Hydroinformatik II

"Prozesssimulation und Systemanalyse"

HyBHW-1-02-9 @ 2021

Finite-Differenzen-Methode II

Olaf Kolditz

*Helmholtz Centre for Environmental Research – UFZ

[1]Technische Universität Dresden – TUDD

[2]Centre for Advanced Water Research – CAWR

18.06.2021 - Dresden

TECHNISCHE UNIVERSITÄT DRESDEN

Zeitplan: Hydroinformatik II

Datum	V	Thema	T
16.04.2021	01	Einführung in die Lehrveranstaltung | Tools	L
23.04.2021	02	Grundlagen: Kontinuumsmechanik	L
30.04.2021	03	Grundlagen: Hydromechanik	L
07.05.2021	04	Grundlagen: Partielle Partialgleichungen	L
14.05.2021	05a	Tools: Compiler, Python, Jupyter	E
14.05.2021	05b	Übung: Elliptische PDG	E
21.05.2021	06	Übungen: Übersicht und Werkzeuge	E
28.05.2021	–	Pfingsten	
04.06.2021	07	Grundlagen: Näherungsverfahren	L
11.06.2021	08	Numerik: Finite-Differenzen-Methode (explizit)	L
18.06.2021	09	Numerik: Finite-Differenzen-Methode (implizit)	L
25.06.2021	10	Reserve	L/E
02.07.2021	11	Übung: Diffusionsprozesse	E
09.07.2021	12	Übung: Gerinnehydraulik	E
16.07.2021	13	Übung: Grundwassermodellierung	E
23.07.2021	14	Beleg: Besprechung zur Vorbereitung	L

TECHNISCHE UNIVERSITÄT DRESDEN

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

1. EX08-fdm-explicit: Übung explizite FDM
2. HA: bis zum stationären Zustand rechnen
3. BHYWI-08-03E: Qt Version

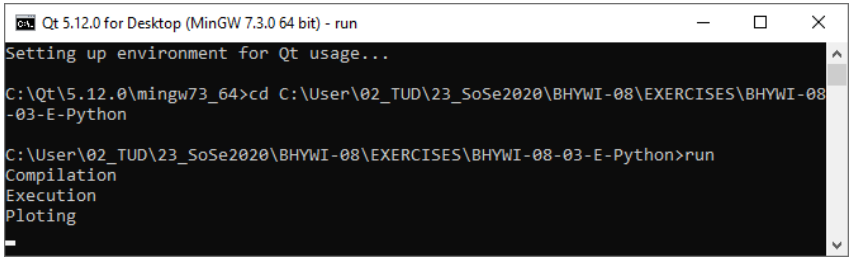
---

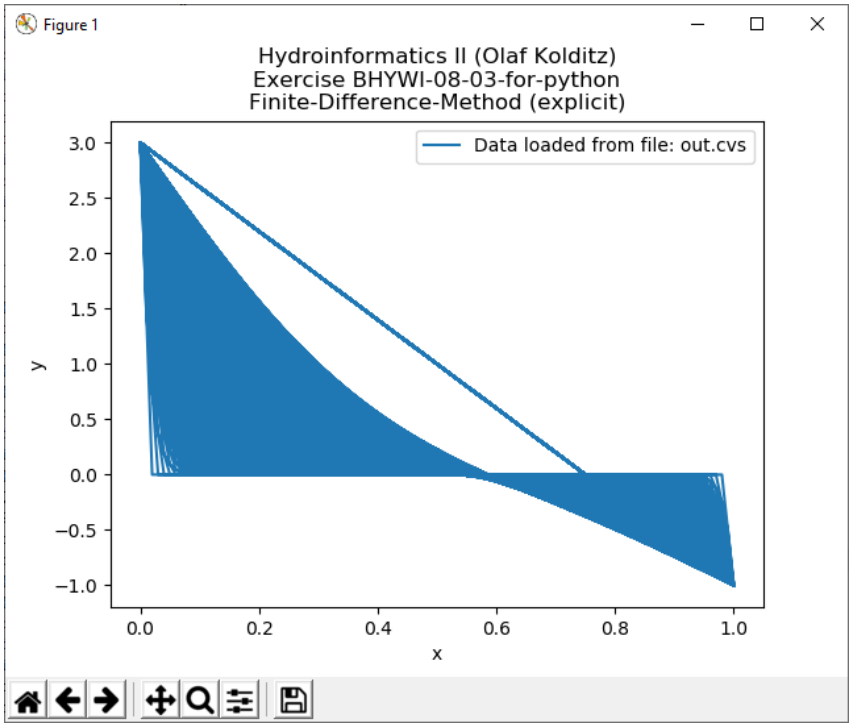
4. EX09-fdm-implicit: Übung implizite FDM
5. BHYWI-08-04E: Qt Version
6. (Tabelle Vorlesungen/Übungen/Hausaufgaben > neue Nummerierung)

`https://github.com/OlafKolditz/HYDROINFORMATIK-II`

TECHNISCHE UNIVERSITÄT DRESDEN

Letzte Vorlesung: explizite FDM mit Python





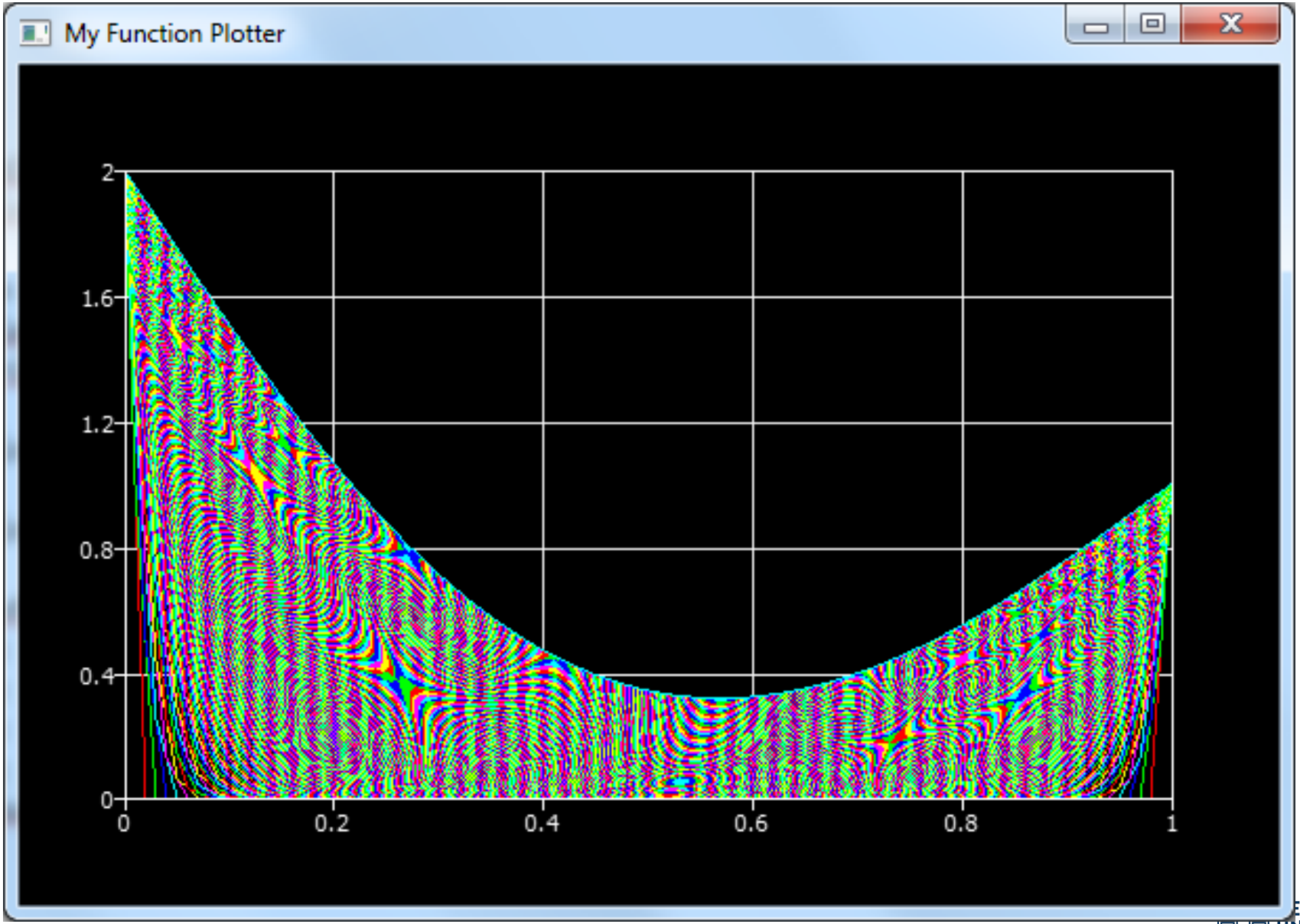
- Qt Installation (C++ Compiler und Konsole)
- Python Installation (weiter Pakete laden, `mathplotlib`)

BHYWI-08-03HW2 (Download von Webseite)

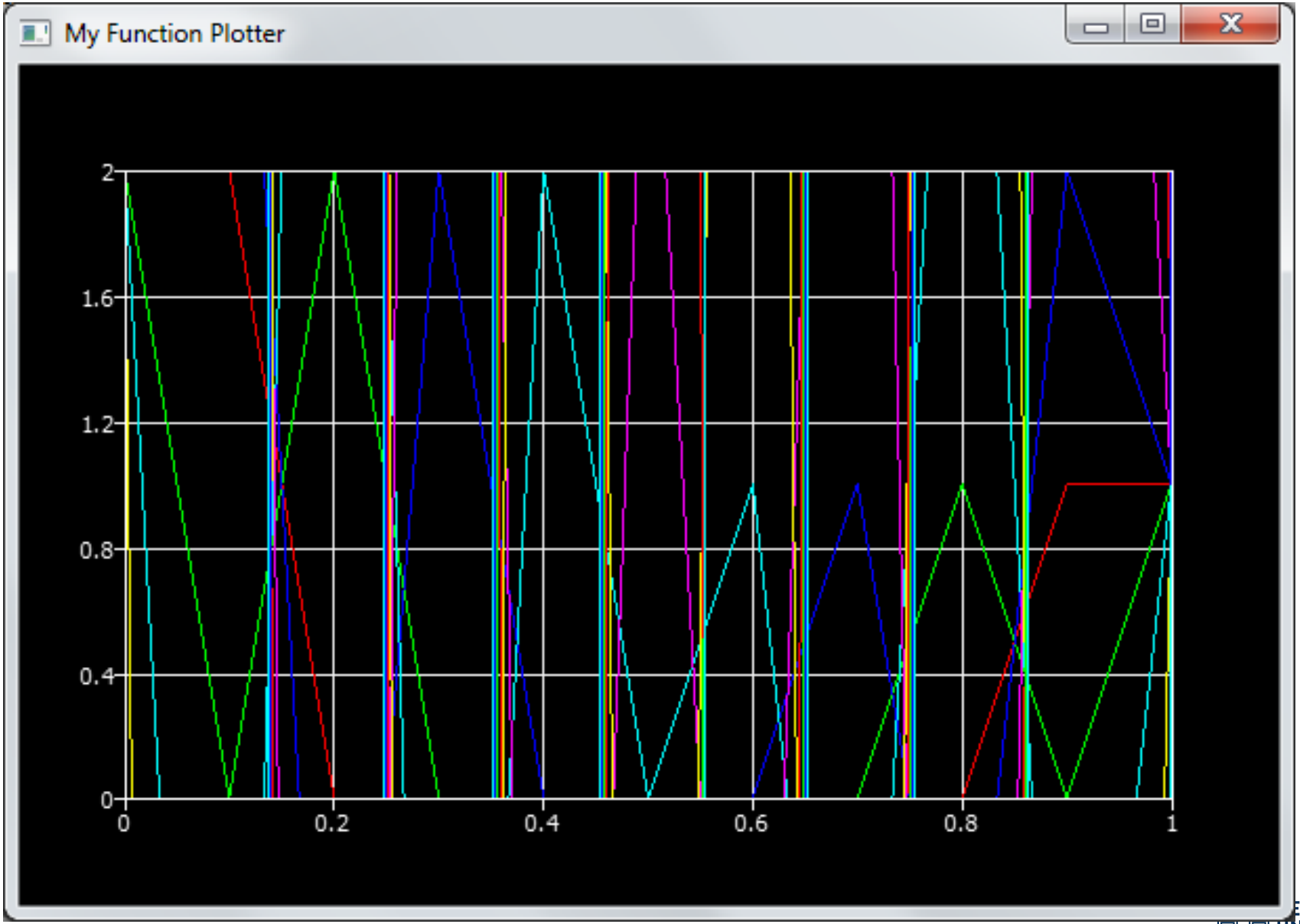
- Namen/Matrikelnummer in den Plot eintragen
- Nur jede 10. Kurve plotten

TECHNISCHE UNIVERSITÄT DRESDEN

Letzte Vorlesung: explizite FDM mit Qt



Letzte Vorlesung: explizite FDM - Zeitschrittbegrenzung

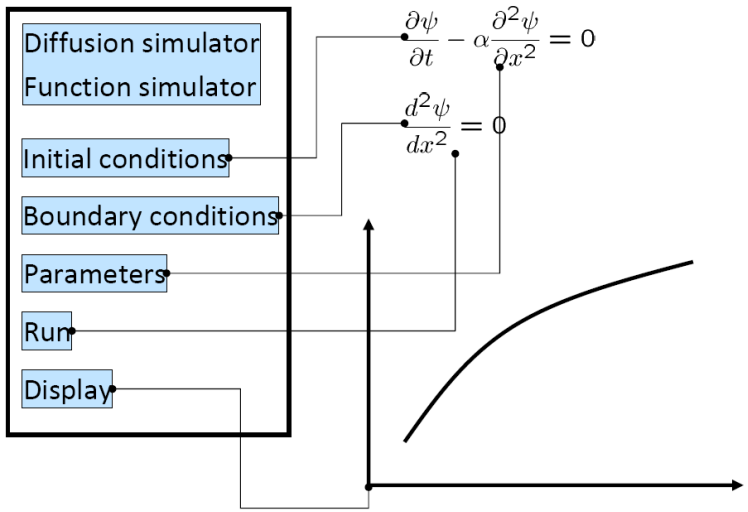


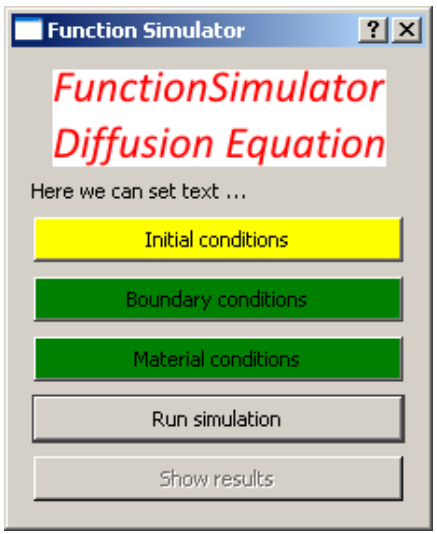
Letzte Vorlesung: explizite FDM - Zeitschrittbegrenzung

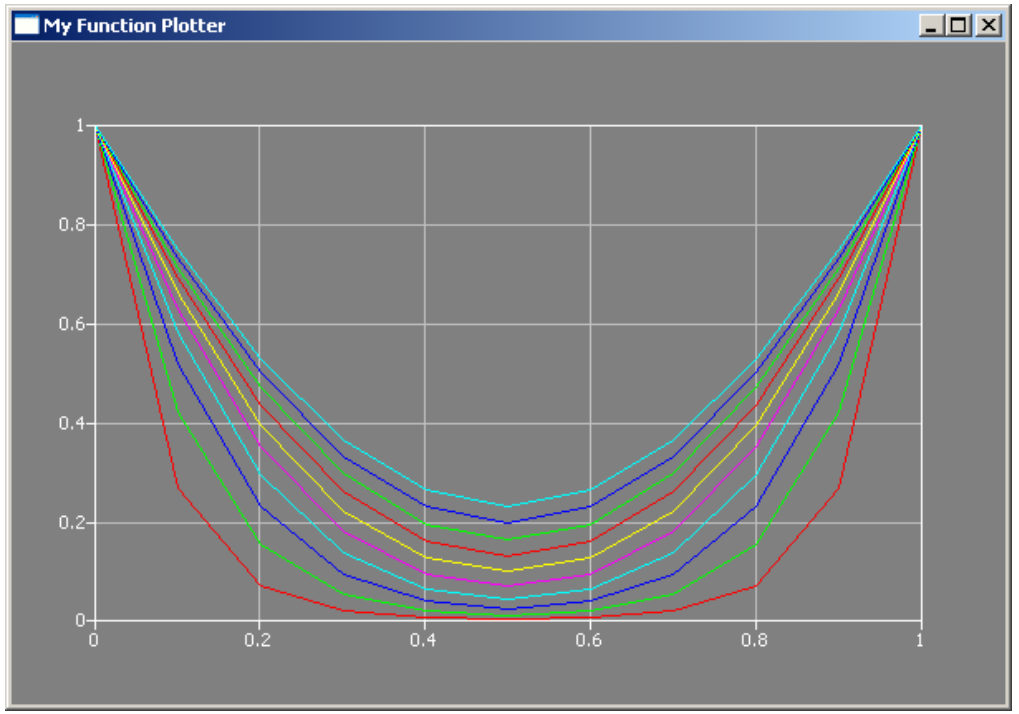
$$Ne = \alpha \frac{\Delta t}{\Delta x^2} \leq 0.5 \tag{1}$$

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

Ziel der Vorlesung







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

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

- Forward time / centered space

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

- (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)

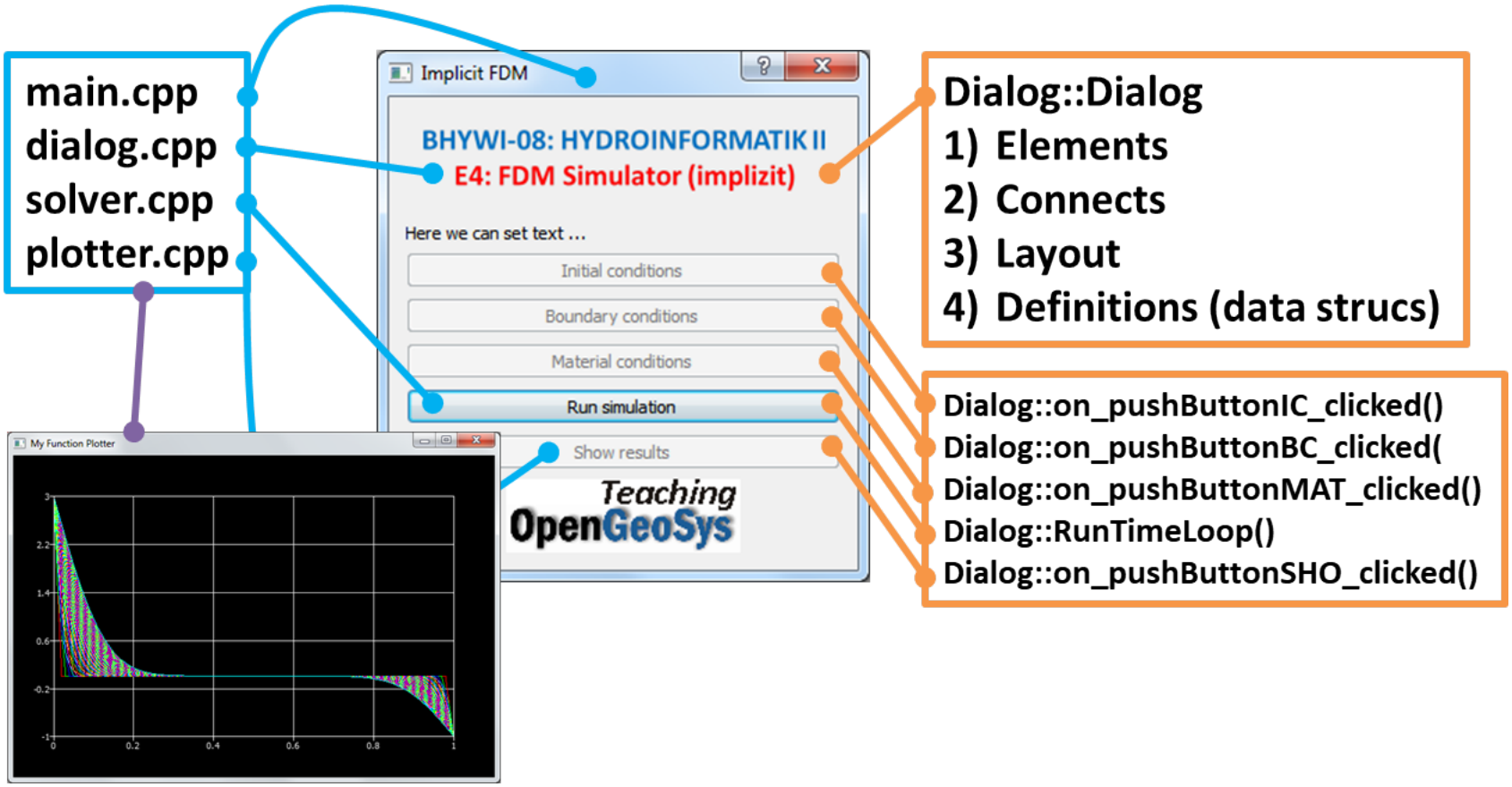
$$Ne\,(-u_{j-1}^{n+1} + 2u_j^{n+1} - u_{j+1}^{n+1}) + u_j^{n+1} = u_j^n \tag{11}$$

- Let's take a closer look ...

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

Qt Version

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;
}
```

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

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

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

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;
    }
  }
}
```

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

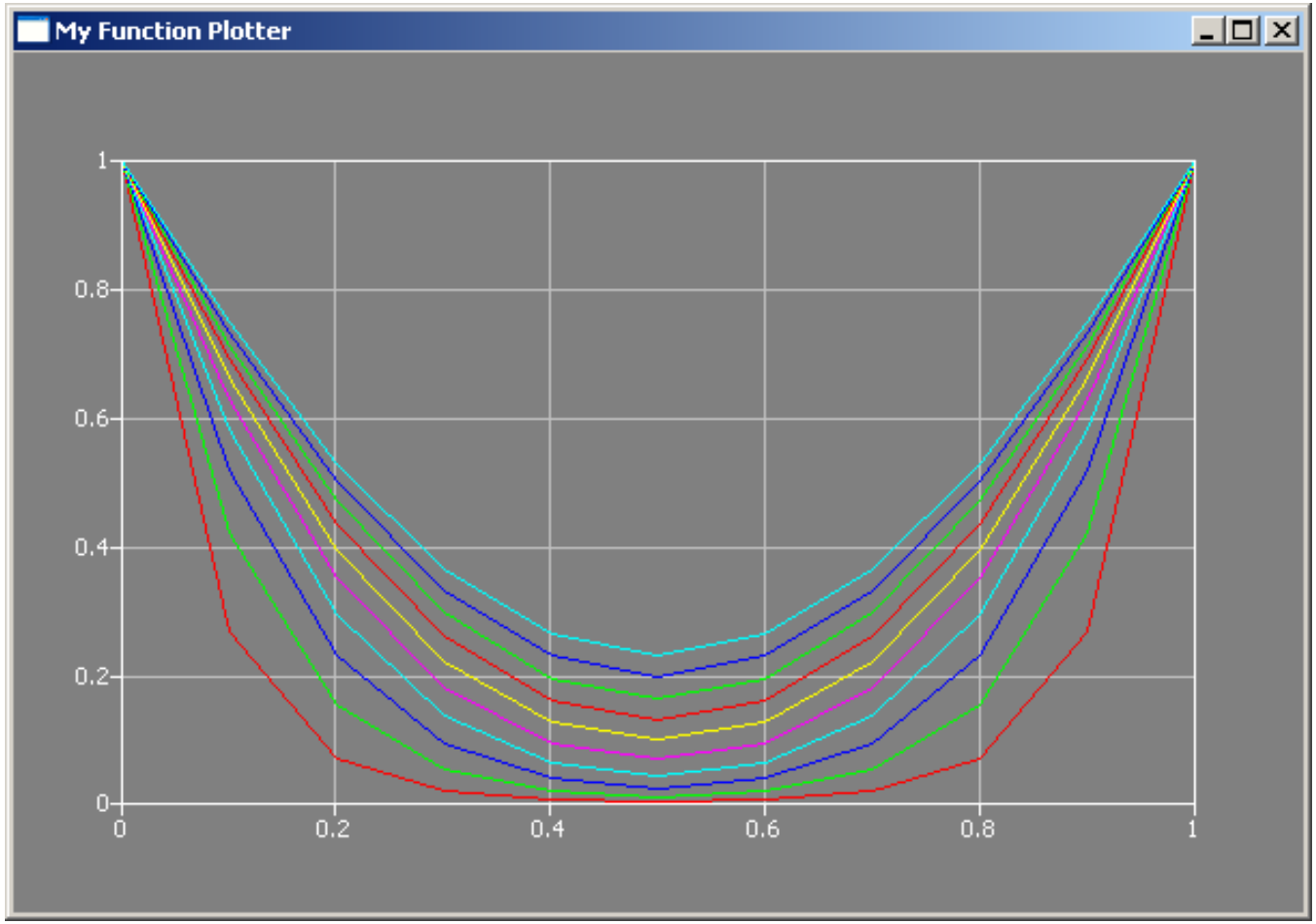
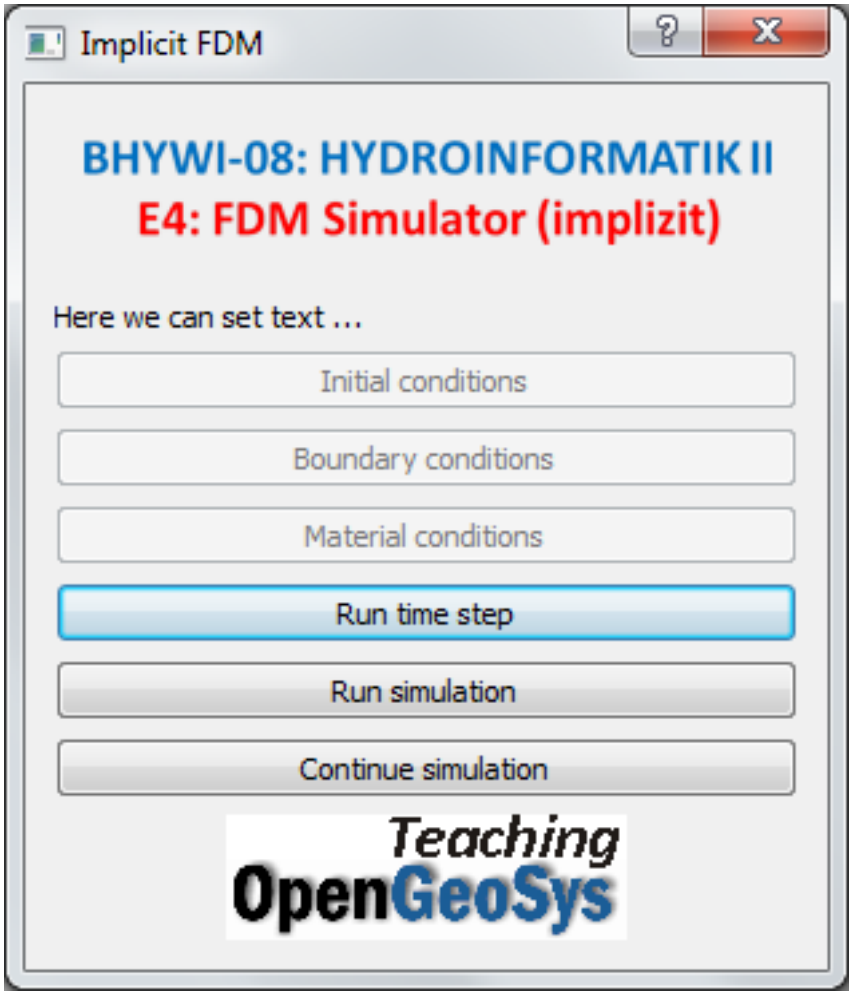


Figure: Zeitliche Entwicklung des Diffusionsprofils - implizites Verfahren (Wahoo...)

TECHNISCHE UNIVERSITÄT DRESDEN

Implementation #6: Run multiple time steps



- Einzelne Zeitschritte
- Mehrere Zeitschritte
- Weiterführen der Berechnung

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

Python Version

https://github.com/OlafKolditz/HYDROINFORMATIK-II