Modellierung von Hydrosystemen

"Numerische und daten-basierte Methoden"

BHYWI-22-04 @ 2020

Finite-Differenzen-Methode

Olaf Kolditz

*Helmholtz Centre for Environmental Research – UFZ

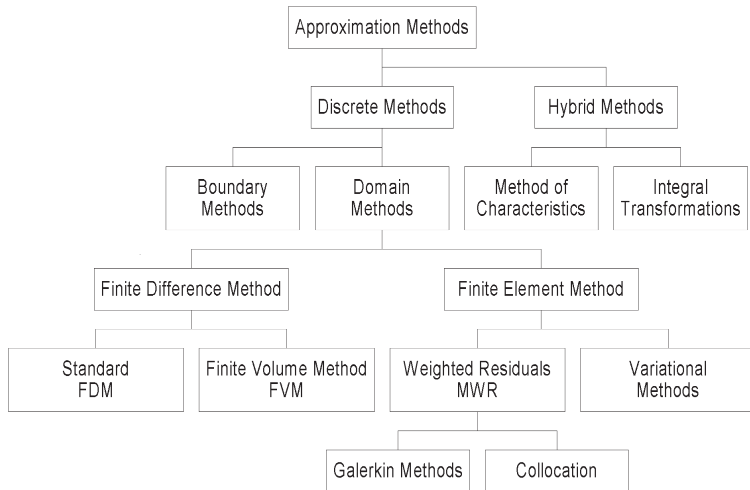
[1]Technische Universität Dresden – TUDD

[2]Centre for Advanced Water Research – CAWR

19.06.2020 - Dresden

TECHNISCHE UNIVERSITÄT DRESDEN

Näherungsverfahren



ECHNISCHE
NIVERSITÄT
RESDEN

Fahrplan für heute ...

Bisher

- Wiederholung Hydromechanik: Grundwasserströmungsgleichung
- Übung Einzugsgebiet: Berechnungsverfahren

Heute: Finite-Differenzen-Verfahren

- Grundlagen - GWE
- Grundlagen - TSE
- Übungen zur expliziten FDM (page 12) [BHYWI-22-E2]

Übungen: Werkzeuge

- PDE

$$S\frac{\partial h}{\partial t} - \frac{\partial}{\partial x}\left(K_x \frac{\partial h}{\partial x}\right) - \frac{\partial}{\partial y}\left(K_y \frac{\partial h}{\partial y}\right) = Q \qquad (1)$$

TECHNISCHE UNIVERSITÄT DRESDEN

in time ($\Delta t = t^{n+1} - t^n$)

$$u_j^{n+1} = \sum_{m=0}^{\infty} \frac{\Delta t^m}{m!} \left[\frac{\partial^m u}{\partial t^m} \right]_j^n \tag{2}$$

in space ($\Delta x = x_{i+1}^n - x_i^n$)

$$u_{i+1}^n = \sum_{m=0}^{\infty} \frac{\Delta x^m}{m!} \left[\frac{\partial^m u}{\partial x^m} \right]_i^n \tag{3}$$

in space ($\Delta y = y_{j+1}^n - y_j^n$)

$$u_{j+1}^n = \sum_{m=0}^{\infty} \frac{\Delta y^m}{m!} \left[\frac{\partial^m u}{\partial y^m} \right]_j^n \tag{4}$$

Grundlagen - FDM

- Zeitableitung

$$\left[\frac{\partial u}{\partial t}\right]_j^n = \frac{u_j^{n+1} - u_j^n}{\Delta t} - \frac{\Delta t}{2}\left[\frac{\partial^2 u}{\partial t^2}\right]_j^n - 0(\Delta t^2) \tag{5}$$

- in space

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_{i,j}^n = \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} - \frac{\Delta x^2}{12}\left[\frac{\partial^4 u}{\partial x^4}\right]_{i,j}^n - \dots \tag{6}$$

$$\left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j}^n = \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2} - \frac{\Delta y^2}{12}\left[\frac{\partial^4 u}{\partial y^4}\right]_{i,j}^n - \dots \tag{7}$$

TECHNISCHE UNIVERSITÄT DRESDEN

FDM - explizites Schema

$$
\begin{aligned}
& S_{i,j} \frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} \\
-K_{i,j}^{x} \frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{\Delta x^2} \quad - \quad & K_{i,j}^{y} \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{\Delta y^2} = Q_{i,j}
\end{aligned} \tag{8}
$$

$$
\begin{aligned}
u_{i,j}^{n+1} &= u_{i,j}^{n} \\
&+ \frac{K_{i,j}^{x}}{S_{i,j}} \frac{\Delta t}{\Delta x^2} u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n} \\
&+ \frac{K_{i,j}^{y}}{S_{i,j}} \frac{\Delta t}{\Delta y^2} u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n} \\
&+ \frac{Q_{i,j}}{S_{i,j}}
\end{aligned} \tag{9}
$$

TECHNISCHE UNIVERSITÄT DRESDEN

Grundlagen - FDM

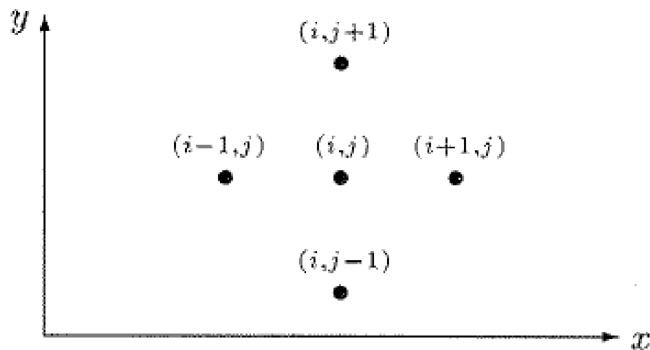


Figure: 5-Punkte-Stern (Knabner und Angermann 2000)

- Datenstrukturen

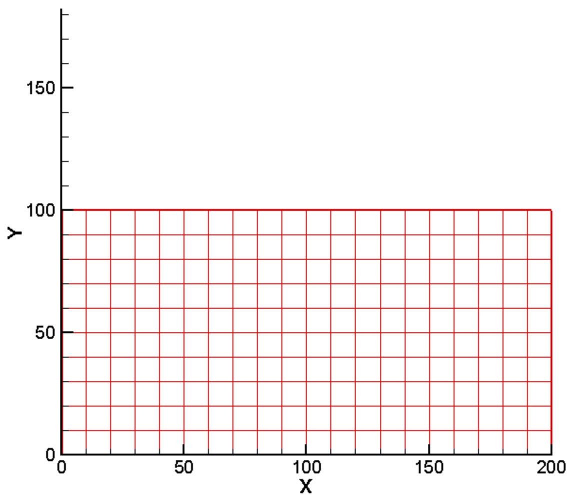
Table:

Feldgröße	u
Physikalische Parameter	S, K, Q
Numerische Parameter	$\Delta t, \Delta x, \Delta y$

Die Minimal-Datenstrukturen für die Programmierung der Gleichung (**??**) sind damit:

```
std::vector<float>u_new;
std::vector<float>u_old;
float S0,Kf,Q;
float dx,dy,dt;
```

Übung - BHYWI-22-E2 - QAD



Um dieses Gitter "abtasten" zu können, schreiben wir folgende doppelte Schleife.

```
for(j=0;j<jy;j++)
{
  nn = j*ix;
  for( i=0;i<ix;i++)
  {
    n = nn+i;
    u_new[n] = u[n] \
             + Kf/S0*dt/dx2 * (u[n+1]-2*u[n]+u[n-1]) \
             + Kf/S0*dt/dy2 * (u[(j+1)*ix+i]-2*u[n]+u[(j-1)*ix+i]) \
             + Q/S0;
  }
}
```

TECHNISCHE UNIVERSITÄT DRESDEN

Um dieses Gitter "abtasten" zu können, schreiben wir folgende doppelte Schleife.

```
for(int j=0;j<jy;j++)
{
  nn = j*ix;
  for(int i=0;i<ix;i++)
  {
    n = nn+i;
    if(IsBCNode(n,bc_nodes))
      continue;
    u_new[n] = u[n] \
             + Kf/S0*dt/dx2 * (u[n+1]-2*u[n]+u[n-1]) \
             + Kf/S0*dt/dy2 * (u[(j+1)*ix+i]-2*u[n]+u[(j-1)*ix+i]) \
             + Q/S0;
  }
}
```

TECHNISCHE UNIVERSITÄT DRESDEN

Dabei ist j der Laufindex über die y Richtung und i der Laufindex über die x Richtung. Ganz wichtig ist natürlich, den Speicher für die Vektoren bereitzustellen, bevor es los geht.

```
u.resize(ix*jy);
u_new.resize(ix*jy);
```

- Welche Rolle spielen `ix` und `jy` bei der Speicherreservierung?

TECHNISCHE UNIVERSITÄT DRESDEN

Übung - BHYWI-22-E2 - QAD

Natürlich müssen auch die Parameter vor der Berechnung initialisiert werden

```
ix = 21;
jy = 11;
dx = 10.;
dy = 10.;
dt = 0.25e2;
S0 = 1e-5;
Kf = 1e-5;
Q  = 0.;
u0 = 0.;
```

- Welche Einheiten haben die einzelnen Parameter?

TECHNISCHE UNIVERSITÄT DRESDEN

Das mit den Anfangsbedingungen ist eine einfache Sache. Mit der Doppelschleife über alle Knoten, können wir sehr einfach einen Wert u0 als Anfangsbedingung überall zuweisen.

```
for(int i=0;i<ix;i++)
  for(int j=0;j<jy;j++)
  {
    u[j*(ix+1)] = u0;
    u_new[j*(ix+1)] = u0;
  }
}
```

TECHNISCHE UNIVERSITÄT DRESDEN

Mit den Randbedingungen ist es etwas kniffliger ...

```
//top and bottom
int l;
for(int i=0;i<ix;i++)
{
  bc_nodes.push_back(i); u[i] = u_top  u_new[i] = u_top;
  l = ix*(jy-1)+i;
  bc_nodes.push_back(l); u[l] = u_bottom;  u_new[l] = u_bottom;
}
//left and right side
for(int j=1;j<jy-1;j++)
{
  l = ix*j;
  bc_nodes.push_back(l); u[l] = u_left;  u_new[l] = u_left;
  l = ix*j+ix-1;
  bc_nodes.push_back(l); u[l] = u_right;  u_new[l] = u_right;
}
```

TECHNISCHE UNIVERSITÄT DRESDEN

Übung - BHYWI-22-E2 - QAD

Sie sehen, dass wir für die Zuweisung der Randbedingungen eine neue Datenstruktur eingeführt haben.

```
std::vector<float>u_bc;
```

TECHNISCHE UNIVERSITÄT DRESDEN

Übung - BHYWI-22-E2 - QAD

Das Einbauen der Randbedingungen integrieren wir direkt in die Doppelschleife zur Berechnung der Knotenwerte. Dabei kommt eine neue Funktion `IsBCNode` ins Spiel, die wir uns gleich noch näher anschauen. `IsBCNode` soll eigentlich nichts anderes machen, als beim Auftreten einer Randbedingung nichts zu tun (i.e. `continue`). Randbedingungswerte sind gesetzt, müssen also nicht gerechnet werden.

```
for(int j=0;j<jy;j++)
{
  nn = j*ix;
  for(int i=0;i<ix;i++)
  {
    n = nn+i;
    if(IsBCNode(n,bc_nodes))
      continue;
    ...
  }
}
```

TECHNISCHE UNIVERSITÄT DRESDEN

Wie funktioniert nun `IsBCNode`?

```
bool IsBCNode(int n,std::vector<int>bc_nodes)
{
  bool is_node_bc = false;
  for(int k=0;k<(size_t)bc_nodes.size();k++)
  {
    if(n==bc_nodes[k])
    {
      is_node_bc = true;
      return is_node_bc;
    }
  }
  return is_node_bc;
}
```

Struktur der Funktion:

- Rückgabewert: logischer Wert wahr oder falsch
- Parameter: aktueller Gitterpunkt und Randbedingungsknotenvektor

Die Funktion überprüft, ob der Gitterpunkt n ein Randbedingungsknoten ist und gibt den entsprechenden logischen Wert zurück.

Das Ergebnis der finite Differenzen Simulation sehen wir in der Abb.

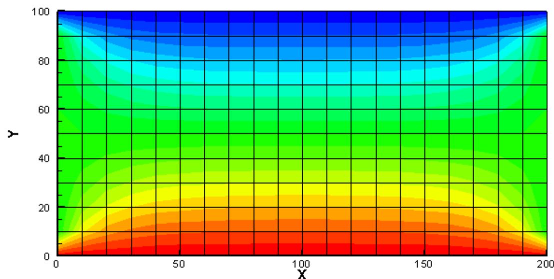


Figure: Berechnete Druckverteilung im Rechteck-Aquifer nach 100 Zeitschritten $\Delta t = 25$ sec

Übung - BHYWI-22-E2 - QAD

Jetzt werden wir mutig und vergrößern mal den Zeitschritt, sagen wir mal verdoppeln: $\Delta t = 50$ sec. Das Maleur sehen wir in der Abb. Was ist hier los?

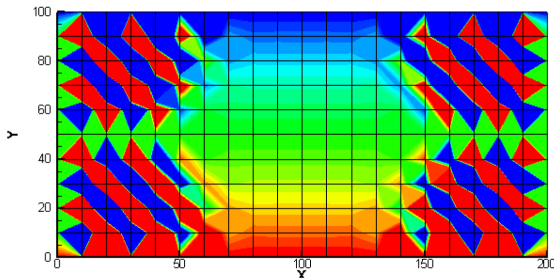


Figure: Berechnete Druckverteilung im Rechteck-Aquifer nach 100 Zeitschritten $\Delta t = 50$ sec

Übung - BHYWI-22-E2 - QAD

Wir erinnern uns noch dunkel daran, dass der Preis für das einfache explizite FDM ein strenges Stabilitätskriterium war (siehe Hydroinformatik, Teil II, Abschn. 3.2.2 und Abschn. 4.1). Dabei muss die Neumann-Zahl kleiner einhalb sein.

$$Ne = \alpha \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2} \tag{10}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Übung - BHYWI-22-E2 - QAD

Prima, aber was ist jetzt α und warum steht nur Δx und nicht auch Δy in der Gleichung? Zur bestimmung des α schreiben wir die Grundwassergleichung in eine Diffusionsgleichung wie folgt um.

$$\frac{\partial h}{\partial t} = \frac{K_x}{S}\frac{\partial^2 h}{\partial x^2} + \frac{K_y}{S}\frac{\partial^2 h}{\partial y^2} + \frac{Q}{S} \tag{11}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Wir sehen, dass es eigentlich zwei α-s gibt, für jede Richtung eins.

$$\alpha_x = \frac{K_x}{S}$$
$$\alpha_y = \frac{K_y}{S} \tag{12}$$

- Welche Einheit hat unser Grundwasser-α ?

Der richtige Zeitschritt für unser explizites FD Verfahren ergibt sich somit zu:

$$\Delta t \leq \frac{\min(\Delta x^2, \Delta y^2)}{2\alpha} \tag{13}$$

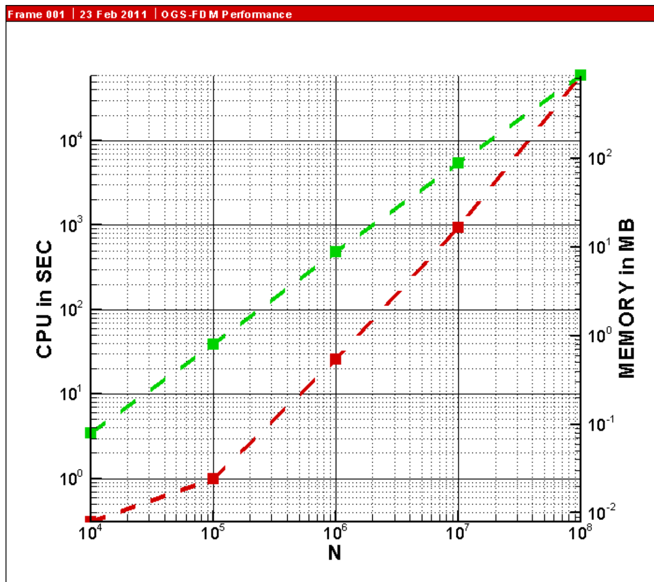
Übung - BHYWI-22-E2 - QAD

Die numerischen und hydraulischen Parameter sind in der nachfolgenden Tabelle **??** zu finden. Glücklicherweise sind die Ortdiskretisierungen und die hydraulischen Leitfähigkeiten in beide Koordinatenrichtungen gleich (isotropes Problem).

TECHNISCHE UNIVERSITÄT DRESDEN

$$\Delta t \leq \frac{100m^2}{2 \times 1m^2/s} = 50s \tag{14}$$

Übung - BHYWI-22-E2 - QAD



Übung - BHYWI-22-E2 - QAD

Wie stellen wir eine Zeitmessung in einem Programm an.

```
clock_t start, end; Definitionen
...
start = clock();    Beginn Zeitmessung
...
end = clock();      Ende Zeitmessung
...
time= (end-start)/(double)(CLOCKS_PER_SEC); Differenz
```

Übung E2 Der Quelltext für diese Übung befindet sich in `EXERCISES`.

TECHNISCHE UNIVERSITÄT DRESDEN

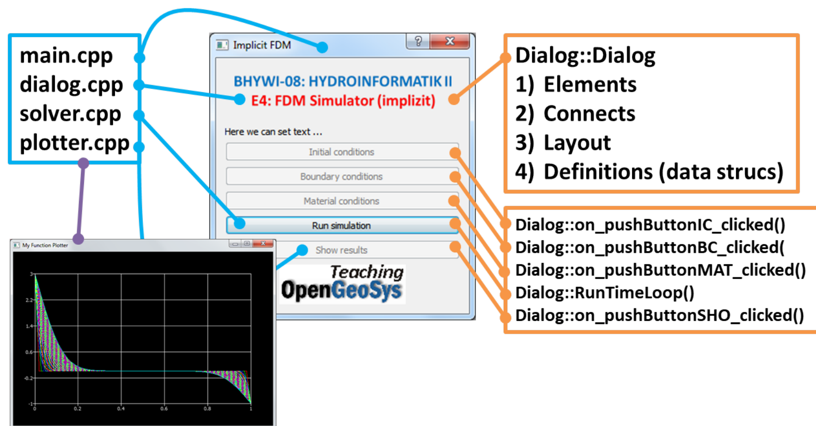
Übungen: Werkzeuge

TECHNISCHE UNIVERSITÄT DRESDEN

- Editor
- Compiler
- Workflows
- Ergebnisdarstellung

Übungen

Qt Überblick



TECHNISCHE UNIVERSITÄT DRESDEN

Modellierung von Hydrosystemen
"Numerische und daten-basierte Methoden"
BHYWI-22-V2-08 @ 2020
implizite Finite-Differenzen-Methode
Selke-Modell

Olaf Kolditz

*Helmholtz Centre for Environmental Research – UFZ

[1]Technische Universität Dresden – TUDD

[2]Centre for Advanced Water Research – CAWR

03.07.2020 - Dresden

TECHNISCHE UNIVERSITÄT DRESDEN

Fahrplan ...

- (void FDM::OutputResultsVTK(int t))
- implizite FDM (2D)
- Gleichungssysteme

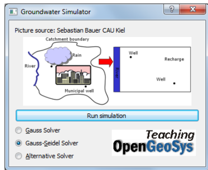
TECHNISCHE UNIVERSITÄT DRESDEN

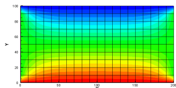
Konzept

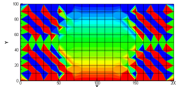
explizite FDM

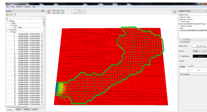
implizite FDM

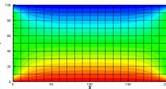
FEM

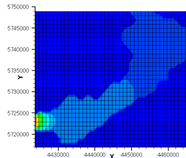




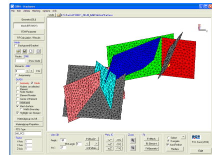


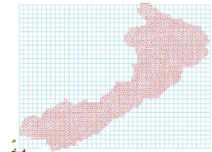






- ...





TECHNISCHE UNIVERSITÄT DRESDEN

Selke Einzugsgebiet

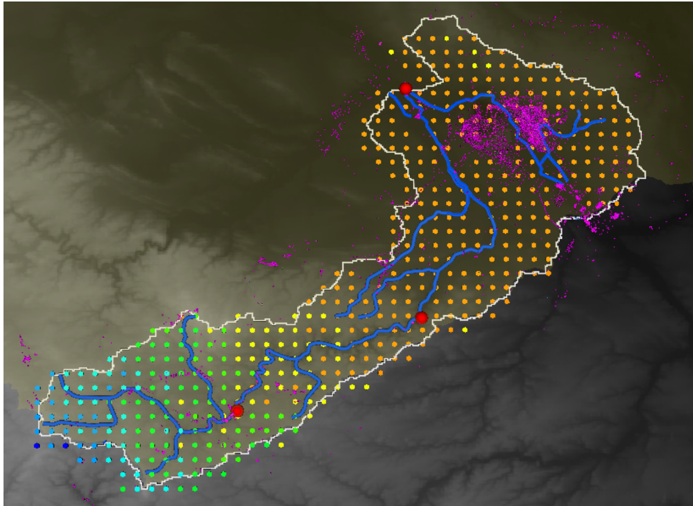
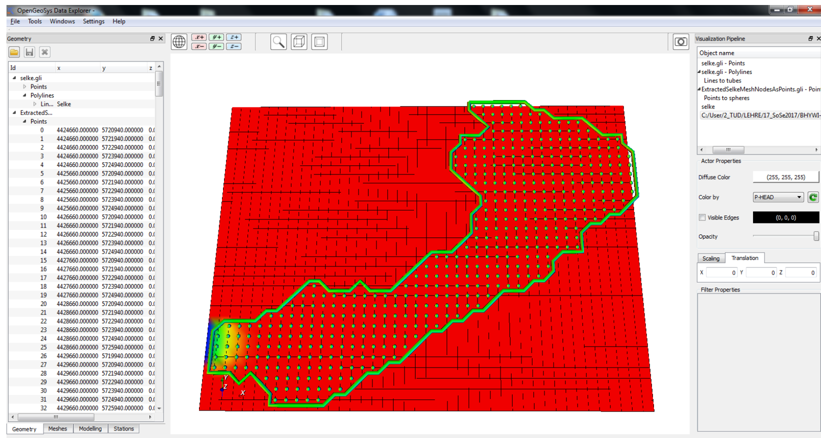


Figure: Untersuchungsgebiet - Selke

TECHNISCHE UNIVERSITÄT DRESDEN

Selke Einzugsgebiet



2D implizite FDM

siehe auch Abschn. 4.2 Hydroinformatik II

- Auswertung der Ableitungen zum neuen Zeitpunkt t^{n+1}

$$\left[\frac{\partial^2 h}{\partial x^2}\right]_{i,j}^{n+1} \approx \frac{h_{i-1,j}^{n+1} - 2h_{i,j}^{n+1} + h_{i+1,j}^{n+1}}{\Delta x^2} \quad (1)$$

$$\left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j}^{n+1} \approx \frac{h_{i,j-1}^{n+1} - 2h_{i,j}^{n+1} + h_{i,j+1}^{n+1}}{\Delta y^2} \quad (2)$$

TECHNISCHE UNIVERSITÄT DRESDEN

- Differenzen-Schema

$$S_{i,j}\frac{u_{i,j}^{n+1}-u_{i,j}^{n}}{\Delta t} \tag{3}$$
$$-K_{i,j}^{x}\frac{u_{i+1,j}^{n+1}-2u_{i,j}^{n+1}+u_{i-1,j}^{n+1}}{\Delta x^2} - K_{i,j}^{y}\frac{u_{i,j+1}^{n+1}-2u_{i,j}^{n+1}+u_{i,j-1}^{n+1}}{\Delta y^2} = Q_{i,j}$$

2D implizite FDM

- Gleichungssystem

$$
\begin{aligned}
& \left(\frac{S}{\Delta t} + 2\frac{K^x}{\Delta x^2} + 2\frac{K^y}{\Delta y^2}\right) u_{i,j}^{n+1} \\
- & \left(\frac{K^x}{\Delta x^2}\right)(u_{i-1,j}^{n+1} + u_{i+1,j}^{n+1}) - \left(\frac{K^y}{\Delta y^2}\right)(u_{i,j-1}^{n+1} + u_{i,j+1}^{n+1}) \\
= & \frac{S}{\Delta t} u_{i,j}^{n} + Q_{i,j} \qquad (4)
\end{aligned}
$$

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

Wir vereinfachen die Gleichung (4), indem wir für den Moment annehmen, dass $K^x = K^y = K$ (Isotropie) und $\Delta x = \Delta y = \Delta l$ (gleichförmige Diskretisierung). Die Multiplikation mit $\Delta t / S$ ergibt dann folgende Beziehung.

$$\begin{aligned} & \left(1 + 4\frac{K\Delta t}{S\Delta l^2}\right) u_{i,j}^{n+1} \\ - & \left(\frac{K\Delta t}{S\Delta l^2}\right)\left(u_{i-1,j}^{n+1} + u_{i+1,j}^{n+1} + u_{i,j-1}^{n+1} + u_{i,j+1}^{n+1}\right) \\ = & \; u_{i,j}^{n} + \frac{\Delta t}{S} Q_{i,j} \end{aligned} \tag{5}$$

K : Vergleichen Sie die Beziehung (5) mit der Gleichung (4.10, Skript Hydroinformatik II).

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

Der Ausdruck $K/S = \alpha$ entspricht dem Diffusivitätskoeffizienten (Überprüfen sie dies anhand der Einheiten). Damit ist die Neumann-Zahl

$$Ne = \frac{K}{S} \frac{\Delta t}{\Delta l^2} \tag{6}$$

Nun versuchen wir anhand der Gleichung (5) die Struktur des zu lösenden Gleichungssystems zu beschreiben. Wir gehen wieder ganz genau so vor wie bei der 1D Diffusionsgleichung im Abschn. 4.2 (Hydroinformatik II).

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM - Gleichnungssystem

$$\underbrace{\begin{bmatrix} 1+4Ne & -Ne & & & & & & \\ -Ne & \dots & \dots & & & & & \\ & \dots & \dots & \dots & & & & \\ & & -Ne & 1+4Ne & & & & \\ & & & & 1+4Ne & -Ne & & \\ & & & & & \dots & \dots & \dots \\ & & & & & & -Ne & 1+4Ne \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} u_{0,0}^{n+1} \\ u_{1,0}^{n+1} \\ \dots \\ u_{I-1,0}^{n+1} \\ u_{0,1}^{n+1} \\ \dots \\ u_{I-1,J-1}^{n+1} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} u_{0,0}^{n} + b_{0,0} \\ u_{1,0}^{n} + b_{1,0} \\ \dots \\ u_{I-1,0}^{n} + b_{I-1,0} \\ u_{0,1}^{n} + b_{0,1} \\ \dots \\ u_{I-1,J-1}^{n} + b_{I-1,J-1} \end{bmatrix}}_{\mathbf{b}} \qquad (7)$$

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

Auch was die Programmierung betrifft, können wir auf unsere Erfahrungen in Hydroinformatik II aufbauen. Es gibt praktisch keinen Unterschied, ob wir es mit einem 1D oder 2D Problem zu tun haben. Wir müssen lediglich aufpassen, dass wir die Indizes richtig zählen.

Wir benutzen die Grundstruktur des objekt-orientierten Programms für das explizite FD Verfahren (Abschn. **??**). Die wesentlichen Unterschiede der impliziten zur expliziten FDM sind, dass wir ein Gleichungssystem aufbauen und lösen müssen.

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM - die main function

```
#include <iostream>
#include "fdm.h"
#include <time.h>
extern void Gauss(double*,double*,double*,int);
int main(int argc, char *argv[])
{
  //----------------------------------------------
  FDM* fdm = new FDM();
  fdm->SetInitialConditions();
  fdm->SetBoundaryConditions();
  //----------------------------------------------
  int tn = 2;
  for(int t=0;t<tn;t++)
  {
    fdm->AssembleEquationSystem();
    Gauss(fdm->matrix,fdm->vecb,fdm->vecx,fdm->IJ);
    fdm->SaveTimeStep();
    fdm->OutputResults(t);
  }
  //----------------------------------------------
  fdm->out_file.close();
  return 0;
}
```

2D implizite FDM

Dennoch können wir erstaunlich viel wiederverwenden, bis auf

```
fdm->AssembleEquationSystem();
Gauss(fdm->matrix,fdm->vecb,fdm->vecx,fdm->IJ);
```

Der Gleichungslöser `Gauss` ist übrigens genau der gleiche, den wir schon für die Lösung des impliziten FD Verfahrens für die Diffusionsgleichung in Hydroinformatik II benutzt haben.

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

Der Reihe nach. Die Assemblierfunktion soll das Gleichungssystem (7) aufbauen. Vom Prinzip her das Gleiche wie beim 1D FD Verfahren:

- Die Hauptdiagonale bekommt den Wert $1 + 4Ne$,
- die Nebendiagonalen haben den Wert $-Ne$.

Dies lässt sich programmtechnisch recht einfach bewerkstelligen (sie erinnern sich, wie wir in einer Doppelschleife, die Hauptdiagonale herausfinden können)

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

```
void FDM::AssembleEquationSystem()
{
  // Matrix entries
  for(i=0;i<IJ;i++)
  {
    vecb[i] = u[i];
    for(j=0;j<IJ;j++)
    {
      matrix[i*IJ+j] = 0.0;
      if(i==j)
        matrix[i*IJ+j] = 1. + 4.*Ne;
      else if(abs((i-j))==1)
        matrix[i*IJ+j] = - Ne;
    }
  }
  // Incorporate boundary conditions
  IncorporateBoundaryConditions();
  // Matrix output
  WriteEquationSystem();
}
```

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

Um die Assemblierfunktion zu testen bauen wir uns ein ganz einfaches Beispiel bestehend aus nur 9 Knoten (Abb. 2) - je einfacher, desto besser.

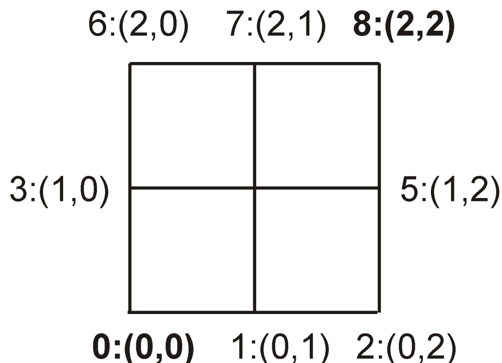


Figure: Testbeispiel

2D implizite FDM

Mit Hilfe der nützlichen Funktion `WriteEquationSystem()` können wir das Gleichungssystem in eine Datei schreiben, das Ergebnis passt.

```
2 -0.25 0 0 0 0 0 0 0
-0.25 2 -0.25 0 0 0 0 0 0
0 -0.25 2 -0.25 0 0 0 0 0
0 0 -0.25 2 -0.25 0 0 0 0
0 0 0 -0.25 2 -0.25 0 0 0
0 0 0 0 -0.25 2 -0.25 0 0
0 0 0 0 0 -0.25 2 -0.25 0
0 0 0 0 0 0 -0.25 2 -0.25
0 0 0 0 0 0 0 -0.25 2
```

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

Etwas kniffliger is es mit dem Einbauen der Randbedingungen. Wir erinnern uns, der Trick war eine Manipulation der Matrix und des RHS (right-hand-side) vectors, um den vorgegebenen Wert der Randbedingung zu erzwingen. Der Code zeigt das Beispiel für den Einbau einer Randbedingung im Knoten (also oben rechts in unseren kleinen Testbeispiel).

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

```
void FDM::IncorporateBoundaryConditions()
{
  size_t i_bc;
  int i_row, k;
  for(i_bc=0;i_bc<bc_nodes.size();i_bc++)
  {
    i_row = bc_nodes[i_bc];
    // Null off-diangonal entries of the related row and columns
    // Apply contribution to RHS by BC
    for(j=0;j<IJ;j++)
    {
      if(i_row == j)
        continue; // do not touch diagonals
      matrix[i_row*(IJ)+j] = 0.0; // NUll row
      k = j*(IJ)+i_row;
      // Apply contribution to RHS by BC
      vecb[j] -= matrix[k]*u[i_row];
      matrix[k] = 0.0; // Null column
    }
    // Apply Dirichlet BC
    vecb[i_row] = u[i_row]*matrix[i_row*(IJ)+i_row];
  }
}
```

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

Wir schreiben wieder das Gleichungssystem mit `WriteEquationSystem()` in eine Datei und schauen uns jeden Schritt genau an.

- Diagonalelemente werden nicht angefasst.
- Reihe zu Null setzen

```
2 0 0 0 0 0 0 0 0          b: 1
-0.25 2 -0.25 0 0 0 0 0 0 b: 0
0 -0.25 2 -0.25 0 0 0 0 0 b: 0
0 0 -0.25 2 -0.25 0 0 0 0 b: 0
0 0 0 -0.25 2 -0.25 0 0 0 b: 0
0 0 0 0 -0.25 2 -0.25 0 0 b: 0
0 0 0 0 0 -0.25 2 -0.25 0 b: 0
0 0 0 0 0 0 -0.25 2 -0.25 b: 0
0 0 0 0 0 0 0 0 2          b: -1
```

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

- Rechte Seite manipulieren

```
2 0 0 0 0 0 0 0 0         b: 1
-0.25 2 -0.25 0 0 0 0 0 0 b: 0.25
0 -0.25 2 -0.25 0 0 0 0 0 b: 0
0 0 -0.25 2 -0.25 0 0 0 0 b: 0
0 0 0 -0.25 2 -0.25 0 0 0 b: 0
0 0 0 0 -0.25 2 -0.25 0 0 b: 0
0 0 0 0 0 -0.25 2 -0.25 0 b: 0
0 0 0 0 0 0 -0.25 2 -0.25 b: -0.25
0 0 0 0 0 0 0 0 2         b: -1
```

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

- Spalte Null setzen

```
2 0 0 0 0 0 0 0 0            b: 1
0 2 -0.25 0 0 0 0 0 0        b: 0.25
0 -0.25 2 -0.25 0 0 0 0 0 b: 0
0 0 -0.25 2 -0.25 0 0 0 0 b: 0
0 0 0 -0.25 2 -0.25 0 0 0 b: 0
0 0 0 0 -0.25 2 -0.25 0 0 b: 0
0 0 0 0 0 -0.25 2 -0.25 0 b: 0
0 0 0 0 0 0 -0.25 2 0        b: -0.25
0 0 0 0 0 0 0 0 2            b: -1
```

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

- Neumann Randbedingungen setzen

```
2 0 0 0 0 0 0 0 0         b:2
0 2 -0.25 0 0 0 0 0 0     b:0.25
0 -0.25 2 -0.25 0 0 0 0 0 b:0
0 0 -0.25 2 -0.25 0 0 0 0 b:0
0 0 0 -0.25 2 -0.25 0 0 0 b:0
0 0 0 0 -0.25 2 -0.25 0 0 b:0
0 0 0 0 0 -0.25 2 -0.25 0 b:0
0 0 0 0 0 0 -0.25 2 0     b:-0.25
0 0 0 0 0 0 0 0 2         b:-2
```

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM

Schließlich ergibt sich folgendes Gleichungssystem zur Lösung durch das Gauss-Verfahren noch mal richtig aufgeschrieben.

$$\begin{aligned}
2h_1^{n+1} &= 2h_1^n \\
2h_2^{n+1} - 0.25h_3^{n+1} &= h_2^n + 0.25h_1^n \\
-0.25h_2 + 2h_3 - 0.5h_4 &= 0 \\
-0.25h_3 + 2h_4 - 0.5h_5 &= 0 \\
-0.25h_4 + 2h_5 - 0.5h_6 &= 0 \\
-0.25h_5 + 2h_6 - 0.5h_7 &= 0 \\
-0.25h_6 + 2h_7 - 0.5h_8 &= 0 \\
-0.25h_7 + 2h_8 &= -0.25 \\
2h_9^{n+1} &= h_9^n - 2
\end{aligned} \tag{8}$$

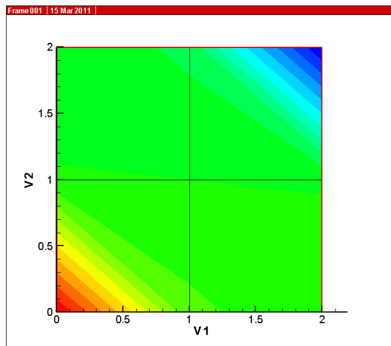
2D implizite FDM

Das geschriebene Ergebnisfile sieht dann folgendermaßen aus.

```
ZONE T="0.25", I=3, J=3, DATAPACKING=POINT
0 0 1
1 0 0.127016
2 0 0.016129
0 1 0.00201613
1 1 0
2 1 -0.00201613
0 2 -0.016129
1 2 -0.127016
2 2 -1
ZONE T="100.", I=3, J=3, DATAPACKING=POINT
0 0 1
1 0 0.267857
2 0 0.0714286
0 1 0.0178571
1 1 1.80718e-19
2 1 -0.0178571
0 2 -0.0714286
1 2 -0.267857
2 2 -1
```

TECHNISCHE UNIVERSITÄT DRESDEN

2D implizite FDM



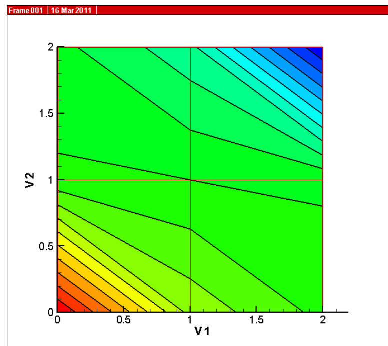


Figure: Ergebnisse des impliziten FD Verfahrens für t=0.25, 10 sec

Übung GW4 Der Quelltext für diese Übung befindet sich in `EXERCISES`.

TECHNISCHE UNIVERSITÄT DRESDEN

Modellierung von Hydrosystemen
"Numerische und daten-basierte Methoden"
BHYWI-22-V2-09 @ 2020
Finite-Elemente-Methode
Säulen-Modell

Olaf Kolditz

*Helmholtz Centre for Environmental Research – UFZ

[1]Technische Universität Dresden – TUDD

[2]Centre for Advanced Water Research – CAWR

10.07.2020 - Dresden

TECHNISCHE UNIVERSITÄT DRESDEN

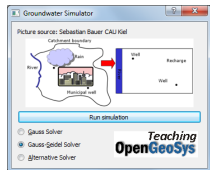
Lecture Table of Contents

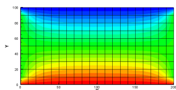
- Zusammenfassung FDM
- Einführung in die Finite-Elemente-Methode (am Beispiel einer Bodensäule)
- Implementierung
- Testbeispiel

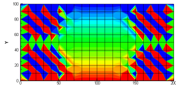
TECHNISCHE UNIVERSITÄT DRESDEN

Zusammenfassung

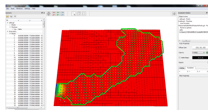
explizite FDM

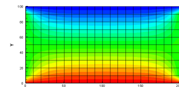


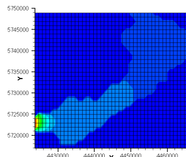




implizite FDM

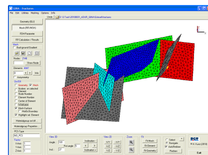


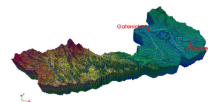




- Pro / Cons
- FDM: einfache Implementierung, starre Geometrien
- FEM: schwieriger zu implementieren (heute), flexible Geometrien

FEM





Finite-Elemente-Methode (FEM)

Wir haben uns sehr intensiv mit der Methode der finiten Differenzen beschäftigt. Bei der Einführung der numerischen Berechnungsmethoden in der Hydroinformatik II Veranstaltung haben wir gesehen, dass es ein ganzes Arsenal von Verfahren gibt (Abb. 2.1, Hydroinformatik II Skript), welche für bestimmte Problemstellungen geeignet oder ungeeignet sind. In den Visualisierungsübungen im VISLab werden wir sehen, dass FD Verfahren Grenzen haben, wenn es um die exakte Beschreibung komplexer Geometrien geht. Hier sind Verfahren im Vorteil, die sogenannte unstrukturierte Rechengitter benutzen können. Hierzu zählt z.B. die Finite Elemente Methode, mit der wir uns nun etwas näher beschäftigen möchten. Die Abb. 1 zeigt uns ein aktuelles Beispiel aus einem Forschungsvorhaben zusammen mit der Bundesanstalt für Geowissenschaften und Rohstoffe (BGR) in Hannover

TECHNISCHE UNIVERSITÄT DRESDEN

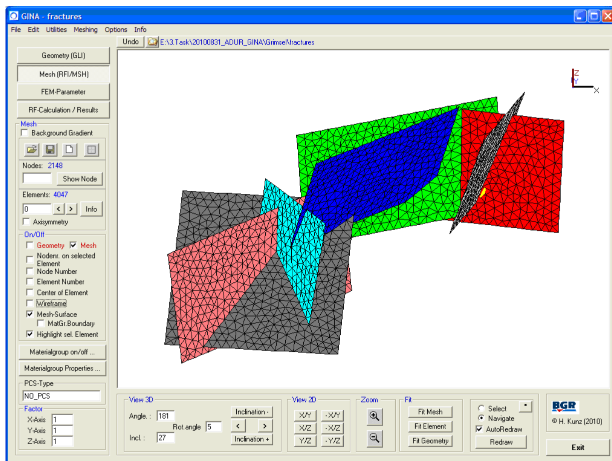


Figure: Modellierung eines Kluftsystems im Kristallin (Herbert Kunz, BGR)

Finite-Elemente-Methode (FEM) - Motivation

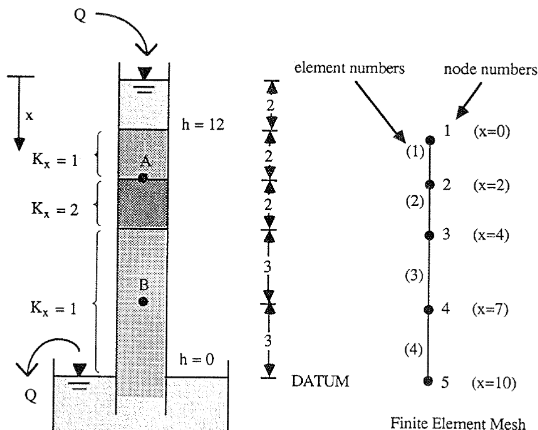


Figure: Bodensäulenmodell zur Erläuterung der FE Methode nach Istok (1989)

Finite-Elemente-Methode (FEM) - Konzept

Wir betrachten ein 1D stationäres Grundwasserströmungsproblem.

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial h}{\partial x}\right) = 0 \tag{1}$$

Ein Näherungsverfahren wird uns eine Näherungslösung $\hat{h}$ liefern, welche die Bilanzgleichung (1) nicht mehr ganz korrekt erfüllt.

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial \hat{h}}{\partial x}\right) = R(x) \neq 0 \tag{2}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Finite-Elemente-Methode (FEM)

Dabei ist $R(x)$ der Fehler, das sogenannte Residuum. Das Residuum kann in den verschiedenen Gitterpunkten i, j unterschiedliche Wert $R_i \neq R_j$ annehmen. Am Knoten 3 hängen die Elemente (2) und (3) (Abb. 2). Daher ergibt sich das Residuum im Knoten 3 aus den Elementwerten wie folgt.

$$R_3 = R_3^{(2)} + R_3^{(3)} \tag{3}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Finite-Elemente-Methode (FEM)

Ohne Beschränkung der Allgemeinheit können wir für jeden Knoten i, das Residuum wie folgt schreiben.

$$R_i = \sum_{e=1}^{p} R_i^{(e)} \tag{4}$$

Dabei ist p die Anzahl der Elemente, die am Knoten i angebunden sind.

TECHNISCHE UNIVERSITÄT DRESDEN

Finite-Elemente-Methode (FEM)

Der Elementbeitrag zum Residuum lässt sich wie folgt berechnen.

$$R_i^{(e)} = \int_{x_i^e}^{x_j^e} N_i^{(e)} \left(K_x^{(e)} \frac{\partial^2 \hat{h}^{(e)}}{\partial x^2} \right) dx \tag{5}$$

Dabei ist $N_i^{(e)} \equiv W_i(x)$ eine Interpolationsfunktion auf dem Element (e) (Abb. 3).

TECHNISCHE UNIVERSITÄT DRESDEN

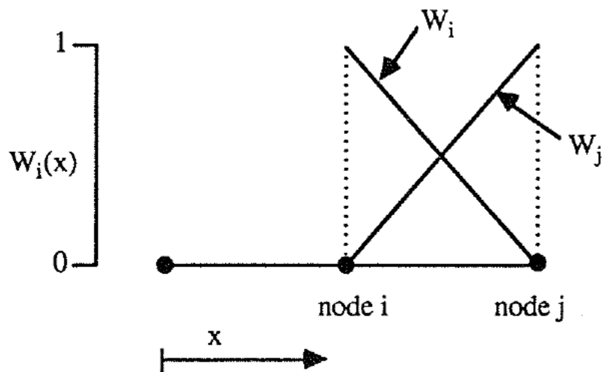


Figure: Interpolationsfunktion für die Galerkin-Methode nach Istok (1989)

Die gleiche Beziehung lässt sich den anderen Element-Knoten j schreiben.

$$R_j^{(e)} = \int_{x_i^{(e)}}^{x_j^{(e)}} N_j^{(e)} \left(K_x^{(e)} \frac{\partial^2 \hat{h}^{(e)}}{\partial x^2} \right) dx \tag{6}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Finite-Elemente-Methode (FEM)

Die linearen Interpolationsfunktionen für 1D Elemente sind

$$N_i^{(e)}(x) = \frac{x_j^{(e)} - x}{L^{(e)}} \quad , \quad N_j^{(e)}(x) = \frac{x - x_i^{(e)}}{L^{(e)}} \tag{7}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Die approximierte Feldgröße h kann nun wie folgt auf dem 1D finiten Element interpoliert werden (Abb. 3).

$$\begin{aligned} \hat{h}^{(e)}(x) &= N_i^{(e)} h_i + N_j^{(e)} h_j \\ &= \frac{x_j^{(e)} - x}{L^{(e)}} h_i + \frac{x - x_i^{(e)}}{L^{(e)}} h_j \end{aligned} \tag{8}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Finite-Elemente-Methode (FEM)

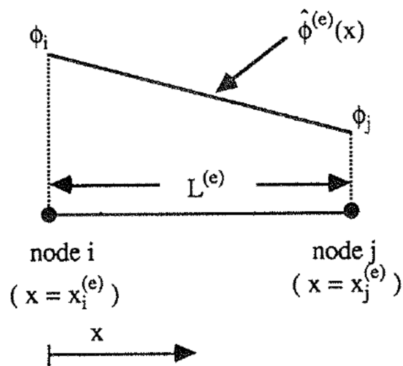


Figure: Interpolierte Näherungslösung auf einem 1D Element nach Istok (1989)

Jetzt stoßen wir auf ein ernsthaftes Problem. In den Gleichungen (5) und (6) müssten wir Ableitungen zweiter Ordnung berechnen, unsere Interpolationsfunktionen sind aber linear - also existieren keine zweiten Ableitung - was tun? Wir machen eine mathematischen Trick in diesen Gleichungen. Ein partielle Integration von (5) ergibt.

$$\int_{x_i^e}^{x_j^e} N_i^{(e)} \left(K_x^{(e)} \frac{\partial^2 \hat{h}^{(e)}}{\partial x^2} \right) dx \tag{9}$$

$$= - \int_{x_i^e}^{x_j^e} K_x^{(e)} \frac{\partial N_i^{(e)}}{\partial x} \frac{\partial \hat{h}^{(e)}}{\partial x} dx + N_i^{(e)} K_x^{(e)} \frac{\partial \hat{h}^{(e)}}{\partial x} \Bigg|_{x_i^e}^{x_j^e}$$

TECHNISCHE UNIVERSITÄT DRESDEN

- Wie können wir die Umformung in Gleichung (10) überprüfen?

Der zweite Term auf der rechten Seite von (10)

$$N_i^{(e)} K_x^{(e)} \frac{\partial \hat{h}^{(e)}}{\partial x} \bigg|_{x_i^e}^{x_j^e} \tag{10}$$

entspricht der Vorgabe von Werten auf den Randknoten x_j^e und x_i^e des Elements (e). Handelt es sich um einen Randknoten, dann geht es um Randbedingungen.

Finite-Elemente-Methode (FEM)

- Um welchen Randbedingungstypen handelt es bei (10)?
- Welche Randbedingung ist es, wenn der Wert von (10) gleich Null ist?
- Was passiert mit inneren Knoten?

TECHNISCHE UNIVERSITÄT DRESDEN

Nun setzen wir die Beziehung (10) in die Gleichung (5) ein und erhalten.

$$
\begin{aligned}
R_i^{(e)} &= \int_{x_i^e}^{x_j^e} N_i^{(e)} \left(K_x^{(e)} \frac{\partial^2 \hat{h}^{(e)}}{\partial x^2} \right) dx \\
&= -\int_{x_i^e}^{x_j^e} K_x^{(e)} \frac{\partial N_i^{(e)}}{\partial x} \frac{\partial \hat{h}^{(e)}}{\partial x} dx + N_i^{(e)} K_x^{(e)} \frac{\partial \hat{h}^{(e)}}{\partial x} \Bigg|_{x_i^e}^{x_j^e} \qquad (11)
\end{aligned}
$$

TECHNISCHE UNIVERSITÄT DRESDEN

Finite-Elemente-Methode (FEM)

Jetzt müssen wir uns um $\partial \hat{h}^{(e)} / \partial x$ kümmern.

$$\frac{\partial \hat{h}^{(e)}}{\partial x} = \frac{\partial}{\partial x} \left(N_i^{(e)} h_i + N_j^{(e)} h_j \right) = \frac{\partial N_i^{(e)}}{\partial x} h_i + \frac{\partial N_j^{(e)}}{\partial x} h_j \qquad (12)$$

TECHNISCHE UNIVERSITÄT DRESDEN

Finite-Elemente-Methode (FEM)

Nach Einsetzen der Interpolationsfunktionen erhalten wir schließlich.

$$\frac{\partial \hat{h}^{(e)}}{\partial x} = \frac{1}{L^{(e)}}(-h_i + h_j) \tag{13}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Für die Ableitungen der linearen Interpolationsfunktionen folgt.

$$\frac{\partial N_i^{(e)}}{\partial x} = \frac{\partial}{\partial x}\left(\frac{x_j^{(e)} - x}{L^{(e)}}\right) = -\frac{1}{L^{(e)}} \tag{14}$$

$$\frac{\partial N_j^{(e)}}{\partial x} = \frac{\partial}{\partial x}\left(\frac{x - x_i^{(e)}}{L^{(e)}}\right) = \frac{1}{L^{(e)}} \tag{15}$$

Setzen wir jetzt die Beziehungen in die Gleichung (11) ein, ergibt sich.

$$
\begin{aligned}
R_i^{(e)} &= \int_{x_i^e}^{x_j^e} K_x^{(e)} \left(-\frac{1}{L^{(e)}}\right) \left(\frac{1}{L^{(e)}}\right) (-h_i + h_j) \\
&= \frac{K_x^{(e)}}{L^{(e)}} (h_i - h_j)
\end{aligned}
\tag{16}
$$

In gleicher Weise bekommen wir.

$$
R_j^{(e)} = \frac{K_x^{(e)}}{L^{(e)}} (-h_i + h_j) \tag{17}
$$

Beide Gleichungen (16) und (17) lassen sich zusammen in einer Matrizen-Form schreiben.

$$\left\{ \begin{array}{c} R_i^{(e)} \\ R_j^{(e)} \end{array} \right\} = \frac{K_x^{(e)}}{L^{(e)}} \underbrace{\left[\begin{array}{cc} +1 & -1 \\ -1 & +1 \end{array} \right]}_{2\times 2} \left\{ \begin{array}{c} h_i \\ h_j \end{array} \right\} \tag{18}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Finite-Elemente-Methode (FEM)

Leitfähigkeitsmatrix

$$[K^{(e)}] = \frac{K_x^{(e)}}{L^{(e)}} \underbrace{\left[\begin{array}{cc} +1 & -1 \\ -1 & +1 \end{array}\right]}_{2\times 2} \quad (19)$$

TECHNISCHE UNIVERSITÄT DRESDEN

Aufgrund der Elementgeometrien $L^{(e)}$ (Abb. 2) ergeben sich folgende Elementmatrizen.

$$[K^{(1)}] = \begin{bmatrix} +1/2 & -1/2 \\ -1/2 & +1/2 \end{bmatrix} \quad , \quad [K^{(2)}] = \begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix}$$
$$\left[K^{(3)}\right] = \begin{bmatrix} +1/3 & -1/3 \\ -1/3 & +1/3 \end{bmatrix} \quad , \quad [K^{(4)}] = \begin{bmatrix} +1/3 & -1/3 \\ -1/3 & +1/3 \end{bmatrix} \tag{20}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Zusammenbauen des Gleichungssystems.

$$\{\mathbf{R}\} = [\mathbf{K}]\{\mathbf{h}\} = 0 \tag{21}$$

$$\{\mathbf{R}\} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \end{bmatrix} \quad , \quad \{\mathbf{h}\} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{bmatrix} \tag{22}$$

$$[\mathbf{K}] = [\mathbf{K}^{(1)}] + [\mathbf{K}^{(2)}] + [\mathbf{K}^{(3)}] + [\mathbf{K}^{(4)}] \tag{23}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Finite-Elemente-Methode (FEM)

$$
[\mathbf{K}] = \begin{bmatrix} 1/2 & -1/2 & 0 & 0 & 0 \\ -1/2 & 1+1/2 & -1 & 0 & 0 \\ 0 & -1 & 1+1/3 & -1/3 & 0 \\ 0 & 0 & -1/3 & 1/3+1/3 & -1/3 \\ 0 & 0 & 0 & -1/3 & 1/3 \end{bmatrix}
= \begin{bmatrix} 1/2 & -1/2 & 0 & 0 & 0 \\ -1/2 & 3/2 & -1 & 0 & 0 \\ 0 & -1 & 4/3 & -1/3 & 0 \\ 0 & 0 & -1/3 & 2/3 & -1/3 \\ 0 & 0 & 0 & -1/3 & 1/3 \end{bmatrix} \tag{24}
$$

TECHNISCHE UNIVERSITÄT DRESDEN

Finite-Elemente-Methode (FEM)

$$\begin{bmatrix} 1/2 & -1/2 & 0 & 0 & 0 \\ -1/2 & 3/2 & -1 & 0 & 0 \\ 0 & -1 & 4/3 & -1/3 & 0 \\ 0 & 0 & -1/3 & 2/3 & -1/3 \\ 0 & 0 & 0 & -1/3 & 1/3 \end{bmatrix} \begin{Bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \quad (25)$$

TECHNISCHE UNIVERSITÄT DRESDEN

Vergleichen wir die Quelltexte der `main` Funktionen für FD und FE Verfahren, sehen wir kaum Unterschiede. Das heisst die Abläufe (Algorithmen) sind sehr ähnlich.

Finite-Elemente-Methode (FEM)

```
extern void Gauss(double*,double*,double*,int);

int main()
{
  //-----------------------------------------------
  FEM* fem = new FEM();
  fem->SetInitialConditions();
  fem->SetBoundaryConditions();
  //-----------------------------------------------
  int tn = 10;
  for(int t=0;t<tn;t++)
  {
    fem->AssembleEquationSystem();
    Gauss(fem->matrix,fem->vecb,fem->vecx,fem->IJ);
    fem->SaveTimeStep();
    fem->OutputResults(t);
  }
  //-----------------------------------------------
  return 0;
}
```

TECHNISCHE UNIVERSITÄT DRESDEN