Hydroinformatik - SoSe 2026

UW-BHW-414-16: FDM implizit

Prof. Dr.-Ing. habil. Olaf Kolditz

[1]Helmholtz Centre for Environmental Research – UFZ, Leipzig

[2]Technische Universität Dresden – TUD, Dresden

[3]Center for Advanced Water Research – CAWR

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

Dresden, 03.07.2026

Technische Universität Dresden

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-414-16	Finite-Differenzen, implizite Verfahren
23	28	10.07.2026	UW-BHW-414-17	Gerinnehydraulik, Grundwasserhydraulik
25	29	17.07.2026	UW-BHW-414-M	Grundwasserhydraulik
27	30	24.07.2026	UW-BHW-414-N	Zusammenfassung, Klausurvorbereitung

Technische Universität Dresden

Fahrplan für heute ...

1 UW-BHW-414-16: FDM implizit
- Semesterplan

Technische Universität Dresden

Fahrplan für heute ...
Implicit FDM for diffusion equation (section 4.2)

0. explizites Finite-Differenzen Verfahren

---

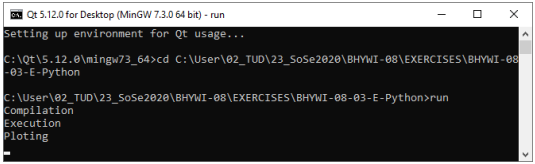
1. Implizites FDM Verfahren
2. Gleichungssysteme lösen
3. Übungen ins eigene Jupyter Notebook übertragen

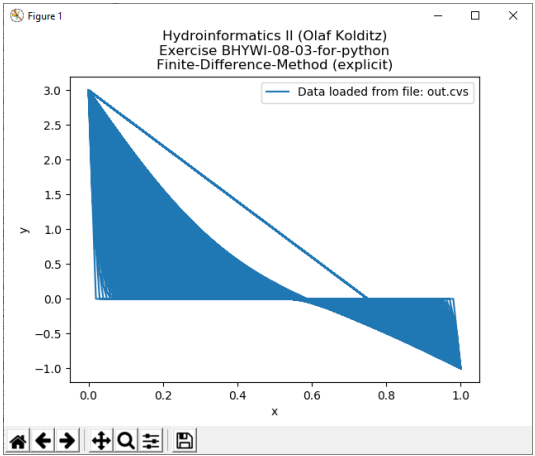
---

4. Gerinnehydraulik (nichtlineares Problem)

Technische Universität Dresden

Letzte Vorlesung: explizite FDM mit Python

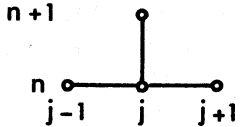




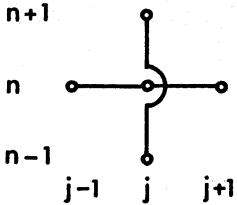
- Einfache Berechnung (keine Gleichungssysteme)
- Zeitschrittbegrenzung: Viele Zeitschritte erforderlich, um den stationären Zustand zu erreichen

>> Übung ins eigene Notebook übertragen

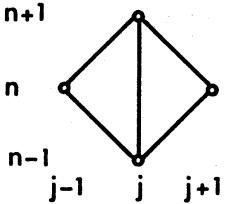
Letzte Vorlesung: Übersicht FDM Verfahren



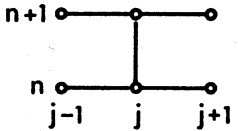
FTCS



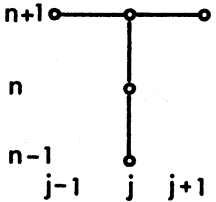
Richardson



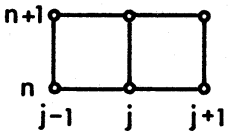
Du Fort-Frankel



Crank-Nicolson



3LFI



Linear F.E.M./
Crank-Nicolson

Technische Universität Dresden

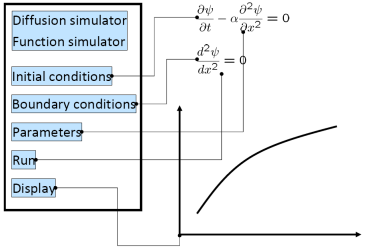
Letzte Vorlesung: explizite FDM - Zeitschrittbegrenzung

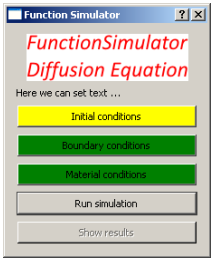
$$Ne = \alpha \frac{\Delta t}{\Delta x^2} \leqslant 0.5 \tag{1}$$

$$\Delta t \leqslant 0.5 \frac{\Delta x^2}{\alpha} \tag{2}$$

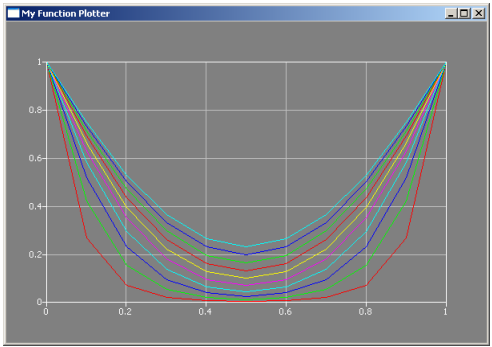
Technische Universität Dresden

Ziel der Vorlesung





<< alte Version: C++ Programmierung (sehr aufwendig)



>> Python / Jupyter Notebooks

Technische Universität Dresden

Implicite FDM - Theory #1 (Skript 4.2)

- PDE for diffusion processes

$$\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0 \tag{3}$$

- Time discretization

$$\left[\frac{\partial u}{\partial t}\right]_j^n \approx \frac{u_j^{\text{n+1}} - u_j^n}{\Delta t} \tag{4}$$

- Forward time / centered space

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_j^{\text{n+1}} \approx \frac{u_{j-1}^{\text{n+1}} - 2u_j^{\text{n+1}} + u_{j+1}^{\text{n+1}}}{\Delta x^2} \tag{5}$$

- (Explicit scheme: Current time / centered space)

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_j^n \approx \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2} \tag{6}$$

Technische Universität Dresden

Implicite FDM - Theory #2 (Skript 4.2)

- Substitute into PDE

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} - \alpha \frac{u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}}{\Delta x^2} = 0 \tag{7}$$

- Algebraic equation (index notation)

$$\frac{\alpha \Delta t}{\Delta x^2} \left(-u_{j-1}^{n+1} + 2u_j^{n+1} - u_{j+1}^{n+1}\right) + u_j^{n+1} = u_j^n \tag{8}$$

- Algebraic equation (matrix notation)

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{9}$$

- Explain steps with black board

Technische Universität Dresden

Implicite FDM - Theory #3 (Skript 4.2)

- Algebraic equation (matrix notation)

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{10}$$

- Algebraic equation (index notation)

$$Ne\,(-u_{j-1}^{n+1} + 2u_j^{n+1} - u_{j+1}^{n+1}) + u_j^{n+1} = u_j^n \tag{11}$$

- Let's take a closer look ...

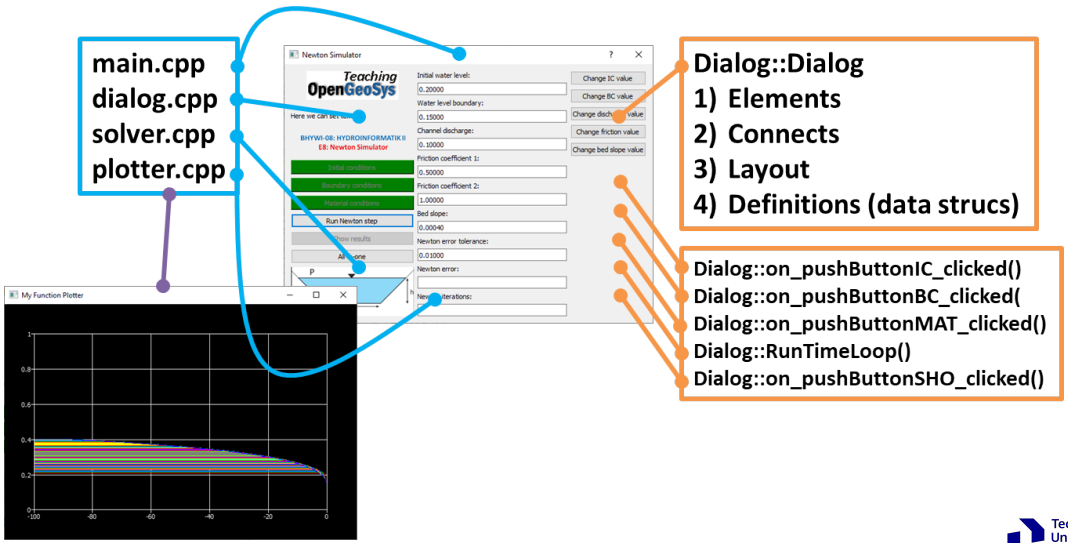
$$\underbrace{\begin{bmatrix} 1+2Ne & -Ne & & & \\ -Ne & \dots & \dots & & \\ & \dots & \dots & \dots & \\ & & \dots & \dots & -Ne \\ & & & -Ne & 1+2Ne \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \dots \\ u_{n-1} \\ u_n \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ \dots \\ b_{n-1} \\ b_n \end{bmatrix}}_{\mathbf{b}} \tag{12}$$

Technische Universität Dresden

Übungen

- **Qt Version: BHYWI-08-04-E**
- Python Übung: EX09-fdm-implicit

Qt in a nutshell



Implementation #1

- Data structures (as usual ...)

```
Dialog::Dialog(QWidget *parent) : QDialog(parent)
{
  matrix = new double[n*n];
  vecb = new double[n];
  vecx = new double[n];
}

Dialog::~Dialog()
{
  delete [] matrix;
  delete [] vecb;
  delete [] vecx;
}
```

Technische Universität Dresden

Implementation #2

- Functions (a pain in the neck ...)

```
AssembleEquationSystem();
Gauss(matrix,vecb,vecx,n);

void Dialog::AssembleEquationSystem()
{...
  int i,j;
  // Matrix entries
  for(i=0;i<n;i++)
  {
    vecb[i] = u_old[i]; // RHS Vektor
    for(j=0;j<n;j++)
    {
      matrix[i*n+j] = 0.0;
      if(i==j) // Hauptdiagonale
        matrix[i*n+j] = 1. + 2.*Ne;
      else if(abs((i-j))==1) // Nebendiagonalen
        matrix[i*n+j] = - Ne;
    }
  }
...}
```

Technische Universität Dresden

Implementation #3

- Boundary conditions - concept

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{13}$$

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \dots \\ u_n \end{bmatrix} = \begin{bmatrix} u_0 \\ 0 \\ \dots \\ 0 \end{bmatrix} \tag{14}$$

Technische Universität Dresden

Implementation #4

- Boundary conditions - implementation

```
void Dialog::AssembleEquationSystem()
{...
  // Treat boundary conditions
  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
    {
      if(i==0||i==n-1)
        matrix[i*n+j] = 0.0;
    }
  for(i=0;i<n;i++)
  {
    if(i!=0&&i!=n-1)
      continue;
    for(j=0;j<n;j++)
    {
       if(i==j)
         matrix[i*n+j] = 1.0;
       else
         matrix[i*n+j] = 0.0;
    }
  }
}
```

Technische Universität Dresden

Solving EQS - How to ... the magic Gauss function

- 1

$$a_{11}u_1 + a_{12}u_2 = b_1 \tag{15}$$
$$a_{21}u_1 + a_{22}u_2 = b_2 \tag{16}$$

- 2

$$a_{21}\frac{a_{11}}{a_{21}}u_1 + a_{22}\frac{a_{11}}{a_{21}}u_2 = \frac{a_{11}}{a_{21}}b_2 \tag{17}$$

- 3

$$\left(\frac{a_{22}a_{11}}{a_{21}} - a_{12}\right)u_2 = \frac{a_{11}}{a_{21}}b_2 - b_1 \tag{18}$$

- 4

$$u_2 = \frac{\frac{a_{11}}{a_{21}}b_2 - b_1}{\frac{a_{22}a_{11}}{a_{21}} - a_{12}} \tag{19}$$

Technische Universität Dresden

Implementation #5

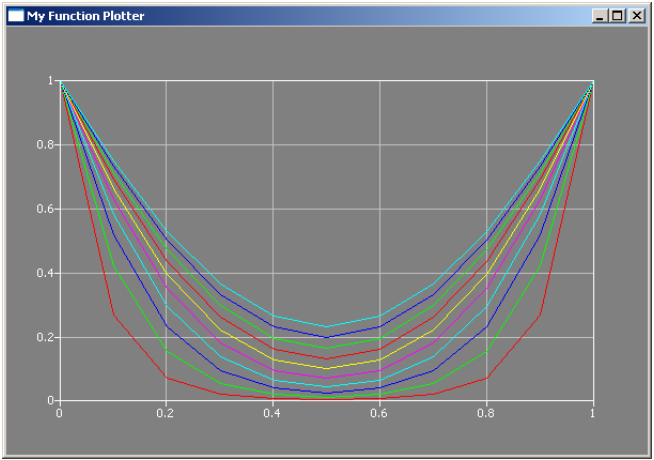
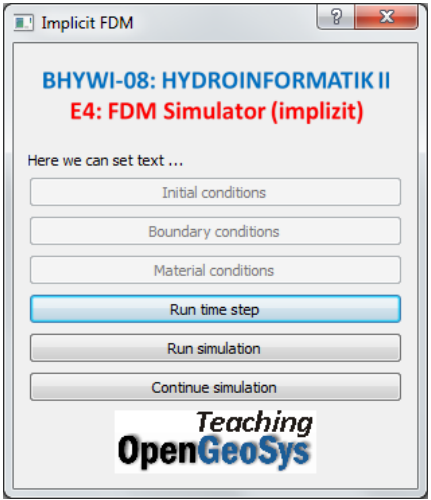


Fig.: Zeitliche Entwicklung des Diffusionsprofils - implizites Verfahren (Wahoo...)

Technische Universität Dresden

Implementation #6: Run multiple time steps



- Einzelne Zeitschritte
- Mehrere Zeitschritte
- Weiterführen der Berechnung

```
void RunTimeStep();
void RunTimeLoop();
void ContinueTimeLoop();
```

Technische Universität Dresden

Übungen

- Qt Version: BHYWI-08-04-E
- **C++/Python Übung: EX09-fdm-implicit**

EX09-fdm-implicit-C++

`run.bat`

- Compilation: `g++ main.cpp solver.cpp`
- Run: `a.exe`
- Plotting: `data_from_file.py`

Technische Universität Dresden

EX09-fdm-implicit-C++

```
#include <vector>
#include <fstream>
extern void Gauss(double *matrix, double *vecb, double *vecx, int g);
void AssembleEquationSystem();
int n = 101;
double* matrix;
double* vecb;
double* vecx;
double Ne;
vector<double> u_new, u_old;
std::ofstream out_file;
int main(int argc, char *argv[])
{
...
}
```

Listing: C++ code for implicit FDM: Data structures

Technische Universität Dresden

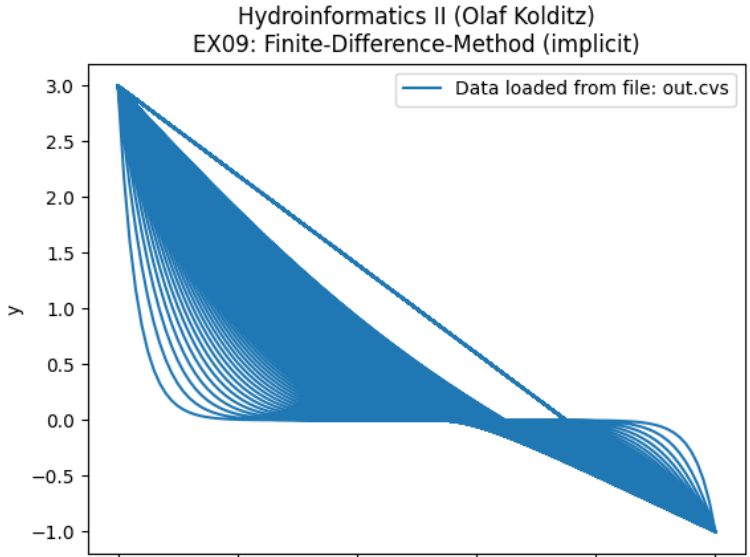
EX09-fdm-implicit-C++

```
int main(int argc, char *argv[])
{
//1-Definitionen
  //1.1 Diskretisierung
  //1.2 Loesungsvektoren
  //1.3 Kennzahlen, Parameter
  //1.4 Ausgabe
//2-Anfangsbedingungen
//3-Randbedingungen
//4-Diskretisierung
//6-Berechnung: FDM Verfahren
  //AssembleEquationSystem();
  //6.1 Matrizen berechnen
  //6.2 Randbedingungen einbauen
  //6.3 Gleichungsloeser: Gauss(matrix,vecb,vecx,nj);
  //6.4 Ergebnisse speichern
}
```

Listing: C++ code for implicit FDM: Code structure

Technische Universität Dresden

EX09-fdm-implicit-C++



Arbeit mit dem git Repo

```
#Aenderungen lokal erledigt

#show remote URL
git remote -v

#add to push list
git add hydroinformatik-jupyter-notebook.ipynb
git add .
#commit
git commit -m "short description what has been done"
#push
git push origin
git push origin [branch]

#advanced (verschiedene Repos)
git push --set-upstream origin [branch]
git push --set-upstream other-repo [branch]
```

Listing: Arbeit mit dem git Repo

Technische Universität Dresden