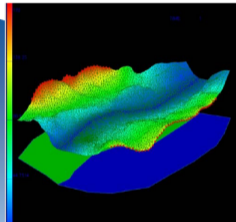
- 1 UW-BHW-414-15: Finite-Differenzen-Methode
 - Semesterplan

$$\frac{d\psi}{dt} = \frac{\partial\psi}{\partial t} + \mathbf{v}^E \nabla\psi$$

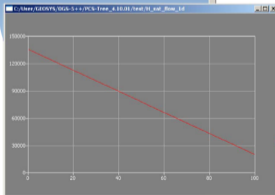
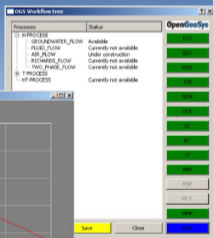


Basics
Mechanik

Anwendung



Numerische
Methoden



Programmierung
Visual C++

Prozessverständnis

0 Jupyter installation: Probleme? (\Rightarrow 11)

1 Grundlagen der Finite Differenzen Methode

2 Approximation methods (FDM, FEM)

3 Finite difference method – FDM (Ch. 3)

4 Taylor series expansion

5 Derivatives

6 Diffusion equation

7 Implizites FDM Verfahren

Jupyter

- Jupyter Notebook
- Jupyter Lab
- Browser-basiert

- "The Jupyter Notebook · The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, ..."
- Webseite: <https://jupyter.org/>
- Vorteil: funktioniert auf allen Rechnern
- ... ein Teil unserer (neuen) Übungen machen wir mit Jupyter Notebooks (>> Demo)



Jupyter: Example

The screenshot shows a JupyterLab window titled "HyBHW-1-01-02". The browser address bar shows "localhost:8888/notebooks/HyBHW-1-01-02-jymb". The notebook content includes the logo of Technische Universität Dresden and the text "Prof. Dr.-Ing. habil. Olof Klocke, Hydromechanik (HyBHW-1-01)". Below this is a section titled "Exercise 2 - Figures" with a code cell. The code imports matplotlib and numpy, defines a range of years and a list of publications, and uses plt.bar to create a bar chart. The resulting bar chart is titled "Hydromechanik I 2019 - Notenspiegel" and shows the number of points for each year from 2010 to 2019.

```
In [2]: from matplotlib import Figure, FigureCanvas
import matplotlib.pyplot as plt
import numpy as np

year = np.arange(2010, 2020)
publications = [1, 1, 7, 4, 8, 7, 6, 7, 3, 2, 3]

fig, ax = plt.subplots()

ax.set_title('Hydromechanik I 2019 - Notenspiegel')
ax.set_ylabel('Anzahl von Noten')

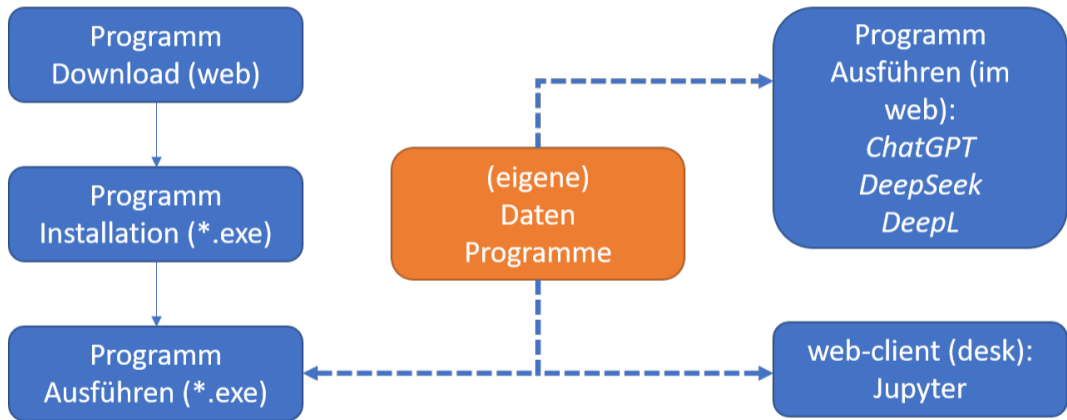
plt.bar(year, publications)
plt.xticks(year, ('1.0', '1.3', '1.7', '2.0', '2.3', '2.7', '3.0', '3.3', '3.7', '4.0', '5.0'))
plt.grid(True)
plt.show()
```

Year	Number of Points
2010	1
2011	1
2012	7
2013	4
2014	8
2015	7
2016	6
2017	7
2018	3
2019	2

The screenshot shows a JupyterLab window titled "JupyterLab" with a file browser on the left. The main area displays two plots. The first plot is titled "Benchmark 1: Sneddon (Opening profile)" and shows a parabolic curve of U_0/U_0^* versus X/λ_0 . The second plot is titled "Benchmark 2: Propagating straight fracture (pressure and crack length)" and shows a plot of p/p_0 versus W/W_0 . The code cell below the second plot shows the plotting commands.

```
[7]: plt.plot(np.arange(0, 1), x_c, np.arange(length), w, 'black', label='Closed form solution')
plt.plot(np.arange(x_0, 1), x_c, np.arange(length_0, 1), p, 'blue', label='p', fillstyle='none', label='Lab')
```

`https://jupyter.org/install`



Jupyter Notebook

- ▶ Zusammenführen von C++ und Python Funktionen

Zusammenführen von Funktionen (C++ und Python)

Jupyter Notebooks

Hydroinformatik-UW BHW-414 | EX09-jupyter-notebook

localhost:8888/notebooks/EX09-jupyter-not... Zusammenfassen

Jupyter EX09-jupyter-notebook Last Checkpoint: 7 minutes ago

File Edit View Run Kernel Settings Help Trusted

Technische Universität Dresden

Professur für Angewandte Umweltsystemanalyse an der TU Dresden
Prof. Dr.-Ing. habil. Olaf Kolditz
Hydroinformatik (UW-BHW-414)
[Lehre-Webseite](#)

Übung: EX09 Jupyter Notebook

In dieser Übung werden die beiden letzten Übungen, Funktionsberechnungen mit C++ und Python, in einem Jupyter Notebook zusammengeführt. Dies zeigt einen der Vorteile von Jupyter Notebook, in dem verschiedene Aufgaben in einer Web-Umgebung zusammengeführt und ausgeführt werden.

Inhalt:

- Funktionsberechnung mit dem C++ code und Ergebnisse als Datei schreiben
- Funktionsberechnung mit dem Python code
- Vergleich der Ergebnisse
- KI Vorschlag für Funktionsberechnungen und Vergleich

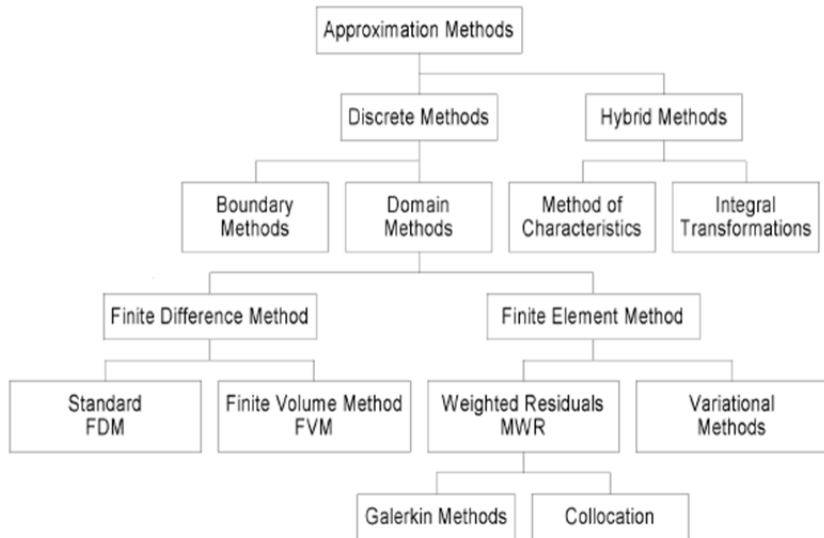
EX08a: Funktionsberechnung mit C++

```
[5]: g++ EX08a-function.cpp  
la.exe
```

Jupyter Notebooks:

- ▶ internet-fähig
- ▶ funktionieren zellenbasiert, Daten bleiben erhalten
- ▶ Beschreibung und Berechnungen, grafische Darstellungen
- ▶ ...

Finite-Differenzen-Methode (FDM)



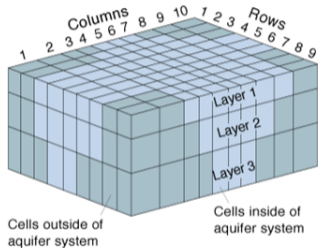


Figure 2. Example of model grid for simulating three-dimensional ground-water flow.

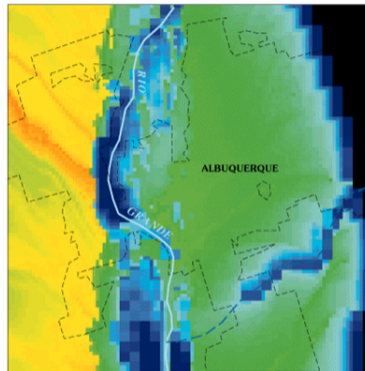
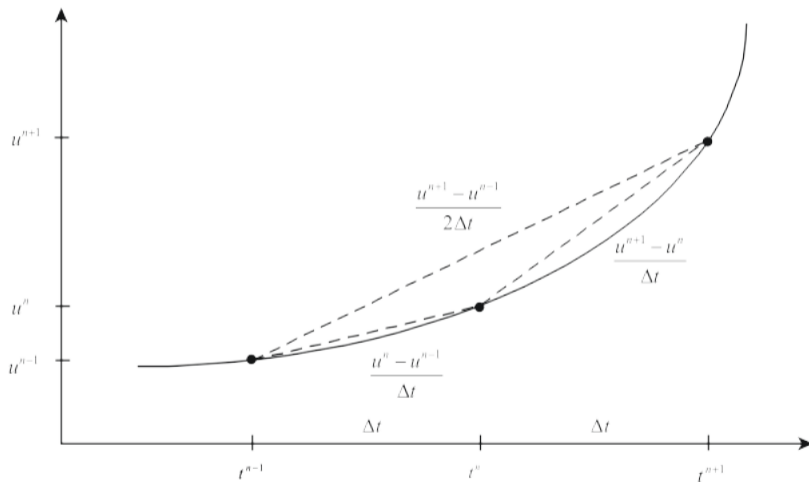


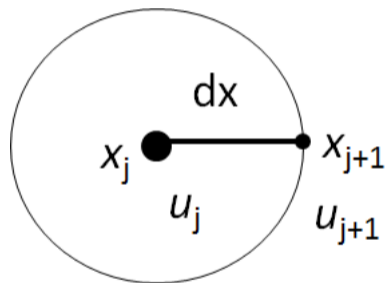
Figure 7. Application of particle tracking to estimate ground-water travel time.



<http://water.usgs.gov/pubs/FS/FS-121-97/images/fig7.gif>

Ableitungen





in time

$$u_j^{n+1} = \sum_{m=0}^{\infty} \frac{\Delta t^m}{m!} \left[\frac{\partial^m u}{\partial t^m} \right]_j \quad (1)$$

$$\Delta t = t^{n+1} - t^n$$

in space

$$u_{j+1}^n = \sum_{m=0}^{\infty} \frac{\Delta x^m}{m!} \left[\frac{\partial^m u}{\partial x^m} \right]_j \quad (2)$$

$$\Delta x = x_{j+1} - x_j$$

$$u_j^{n+1} = u_j^n + \Delta t \left[\frac{\partial u}{\partial t} \right]_j^n + \frac{\Delta t^2}{2} \left[\frac{\partial^2 u}{\partial t^2} \right]_j^n + \mathcal{O}(\Delta t^3) \quad (3)$$

$$u_{j+1}^n = u_j^n + \Delta x \left[\frac{\partial u}{\partial x} \right]_j^n + \frac{\Delta x^2}{2} \left[\frac{\partial^2 u}{\partial x^2} \right]_j^n + \mathcal{O}(\Delta x^3) \quad (4)$$

siehe auch Trunkationsfehler (Vorlesung Näherungsverfahren)

1. Ableitung

$$\left[\frac{\partial u}{\partial t} \right]_j^n = \frac{u_j^{n+1} - u_j^n}{\Delta t} - \frac{\Delta t}{2} \left[\frac{\partial^2 u}{\partial t^2} \right]_j^n + \mathcal{O}(\Delta t^2) \quad (5)$$

$$\left[\frac{\partial u}{\partial x} \right]_j^n = \frac{u_{j+1}^n - u_j^n}{\Delta x} - \frac{\Delta x}{2} \left[\frac{\partial^2 u}{\partial x^2} \right]_j^n + \mathcal{O}(\Delta x^2) \quad (6)$$

Forward difference approximation

$$\left[\frac{\partial u}{\partial x} \right]_j^n = \frac{u_{j+1}^n - u_j^n}{\Delta x} + 0(\Delta x) \quad (7)$$

Backward difference approximation

$$\left[\frac{\partial u}{\partial x} \right]_j^n = \frac{u_j^n - u_{j-1}^n}{\Delta x} + 0(\Delta x) \quad (8)$$

Central difference approximation

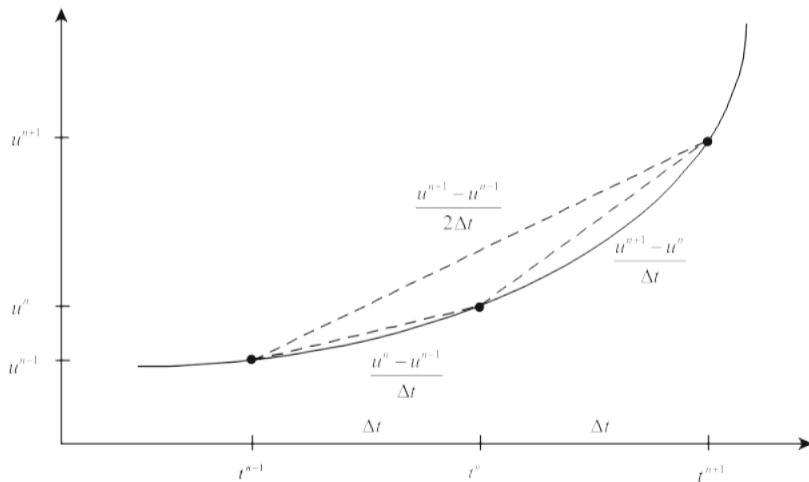
$$\left[\frac{\partial u}{\partial x} \right]_j^n = \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + 0(\Delta x^2) \quad (9)$$

$$\begin{aligned}u_{j+1}^n &= u_j^n + \Delta x \left[\frac{\partial u}{\partial x} \right]_j^n + \frac{\Delta x^2}{2} \left[\frac{\partial^2 u}{\partial x^2} \right]_j^n + \mathcal{O}(\Delta x^3) \\u_{j-1}^n &= u_j^n - \Delta x \left[\frac{\partial u}{\partial x} \right]_j^n + \frac{\Delta x^2}{2} \left[\frac{\partial^2 u}{\partial x^2} \right]_j^n - \mathcal{O}(\Delta x^3)\end{aligned}\tag{10}$$

Central difference approximation

$$\left[\frac{\partial u}{\partial x} \right]_j^n = \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + \mathcal{O}(\Delta x^2)\tag{11}$$

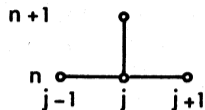
Ableitungen



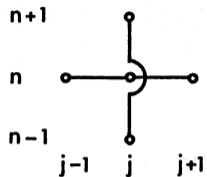
$$\begin{aligned}\left[\frac{\partial^2 u}{\partial x^2}\right]_j^n &\approx \frac{1}{\Delta x} \left(\left[\frac{\partial u}{\partial x}\right]_{j+1}^n - \left[\frac{\partial u}{\partial x}\right]_j^n \right) \\ &\approx \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}\end{aligned}\quad (12)$$

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_j^n = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} + \frac{\Delta x^2}{12} \left[\frac{\partial^4 u}{\partial x^4}\right]_j^n + \dots \quad (13)$$

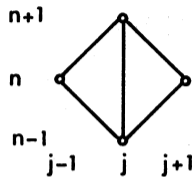
Übersicht Differenzenverfahren



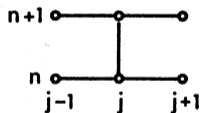
FTCS



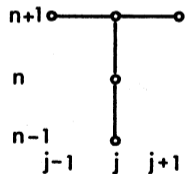
Richardson



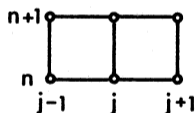
DuFort-Frankel



Crank-Nicolson



3LFI



Linear F.E.M./
Crank-Nicolson

$$\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0 \quad (14)$$

Analytical solution for diffusion equation (Skript 5.2.2)

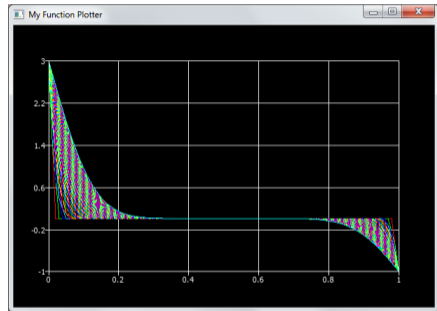
- ▶ Diffusion equation

$$\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0 \quad (15)$$

- ▶ Analytical solution

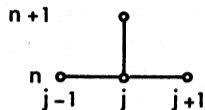
Einschränkungen: Nur für
symmetrische Randbedingungen

- ▶ K: validity

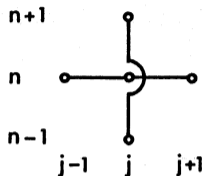


⇒ Übung

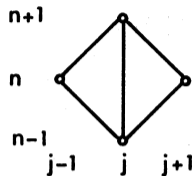
Übersicht Differenzenverfahren



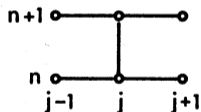
FTCS



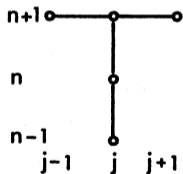
Richardson



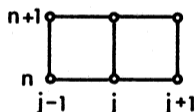
DuFort-Frankel



Crank-Nicolson



3LFI



Linear F.E.M. /
Crank-Nicolson

Explizite FDM - FTCS Verfahren (Skript 3.2.2/4.1)

- ▶ PDE for diffusion processes

$$\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0 \quad (16)$$

- ▶ forward time / centered space

$$\left[\frac{\partial u}{\partial t} \right]_j^n \approx \frac{u_j^{n+1} - u_j^n}{\Delta t} \quad \left[\frac{\partial^2 u}{\partial x^2} \right]_j^n \approx \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2} \quad (17)$$

- ▶ substitute

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} - \alpha \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2} = 0 \quad (18)$$

- ▶ FTCS scheme for diffusion equations

$$u_j^{n+1} = u_j^n + \frac{\alpha \Delta t}{\Delta x^2} (u_{j-1}^n - 2u_j^n + u_{j+1}^n), \quad Ne = \frac{\alpha \Delta t}{\Delta x^2}$$

Analysis of approximation schemes consists of three steps:

- ▶ Develop the **algebraic scheme**,
- ▶ Check **consistency** of the algebraic approximate equation,
- ▶ Investigate **stability** behavior of the scheme.

Analysis of approximation schemes consists of three steps:

- ▶ Develop the **algebraic scheme**,

$$u_j^{n+1} = u_j^n + \frac{\alpha \Delta t}{\Delta x^2} (u_{j-1}^n - 2u_j^n + u_{j+1}^n) \quad (20)$$

- ▶ Check **consistency** of the algebraic approximate equation,

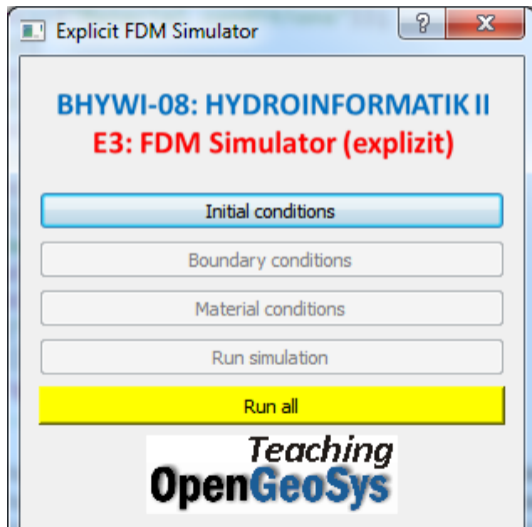
$$\lim_{\Delta t, \Delta x \rightarrow 0} |\hat{L}(u_j^n) - L(u[t_n, x_j])| = 0 \quad (21)$$

- ▶ Investigate **stability** behavior of the scheme.

$$Ne = \frac{\alpha \Delta t}{\Delta x^2} \leq 1/2 \quad (22)$$

Übungen

- (Qt Übung (alt): BHYWI-08-03-E)
- C++ Übung: EX08-fdm-explicit-C++
- Python Übung: EX08-fdm-explicit-python

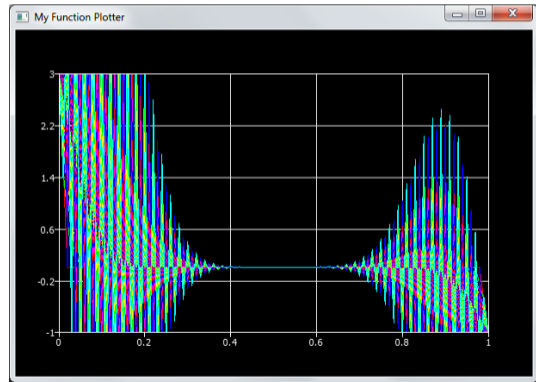
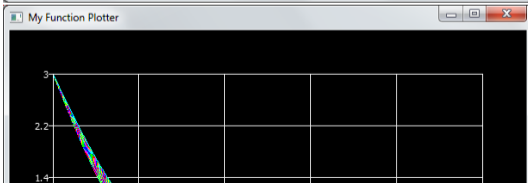
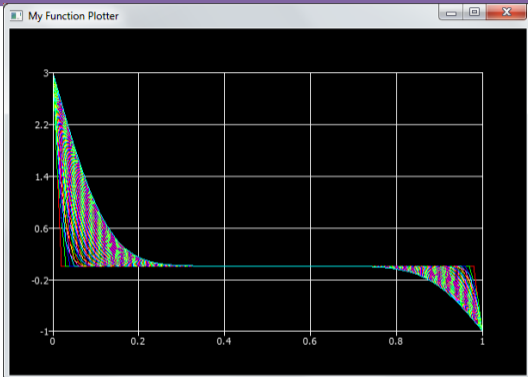


Dialog-Klasse: Konstruktor
Dialog::Dialog

- 1 Elemente
- 2 Connects
- 3 Layout
- 4 Datenstrukturen
(Speicherreservierung)

FDM Übung: Explizit Qt

Ergebnisse

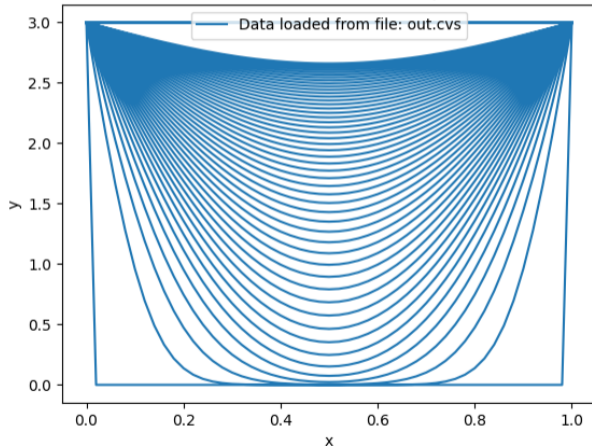


Übung: EX26 (C++): explizite FDM

```
1 echo Compilation
2 g++ main.cpp
3 echo Execution
4 a.exe
5 echo Ploting
6 data_from_file.py
7 echo End
```

Listing: Skript: C++ Berechnung > Python Grafik

hydroinformatics II (Olaf Kolditz)
Exercise BHYWI-08-03-for-python
Finite-Difference-Method (explicit)



Übung: EX26 (Python): explizite FDM

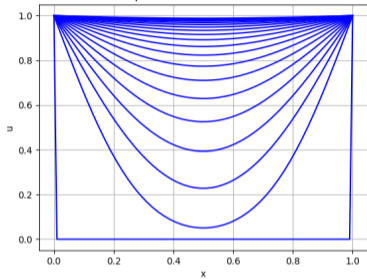
```
1 import math
2 import numpy as np
3 import matplotlib.pyplot as plt
4 #data structures
5 ##physical parameter
6 alpha = 1.0
7 ##numerical parameters (discretization)
8 nx = 10
9 x = np.zeros(nx+1)
10 dx = 1./nx
11 t = [0.01]
12 nt = 10 #wieviele Zeitschritte bis zum stationaeren
        Zustand
13 dt = 0.5 * dx*dx / alpha
14 Ne = alpha * dt / (dx*dx)
15 ##field function
16 u = np.zeros(nx+1)
17 uo = np.zeros(nx+1)
18 #initial condition
19 u_ic = 0.
20 for i in range(nx+1):
21     x[i] = 0
22     u[i] = 0
23     uo[i] = 0
24 #boundary conditions
25 u_bc_l = 1.
26 u_bc_r = 1.
27 u[0] = uo[0] = u_bc_l
28 u[nx] = uo[nx] = u_bc_r
29 #initial state
```

```
1 ...
2 #initial state
3 for i in range(0,nx+1):
4     x[i] = (float(i)/float(nx))
5 plt.plot(x,u,color='blue')
6 #fdm-explicit
7 for n in range(1,nt):
8     for i in range(1,nx):
9         u[i] = uo[i] + Ne *(uo[i-1] - 2*uo[i] + uo[i+1])
10    plt.plot(x,u,color='blue')
11    for i in range(1,nx):
12        uo[i] = u[i]
13 #plots
14 plt.title('EX08: explizite Finite-Differenzen-Methode')
15 plt.xlabel('x')
16 plt.ylabel('u')
17 plt.axis('tight')
18 plt.grid()
19 plt.savefig("fdm-explicit.png")
20 plt.show()
```

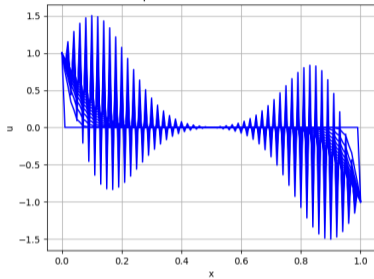
Listing: ...

Übung: EX26 (Python): explizite FDM

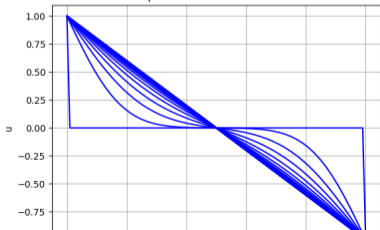
EX08: explizite Finite-Differenzen-Methode



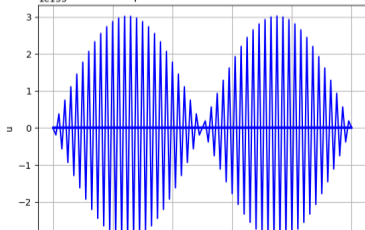
EX08: explizite Finite-Differenzen-Methode



EX08: explizite Finite-Differenzen-Methode

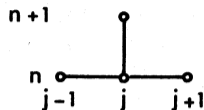


EX08: explizite Finite-Differenzen-Methode

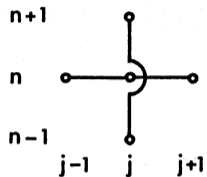


Alternative Verfahren

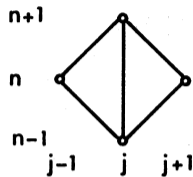
Explizite und implizite Differenzenverfahren



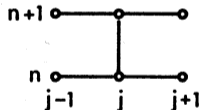
FTCS



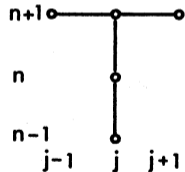
Richardson



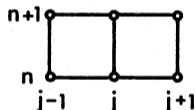
DuFort-Frankel



Crank-Nicolson



3LFI



Linear F.E.M./
Crank-Nicolson

Algebraische Schema:

$$\left[\frac{\partial^2 u}{\partial x^2} \right]_j^{n+1} \approx \frac{u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}}{\Delta x^2} \quad (26)$$

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} - \alpha \frac{u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}}{\Delta x^2} = 0 \quad (27)$$

$$\frac{\alpha \Delta t}{\Delta x^2} (-u_{j-1}^{n+1} + 2u_j^{n+1} - u_{j+1}^{n+1}) + u_j^{n+1} = u_j^n \quad (28)$$

Vorlesung

Übung

Selbststudium (Hausaufgabe)

Klausur

Übersicht

Hausaufgaben

- 1 Skalarprodukt: Schreiben sie das Skalarprodukt $\nabla \cdot \mathbf{v}$ in Komponentenschreibweise.
- 2 Mechanik: Was ist $\mathbf{v} \cdot \nabla \psi$? Physikalische Bedeutung des Terms
- 3 Mechanik: Was ist Φ^ψ ? Physikalische Bedeutung des Terms
- 4 Hydromechanik: Komponentenschreibweise $\nabla \cdot (\mathbf{v}\psi)$
- 5 Hydromechanik: Komponentenschreibweise $\nabla \cdot (\mathbf{D}^\psi \nabla \psi)$
- 6 Analytik: Prüfen Sie die Gültigkeit einer der Lösungen für die Diffusionsgleichung: (20), (21), (22) (siehe Vorlesung 5 >> HyBHW-S2-01-V05).
- 7 Analytik: Stellen Sie die ausgewählte analytische Lösung für die 1-D parabolische Differentialgleichung unter Verwendung der Übung EX06-parabolische-gleichung-1D.py dar. Ergänzen Sie Ihren Namen oder Matrikelnummer mit dem Befehl `plt.title("Name oder Matrikelnummer")`.
- 8 Numerik: Darstellung der numerischen Lösung (explizite FDM) für die 1-D parabolische Differentialgleichung (EX08-fdm-explicit-python). Produzieren Sie eine stabile und instabile Lösung.
- 9 Numerik: Darstellung der numerischen Lösung (implizite FDM) für die 1-D parabolische Differentialgleichung (EX09-fdm-implicit-python). Produzieren Sie die stationäre Lösung.

- zum Internet-Repository gehen (Webseite)
- Python-File editieren (Matrikel-Nummer oder Name)
- Programme zum Rechnen und Darstellen ausführen
- Ergebnis (Abbildung) in die Hausaufgaben einfügen

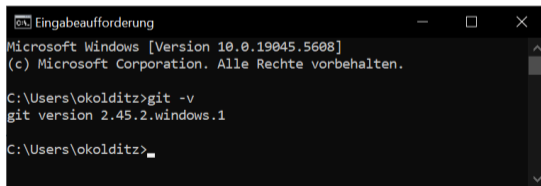
git | GitHub Praxis



The screenshot shows the GitHub homepage. At the top, it says "git --distributed-is-the-new-centralized". Below this, there are two paragraphs of text describing Git as a free and open source distributed version control system. To the right of the text is a diagram showing a network of nodes connected by lines, representing a distributed system. Below the text are four sections: "About", "Documentation", "Downloads", and "Community". The "About" section lists advantages of Git. The "Documentation" section mentions command reference pages. The "Downloads" section lists Git clients and binary releases for various platforms. The "Community" section encourages getting involved. At the bottom, there is a section for "Products providing Git hosting" with logos for Codeberg, sourcehut, tangled.sh, Forgejo, Gitea, GitLab, GitHub, and radicle.

- 1 Teaching Tutorial: Ausführlichere Beschreibung zum Nachlesen (Kapitel 1.1)
- 2 git Installation: Webpage
- 3 git Version

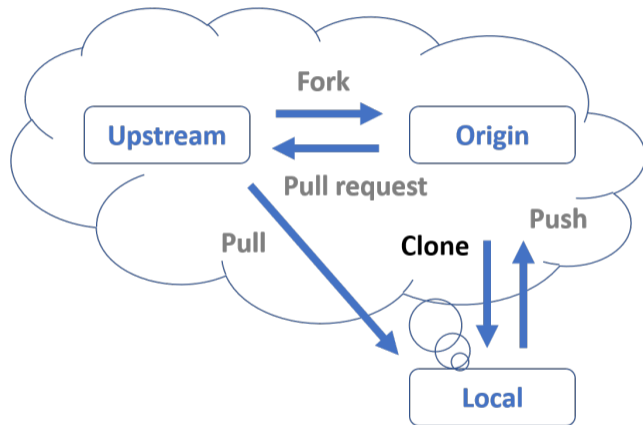
```
1 git -v
```



```
Eingabeaufforderung
Microsoft Windows [Version 10.0.19045.5608]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

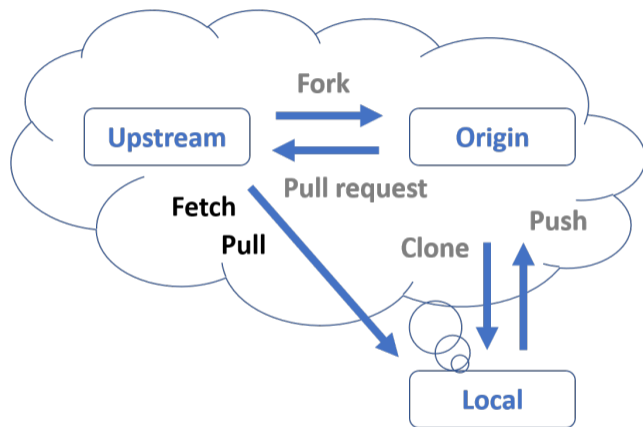
C:\Users\okolditz>git -v
git version 2.45.2.windows.1

C:\Users\okolditz>
```



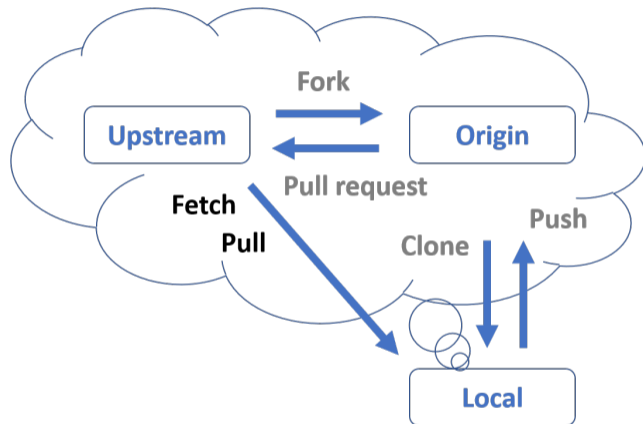
- 1 Quelle auswählen: HYDROINFORMATIK-I
- 2 **Code**: Pfad für's Clonen kopieren
- 3 geeignetes Verzeichnis auf eigenem Rechner auswählen (root)
- 4 Quelle clonen:

```
1 git clone https://github.com/OlafKolditz/HYDROINFORMATIK-I.git
```



- 1 ins richtige Verzeichnis gehen (HYDROINFORMATIK-I)
- 2 `fetch` : Repo anfragen: Gibt's was Neues?
- 3 `pull` : Änderungen in lokalem Repo runterladen

```
1 git fetch --all
2 git pull
```



Wenn Sie in ihrem Repo arbeiten, erzeugen Sie lokale Änderungen, die beim Aktualisieren aus dem Internet-Repo zu Konflikten führen. (Normalerweise erfolgt das lokale Arbeiten immer in sogenannten `branches`). Manchmal wollen Sie aber einfach auch die lokalen Änderungen **überschreiben** und den aktuellen Stand mit dem Internet-Repo auschecken, dann brauchen Sie noch einen `reset` Befehl:

- 1 `git fetch --all`
- 2 `git reset --hard HEAD`
- 3 `git pull`

Dozent: OlafKolditz

- ▶ Template für das Jupyter Notebook
- ▶ Repo auschecken (`git clone / git pull`) in das lokale Repo auf eigenem Rechner kopieren
- ▶ oder einzelne Datei herunterladen
- ▶ Datei in eigenes lokales Repo übernehmen ⇒

Mein Repo (KolditzTUDD)

- ▶ ⇒ Jupyter Notebook lokal auf eigenem Rechner bearbeiten
- ▶ Eigenes Internet Repo aktualisieren: `git push`
- ▶ ... Wiederholen (hierfür mit branching arbeiten)
- ▶ wenn das Notebook fertig ist, dann einen merge request in das Dozenten Notebook anmelden

- ▶ ⇒ Jupyter Notebook lokal auf eigenem Rechner bearbeiten
- ▶ Eigenes Internet Repo aktualisieren: git push
- ▶ ... Wiederholen (hierfür mit branching arbeiten)
- ▶ (wenn das Notebook fertig ist, dann einen merge request in das Dozenten Notebook anmelden)

```
1 //eigenes Repo lokal aktualisieren
2 git fetch --all
3 git pull
4 //branch - Name 'devs' kann frei gewaehlt
   werden
5 git checkout -b devs
6 git branch
7 //neue Datei hinzufuegen
8 git add hydroinformatik-jupyter-notebook.
   ipynb
9 //commit
10 git commit -m "Add mein ipynb zum
   Internet Repo"
11 //push
12 git push origin devs
13 //status pruefen
14 git status
```