Hydroinformatik II - SoSe 2024

HyBHW-S2-01-V11: Finite-Differenzen-Methode II

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, 02.07.2022

TECHNISCHE UNIVERSITÄT DRESDEN

Zeitplan: Hydroinformatik II - SoSe 2024

Datum	HI	II	Thema	Typ
14.06.2024	14	2-01	Einführung in die Lehrveranstaltung - Teil 2	L
14.06.2024	15	2-02	Werkzeuge | Tools	L
14.06.2024	16	2-03	Grundlagen: Kontinuumsmechanik	L
21.06.2024	17	2-04	Grundlagen: Hydromechanik	L
21.06.2024	18	2-05	Grundlagen: Partielle Partialgleichungen	L
21.06.2024	19	2-06	Übung: Analytische Lösungen	E
28.06.2024*	20	2-07	Grundlagen: Näherungsverfahren	L
28.06.2024*	21	2-08	Übung: Jupyter Diffusionsprozess	E
02.07.2024*	22	2-09	Numerik: Finite-Differenzen-Methode (explizit)	L
02.07.2024*	23	2-10	Numerik: Finite-Differenzen-Methode (implizit)	L
12.07.2024	24	2-11	Übung: Finite-Differenzen-Methoden	E
12.07.2024	25	2-12	Grundlagen: Gerinnehydraulik	L
12.07.2024	26	2-13	Übung: Gerinnehydraulik	E
19.07.2024	27	2-14	Ausblick: Grundwassermodellierung	E
19.07.2024	28	2-15	Klausur/Beleg: Besprechung zur Vorbereitung	L

*online Vorlesung

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

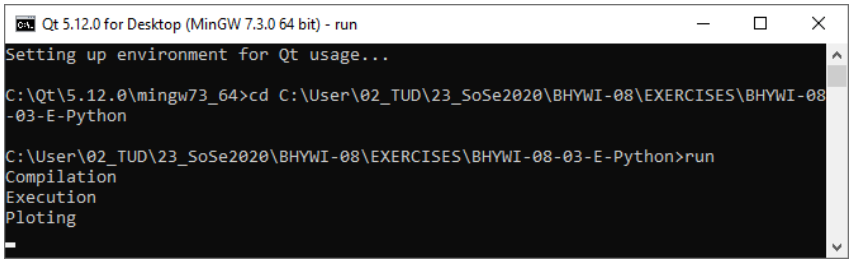
---

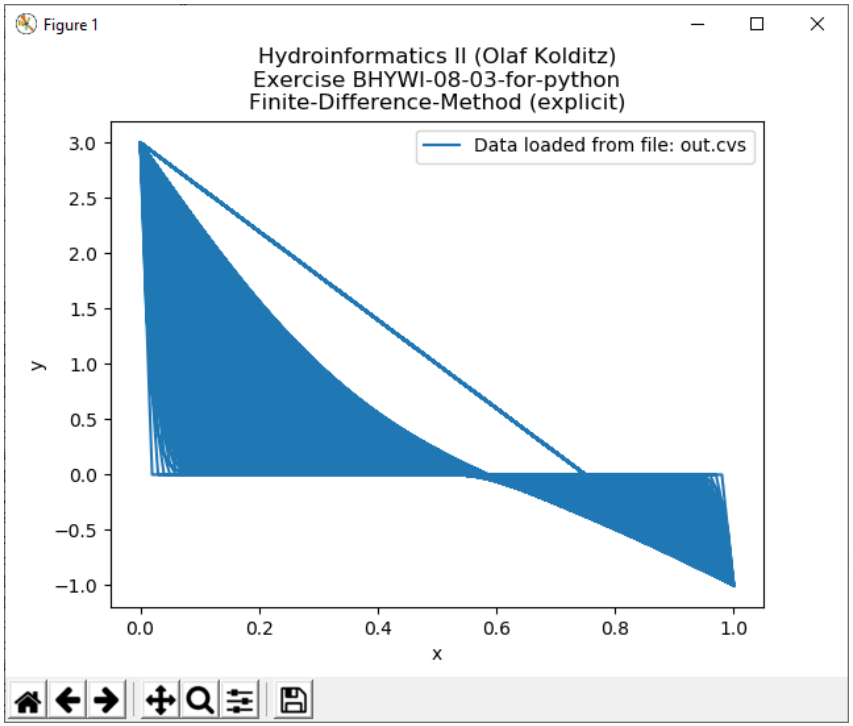
4. EX09-fdm-implicit: Übung implizite FDM
5. BHYWI-08-04E: Qt Version
6. (Tabelle Vorlesungen/Übungen/Hausaufgaben > neue Nummerierung)

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

TECHNISCHE UNIVERSITÄT DRESDEN

Letzte Vorlesung: explizite FDM mit Python





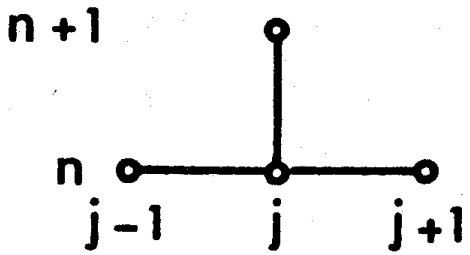
- Qt Installation (C++ Compiler und Konsole)
- Python Installation (weiter Pakete laden, `mathplotlib`)

BHYWI-08-03HW2 (Download von Webseite)

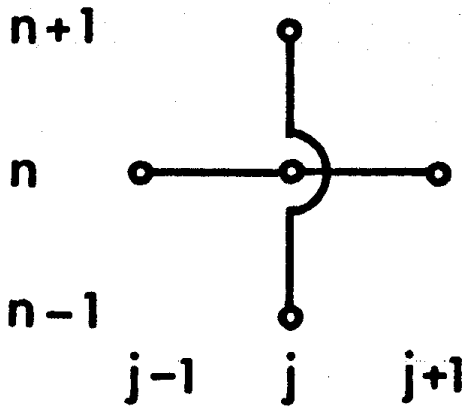
- Namen/Matrikelnummer in den Plot eintragen
- Nur jede 10. Kurve plotten

TECHNISCHE UNIVERSITÄT DRESDEN

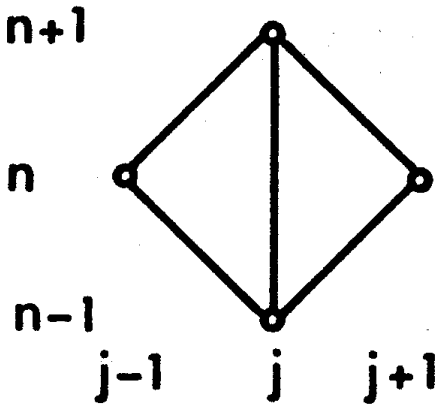
Letzte Vorlesung: explizite FDM mit Qt



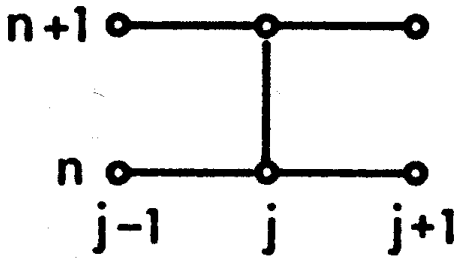
FTCS



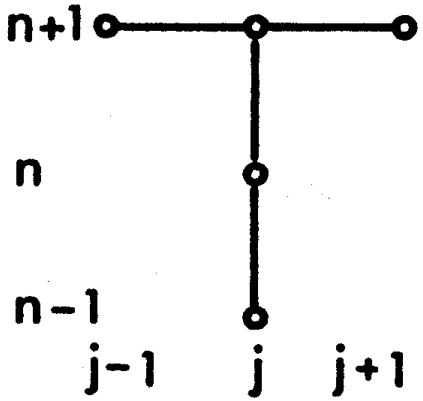
Richardson



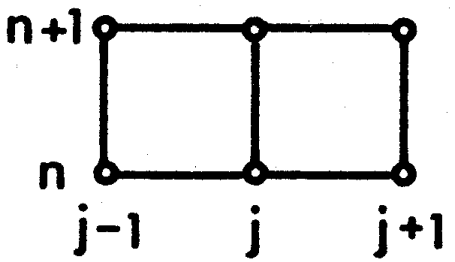
Du Fort-Frankel



Crank-Nicolson

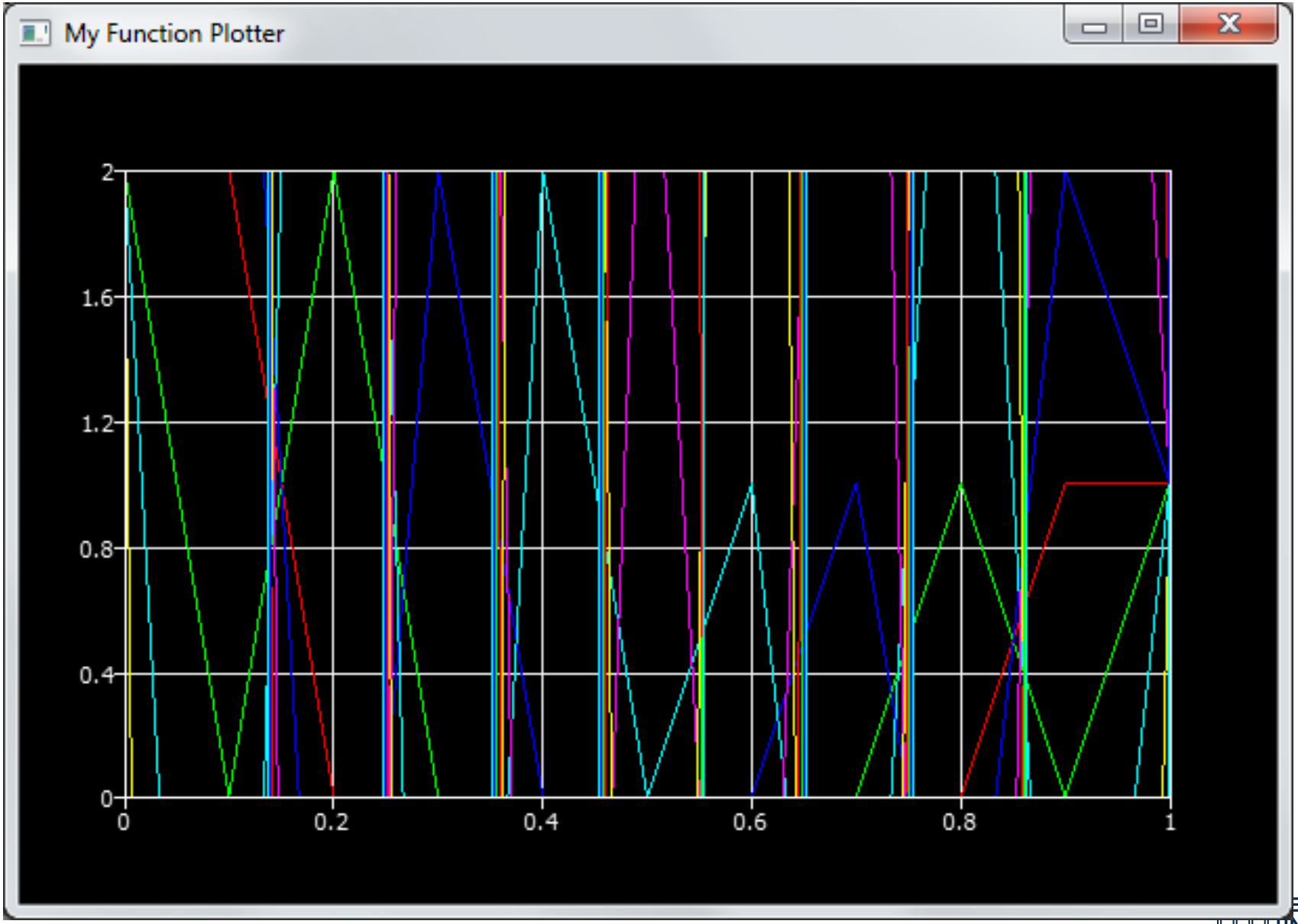


3LFI



Linear F.E.M./
Crank-Nicolson

Letzte Vorlesung: explizite FDM - Zeitschrittbegrenzung

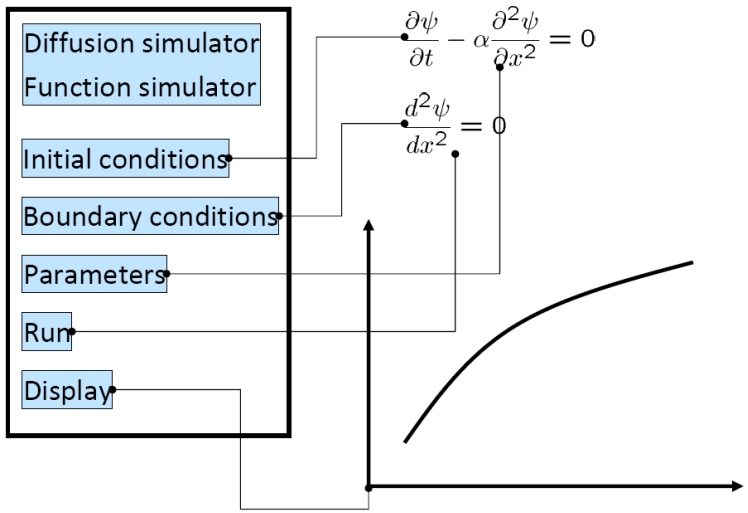


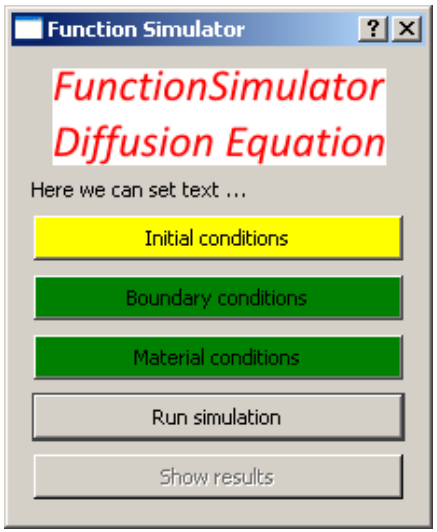
Letzte Vorlesung: explizite FDM - Zeitschrittbegrenzung

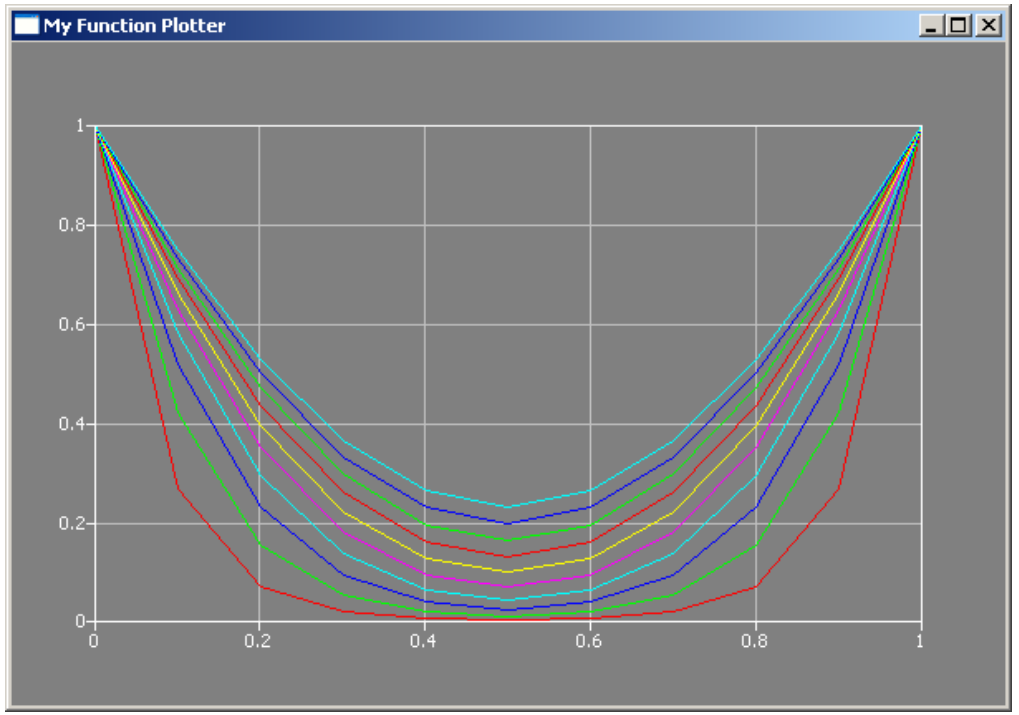
$$Ne = \alpha \frac{\Delta t}{\Delta x^2} \leqslant 0.5 \tag{1}$$

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

Ziel der Vorlesung







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

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

- (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)

$$\mathbf{Ax} = \mathbf{b} \tag{9}$$

- Explain steps with black board

TECHNISCHE UNIVERSITÄT DRESDEN

Implicite FDM - Theory #3 (Skript 4.2)

- Algebraic equation (matrix notation)

$$\mathbf{Ax} = \mathbf{b} \tag{10}$$

- Algebraic equation (index notation)

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

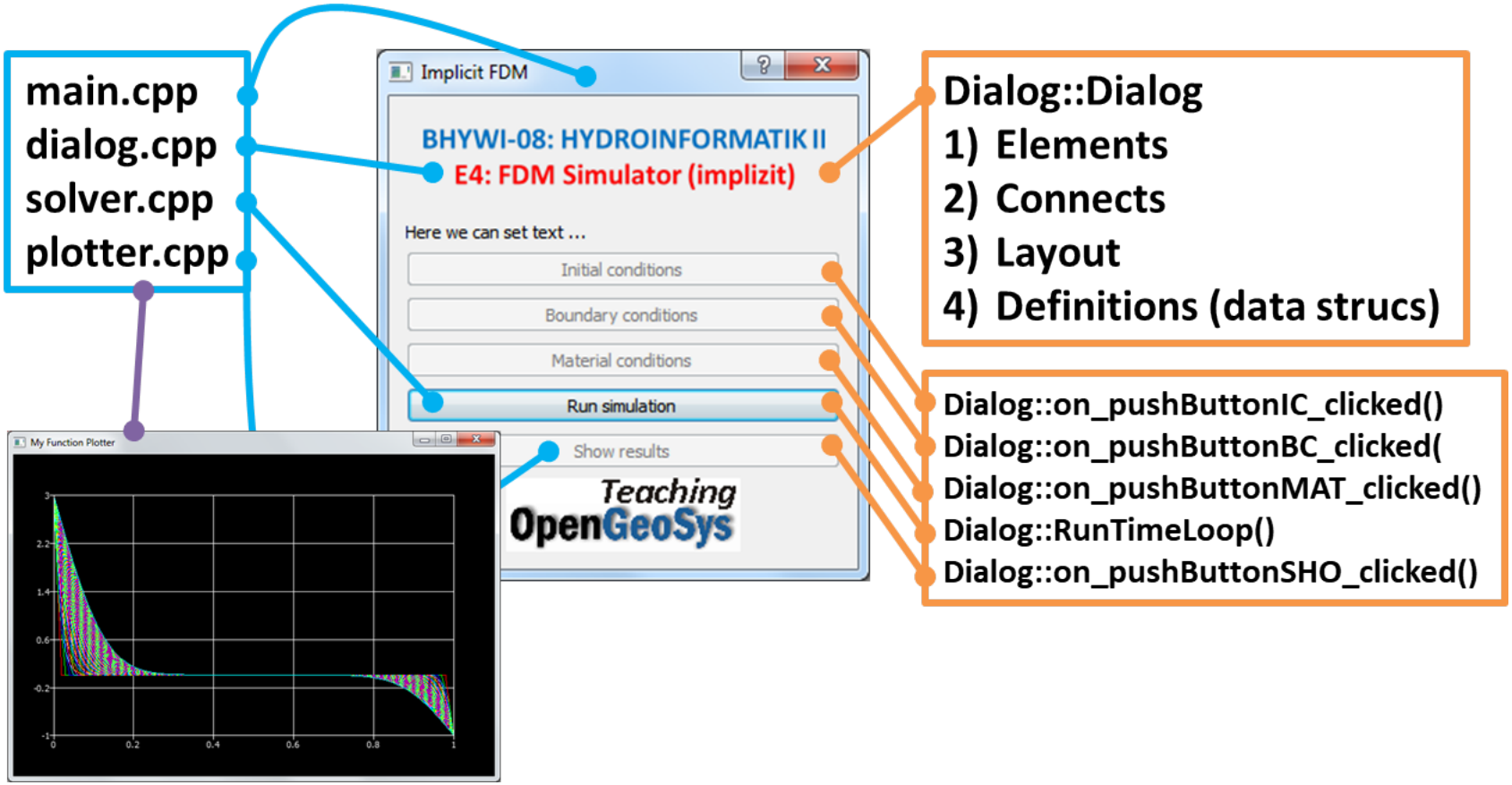
- Let's take a closer look ...

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

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;
}
```

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;
    }
  }
...}
```

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

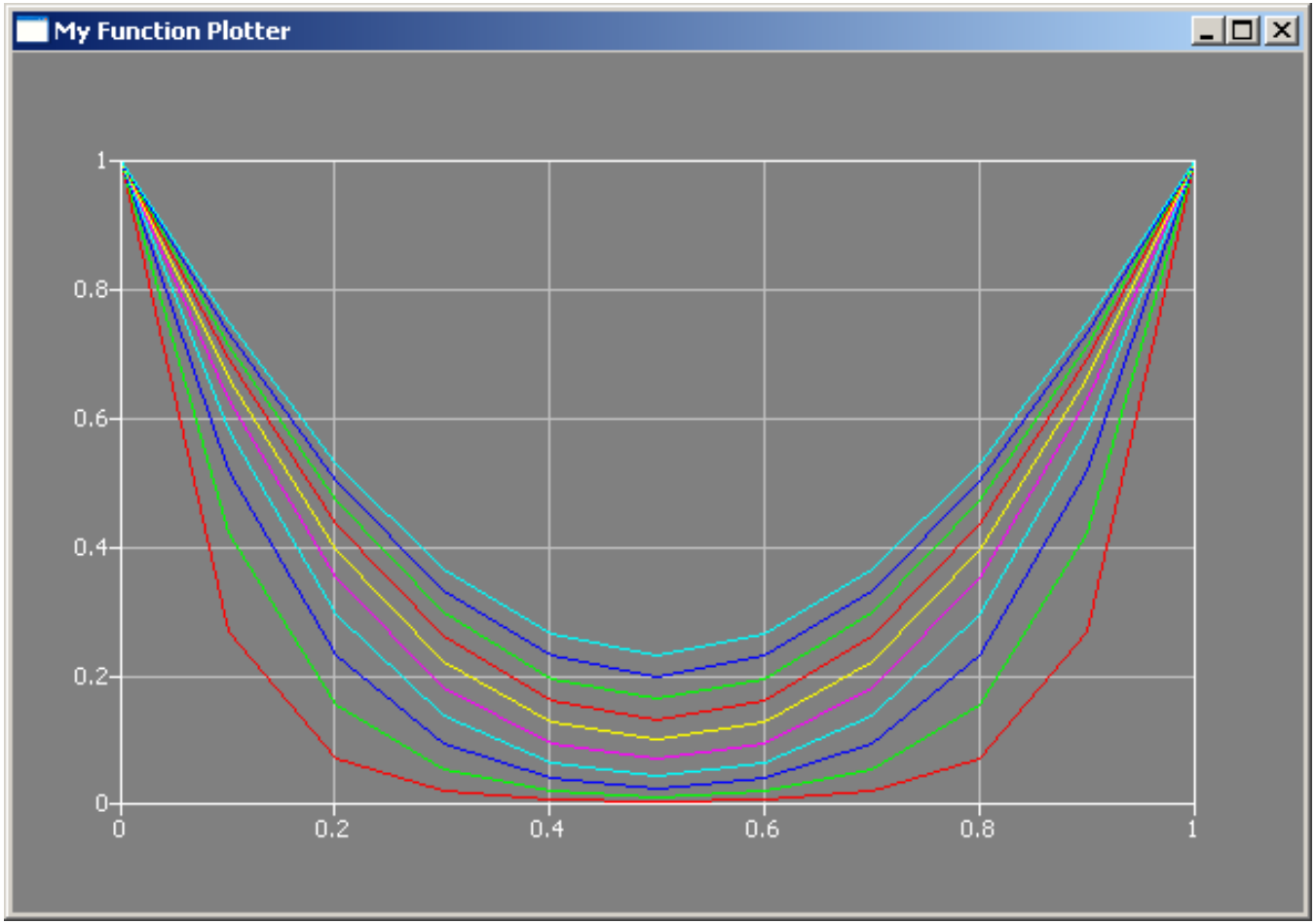
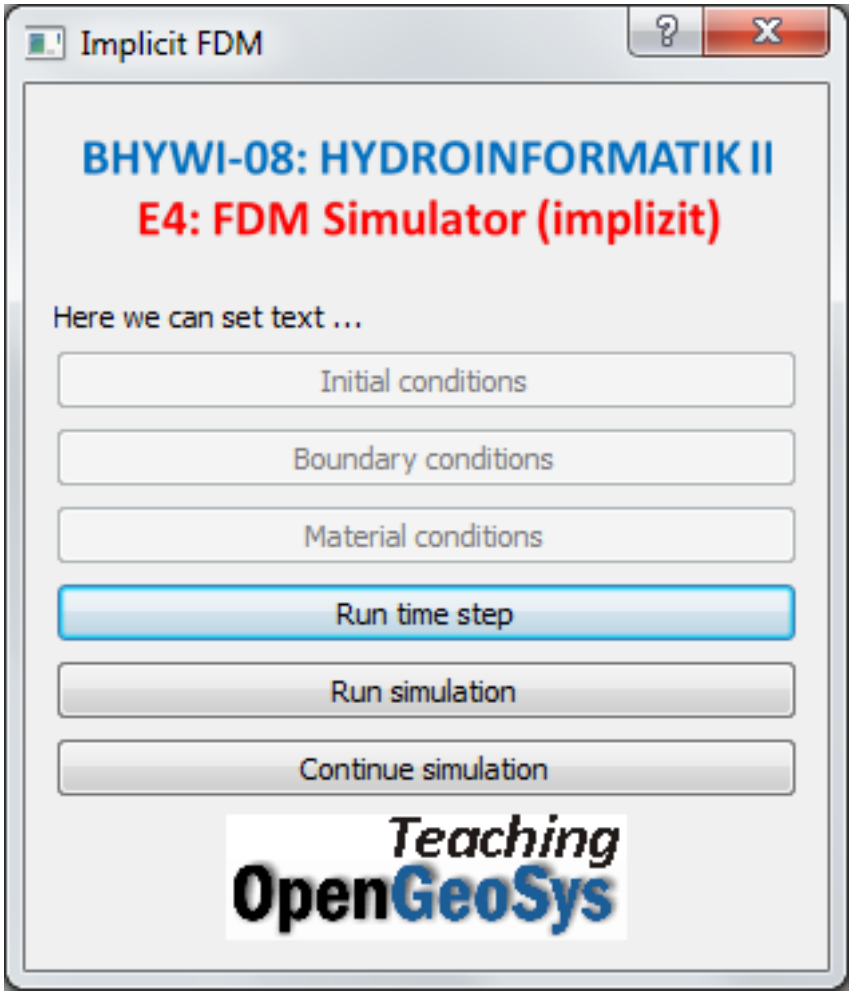


Fig.: Zeitliche Entwicklung des Diffusionsprofils - implizites Verfahren (Wahoo...)

Implementation #6: Run multiple time steps



- Einzelne Zeitschritte
- Mehrere Zeitschritte
- Weiterführen der Berechnung

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

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++

```
1  #include <vector>
2  #include <fstream>
3  extern void Gauss(double *matrix, double *vecb, double *
       vecx, int g);
4  void AssembleEquationSystem();
5  int n = 101;
6  double* matrix;
7  double* vecb;
8  double* vecx;
9  double Ne;
10 vector<double> u_new, u_old;
11 std::ofstream out_file;
12 int main(int argc, char *argv[])
13 {
14 ...
15 }
```

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

TECHNISCHE UNIVERSITÄT DRESDEN

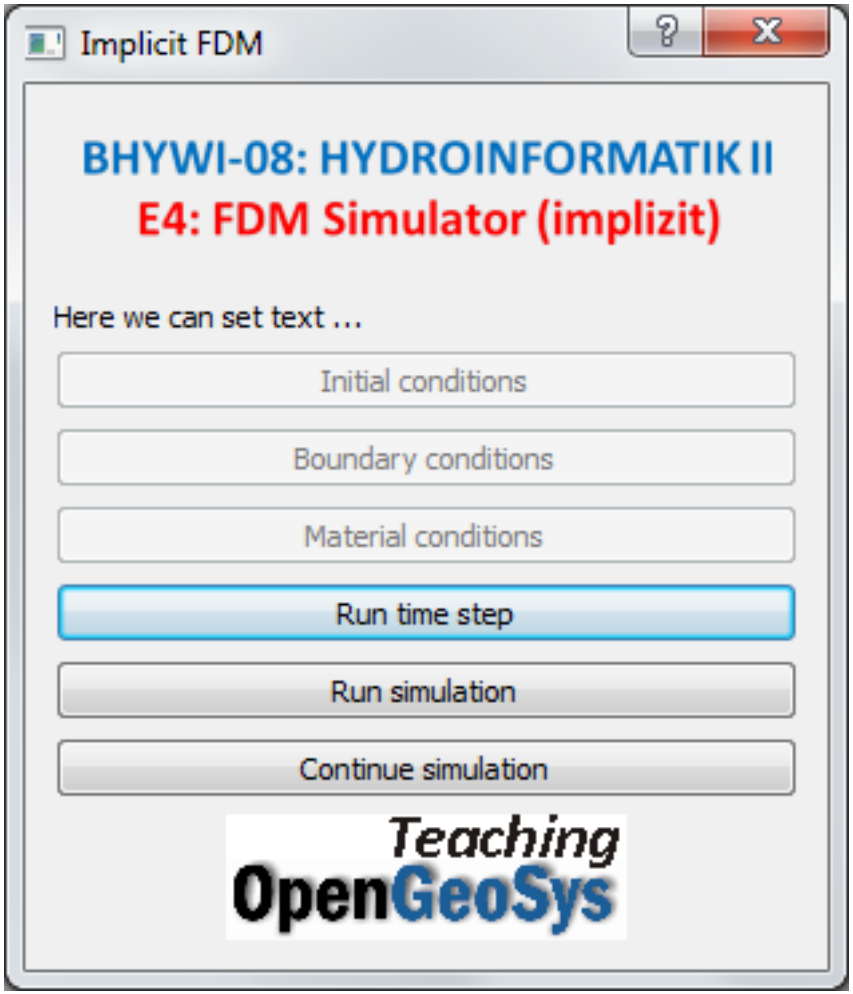
EX09-fdm-implicit-C++

```
1  int main(int argc, char *argv[])
2  {
3  //1-Definitionen
4    //1.1 Diskretisierung
5    //1.2 Loesungsvektoren
6    //1.3 Kennzahlen, Parameter
7    //1.4 Ausgabe
8  //2-Anfangsbedingungen
9  //3-Randbedingungen
10 //4-Diskretisierung
11 //6-Berechnung: FDM Verfahren
12   //AssembleEquationSystem();
13   //6.1 Matrizen berechnen
14   //6.2 Randbedingungen einbauen
15   //6.3 Gleichungsloeser: Gauss(matrix,vecb,vecx,nj);
16   //6.4 Ergebnisse speichern
17 }
```

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

TECHNISCHE UNIVERSITÄT DRESDEN

Qt version (interactivity): Run multiple time steps



99% Wiederverwendung des C++ codes

Vorteil der Qt Version ist die Interaktion:

- Einzelne Zeitschritte
- Mehrere Zeitschritte
- Weiterführen der Berechnung

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

EX09-fdm-implicit-C++