

# Modellierung von Hydrosystemen - SoSe 2026

## UW-BHW-625: B2-T1.3a: Finite-Differenzen-Methode: Explizit

Olaf Kolditz, Lars Bilke, Karsten Rink, Haibing Shao, Erik Nixdorf

<sup>1</sup>Helmholtz Centre for Environmental Research – UFZ, Leipzig

<sup>2</sup>Technische Universität Dresden – TUD, Dresden

<sup>3</sup>Center for Advanced Water Research – CAWR

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

<sup>4</sup>Bundesanstalt für Geowissenschaften und Rohstoffe – BGR, Hannover / Cottbus

Dresden, 22.05.2026

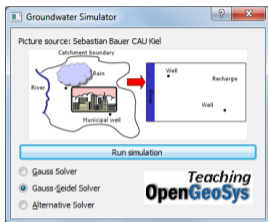
# Zeitplan: Modellierung von Hydrosystemen: Zweiter Block (B2)

Sommersemester 2026: UW-BHW-625

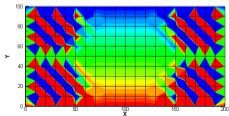
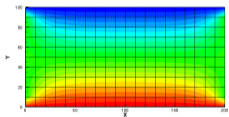
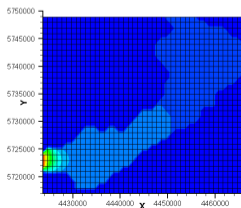
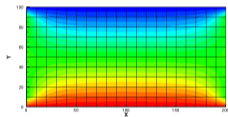
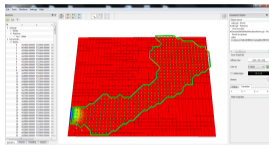
Datum	B2	Thema	Format
17.04.2026	B2-T1.0	Einführung in die Veranstaltung (B2) (Kolditz)	HSZ/403
22.05.2026	B2-T1.1	Hydromechanik und Numerische Methoden (Kolditz)	HSZ/403
22.05.2026	B2-T1.2	Grundwasserhydraulik und Prinzipbeispiel (Kolditz)	HSZ/403
22.05.2026	B2-T1.3	Finite-Differenzen-Methode (Kolditz)	HSZ/403
19.06.2026	B2-T1.4	Finite-Elemente-Methode, Grundwassersysteme (Kolditz)	HSZ/403
26.06.2026	B2-T4.1	Virtuelle VISLAB Tour - Vorlesung (Rink/Bilke)	Online
26.06.2026	B2-T4.2	Virtuelle VISLAB Tour - Demo (Rink/Bilke)	Online
03.07.2026	B2-T2.1	Regionale Grundwassersysteme (Nixdorf, BGR)	HSZ/403
03.07.2026	B2-T2.2	Regionale Grundwassersysteme (Nixdorf, BGR)	HSZ/403
03.07.2026	B2-T2.3	Regionale Grundwassersysteme (Nixdorf, BGR): Übung	HSZ/403
10.07.2026	B2-T3.1	Stofftransport in Hydrosystemen (Selzer/Shao)	HSZ/403
10.07.2026	B2-T3.2	Stofftransport in Hydrosystemen (Selzer/Shao)	HSZ/403
10.07.2026	B2-T3.3	Stofftransport in Hydrosystemen (Selzer/Shao)	HSZ/403
17.07.2026	B2-T1.5	Workflow, Grundwassersysteme (Kolditz)	HSZ/403
24.07.2026	B2-T1.6	Zusammenfassung der Veranstaltung Numerik (Kolditz)	HSZ/403
24.07.2026	B2-T1.7	Zusammenfassung der Veranstaltung (Hartmann/Kolditz)	HSZ/403
24.07.2026	B2-T1.8	Vorbereitung Klausur (Hartmann/Kolditz)	HSZ/403

# Übersicht: Numerische Verfahren

## explizite FDM

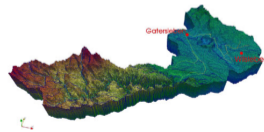
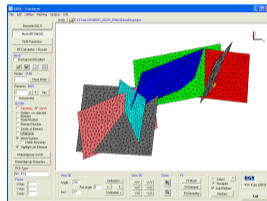


## implizite FDM



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

## FEM



## 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

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

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

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

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

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

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

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

## ► Zeitableitung

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

## ► in space

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

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

$$S_{i,j} \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} - K_{i,j}^x \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} - K_{i,j}^y \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2} = Q_{i,j} \quad (8)$$

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

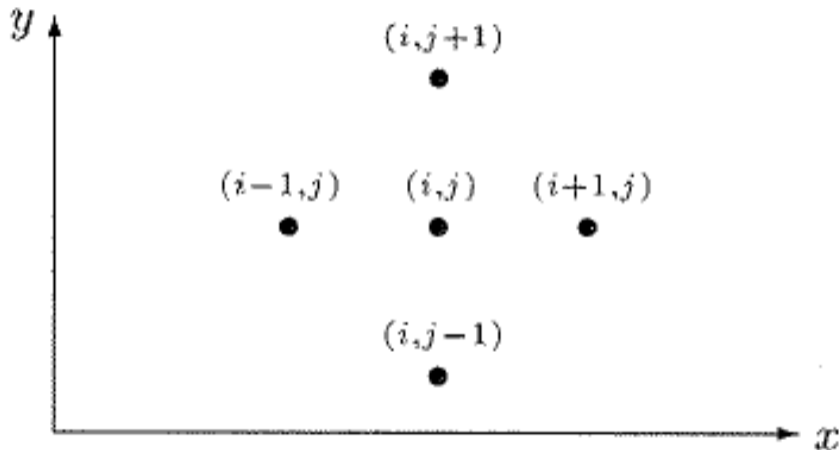


Fig.: 5-Punkte-Stern (Knabner und Angermann 2000)

# Übung

- Finite-Differenzen-Methode: Explizites Verfahren

GitHub-Repository für die Übungen:  
`https://github.com/OlafKolditz/HYDROSYSTEMS`

## ► 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 (9) sind damit:

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

Listing 1: Datenstrukturen

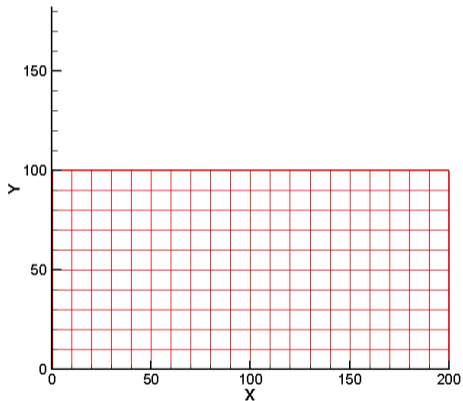


Fig.: Rechen-Gitter für den Rechteck-Aquifer

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

```
1 for(j=0;j<jy;j++)
2 {
3     nn = j*ix;
4     for( i=0;i<ix;i++)
5     {
6         n = nn+i;
7         u_new[n] = u[n] \
8                 + Kf/S0*dt/dx2 * (u[n+1]-2*u[n]+u[n-1]) \
9                 + Kf/S0*dt/dy2 * (u[(j+1)*ix+i]-2*u[n]+u[(j-1)*ix+i]) \
10                + Q/S0;
11     }
12 }
```

Listing 2: Rechenschema

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.

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

Listing 3: Speicher für Lösungsvektoren

- ▶ Welche Rolle spielen  $ix$  und  $jy$  bei der Speicherreservierung?

Natürlich müssen auch die Parameter vor der Berechnung initialisiert werden

```
1  ix = 21;  
2  jy = 11;  
3  dx = 10.; //Einheiten  
4  dy = 10.;  
5  dt = 0.25e2;  
6  S0 = 1e-5;  
7  Kf = 1e-5;  
8  Q  = 0.;  
9  u0 = 0.;
```

Listing 4: Daten-Initialisierung

- ▶ Welche Einheiten haben die einzelnen Parameter?

Das mit den Anfangsbedingungen ist eine einfache Sache. Mit der Doppelschleife über alle Knoten, können wir sehr einfach einen Wert  $u_0$  als Anfangsbedingung überall zuweisen.

```
1 for(int i=0;i<ix;i++)
2   for(int j=0;j<jy;j++)
3   {
4     u[j*(ix+1)] = u0;
5     u_new[j*(ix+1)] = u0;
6   }
7 }
```

Listing 5: Anfangsbedingungen setzen

Mit den Randbedingungen ist es etwas kniffliger ...

```
1 //top and bottom
2 int l;
3 for(int i=0;i<ix;i++)
4 {
5     bc_nodes.push_back(i); u[i] = u_top    u_new[i] = u_top;
6     l = ix*(jy-1)+i;
7     bc_nodes.push_back(l); u[l] = u_bottom;    u_new[l] = u_bottom;
8 }
9 //left and right side
10 for(int j=1;j<jy-1;j++)
11 {
12     l = ix*j;
13     bc_nodes.push_back(l); u[l] = u_left;    u_new[l] = u_left;
14     l = ix*j+ix-1;
15     bc_nodes.push_back(l); u[l] = u_right;    u_new[l] = u_right;
16 }
```

Listing 6: Randbedingungen setzen

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

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

Listing 7: Vektor für Randbedingungen

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.

```
1 for(int j=0;j<jy;j++)
2 {
3     nn = j*ix;
4     for(int i=0;i<ix;i++)
5     {
6         n = nn+i;
7         if(IsBCNode(n, bc_nodes))
8             continue;
9         ...
10 }
```

Listing 8: Randbedingungen setzen

Wie funktioniert nun IsBCNode?

```
1 bool IsBCNode(int n, std::vector<int>bc_nodes)
2 {
3     bool is_node_bc = false;
4     for(int k=0;k<(size_t)bc_nodes.size();k++)
5     {
6         if(n==bc_nodes[k])
7         {
8             is_node_bc = true;
9             return is_node_bc;
10        }
11    }
12    return is_node_bc;
13 }
```

Listing 9: Randbedingungen setzen

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.

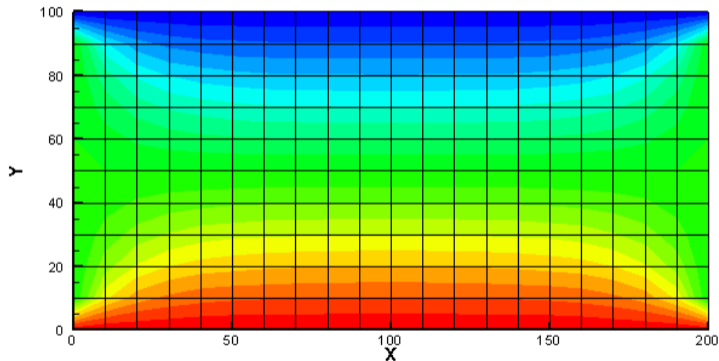
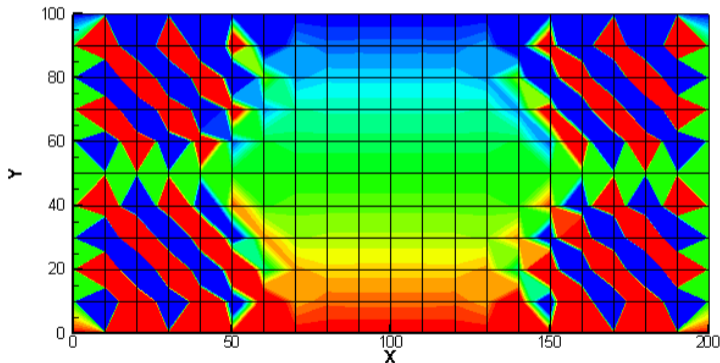


Fig.: Berechnete Druckverteilung im Rechteck-Aquifer nach 100 Zeitschritten  $\Delta t = 25 \text{ sec}$

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?



# Numerische Stabilität

- Finite-Differenzen-Methode: Explizites Verfahren

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.

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

Prima, aber was ist jetzt  $\alpha$  und warum steht nur  $\Delta x$  und nicht auch  $\Delta y$  in der Gleichung? Zur bestimmung des  $\alpha$  schreiben wir die Grundwassergleichung in eine Diffusionsgleichung wie folgt um.

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

Wir sehen, dass es eigentlich zwei  $\alpha$ -s gibt, für jede Richtung eins.

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

- ▶ Welche Einheit hat unser Grundwasser- $\alpha$  ?

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

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

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

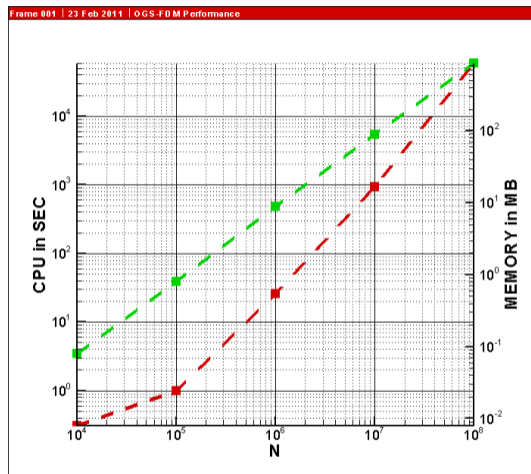


Fig.: Rechenzeit und Speicherbedarf für explizite FDM

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.

# Übung

- Explizite FDM

[https://github.com/OlafKolditz/HYDROSYSTEMS/tree/main/BHYWI-22-E2\\_FDM-explizit-Rechteck-python](https://github.com/OlafKolditz/HYDROSYSTEMS/tree/main/BHYWI-22-E2_FDM-explizit-Rechteck-python)

```
1 cd ... \HYDROSYSTEMS \BHYWI-22-E2_FDM-explizit-Rechteck-python
```

Listing 10: Verzeichnis auswählen

run.bat

```
1 set PATH=%PATH%;C:\MinGW\bin  
2 g++ main.cpp  
3 a.exe  
4 python isolines.py
```

Listing 11: Skript

# Modellierung von Hydrosystemen - SoSe 2026

## UW-BHW-625: B2-T1.3b: Finite-Differenzen-Methode: Implizit

Olaf Kolditz, Lars Bilke, Karsten Rink, Haibing Shao, Erik Nixdorf

<sup>1</sup>Helmholtz Centre for Environmental Research – UFZ, Leipzig

<sup>2</sup>Technische Universität Dresden – TUD, Dresden

<sup>3</sup>Center for Advanced Water Research – CAWR

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

<sup>4</sup>Bundesanstalt für Geowissenschaften und Rohstoffe – BGR, Hannover / Cottbus

Dresden, 22.05.2026

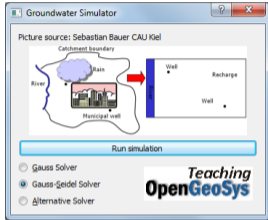
# Zeitplan: Modellierung von Hydrosystemen: Zweiter Block (B2)

Sommersemester 2026: UW-BHW-625

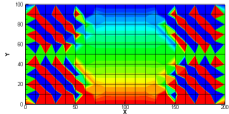
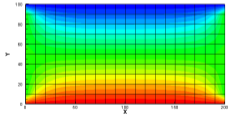
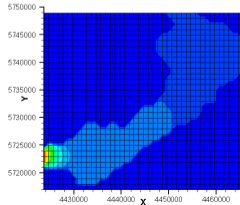
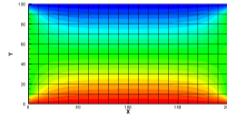
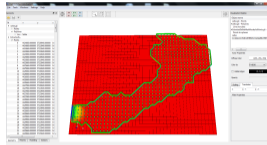
Datum	B2	Thema	Format
17.04.2026	B2-T1.0	Einführung in die Veranstaltung (B2) (Kolditz)	HSZ/403
22.05.2026	B2-T1.1	Hydromechanik und Numerische Methoden (Kolditz)	HSZ/403
22.05.2026	B2-T1.2	Grundwasserhydraulik und Prinzipbeispiel (Kolditz)	HSZ/403
22.05.2026	B2-T1.3	Finite-Differenzen-Methode (Kolditz)	HSZ/403
19.06.2026	B2-T1.4	Finite-Elemente-Methode, Grundwassersysteme (Kolditz)	HSZ/403
26.06.2026	B2-T4.1	Virtuelle VISLAB Tour - Vorlesung (Rink/Bilke)	Online
26.06.2026	B2-T4.2	Virtuelle VISLAB Tour - Demo (Rink/Bilke)	Online
03.07.2026	B2-T2.1	Regionale Grundwassersysteme (Nixdorf, BGR)	HSZ/403
03.07.2026	B2-T2.2	Regionale Grundwassersysteme (Nixdorf, BGR)	HSZ/403
03.07.2026	B2-T2.3	Regionale Grundwassersysteme (Nixdorf, BGR): Übung	HSZ/403
10.07.2026	B2-T3.1	Stofftransport in Hydrosystemen (Selzer/Shao)	HSZ/403
10.07.2026	B2-T3.2	Stofftransport in Hydrosystemen (Selzer/Shao)	HSZ/403
10.07.2026	B2-T3.3	Stofftransport in Hydrosystemen (Selzer/Shao)	HSZ/403
17.07.2026	B2-T1.5	Workflow, Grundwassersysteme (Kolditz)	HSZ/403
24.07.2026	B2-T1.6	Zusammenfassung der Veranstaltung Numerik (Kolditz)	HSZ/403
24.07.2026	B2-T1.7	Zusammenfassung der Veranstaltung (Hartmann/Kolditz)	HSZ/403
24.07.2026	B2-T1.8	Vorbereitung Klausur (Hartmann/Kolditz)	HSZ/403

# Übersicht: Numerische Verfahren

## explizite FDM

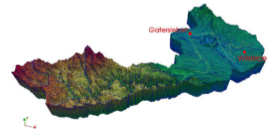
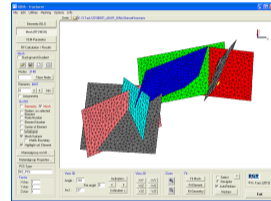


## implizite FDM

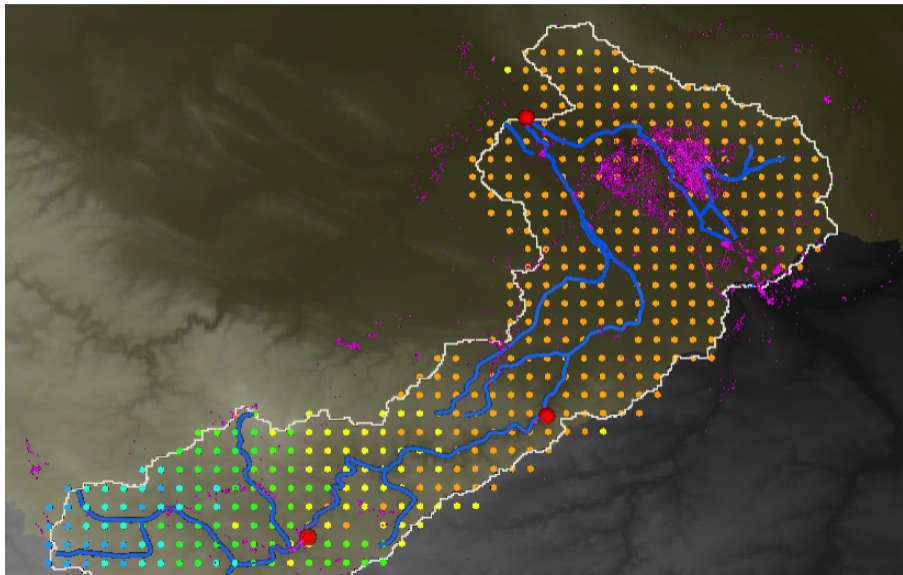


- Pro / Cons
- FDM: einfache Implementierung, starre Geometrien
- FEM: schwieriger zu implementieren (heute), flexible Geometrien

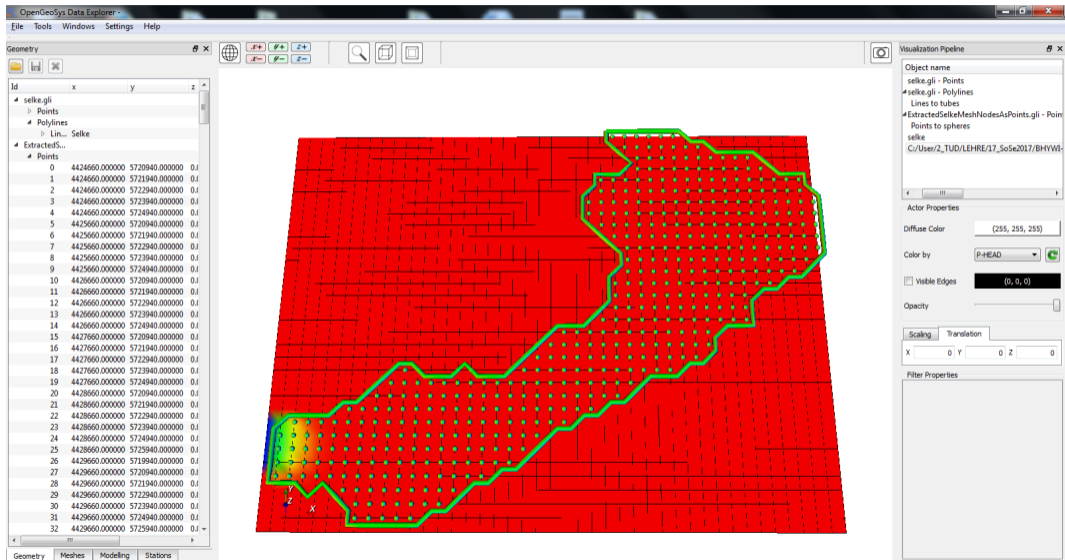
## FEM



# Selke Einzugsgebiet



# Selke Einzugsgebiet



siehe auch Abschn. 4.2 Hydroinformatik II

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

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

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

► Differenzen-Schema

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

► Gleichungssystem

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

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.

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

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

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

$$Ne = \frac{K \Delta t}{S \Delta l^2} \quad (6)$$

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).



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. Die wesentlichen Unterschiede der impliziten zur expliziten FDM sind, dass wir ein Gleichungssystem aufbauen und lösen müssen.

# 2D implizite FDM - die main function

```
1 #include <iostream>
2 #include "fdm.h"
3 #include <time.h>
4 extern void Gauss(double*,double*,double*,int);
5 int main(int argc, char *argv[])
6 {
7     //-----
8     FDM* fdm = new FDM();
9     fdm->SetInitialConditions();
10    fdm->SetBoundaryConditions();
11    //-----
12    int tn = 2;
13    for(int t=0;t<tn;t++)
14    {
15        fdm->AssembleEquationSystem();
16        Gauss(fdm->matrix,fdm->vecb,fdm->vecx,fdm->IJ);
17        fdm->SaveTimeStep();
18        fdm->OutputResults(t);
19    }
20    //-----
21    fdm->out_file.close();
22    return 0;
23 }
```

Listing 1: OOP main function

Dennoch können wir erstaunlich viel wiederverwenden, bis auf

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

## Listing 2: Rechenschema

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.

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)

```
1 void FDM::AssembleEquationSystem()
2 {
3     // Matrix entries
4     for(i=0;i<IJ;i++)
5     {
6         vecb[i] = u[i];
7         for(j=0;j<IJ;j++)
8         {
9             matrix[i*IJ+j] = 0.0;
10            if(i==j)
11                matrix[i*IJ+j] = 1. + 4.*Ne;
12            else if(abs((i-j))==1)
13                matrix[i*IJ+j] = - Ne;
14        }
15    }
16    // Incorporate boundary conditions
17    IncorporateBoundaryConditions();
18    // Matrix output
19    WriteEquationSystem();
20 }
```

Listing 3: Gleichungssystem aufbauen

# Übung

- Explizite FDM

`https://github.com/OlafKolditz/HYDROSYSTEMS/tree/main/BHYWI-22-E3\_FDM-implizit-Rechteck-python`

```
1 cd ... \HYDROSYSTEMS\BHYWI-22-E3_FDM-implizit-Rechteck-python
```

## Listing 4: Verzeichnis auswaehlen

### run.bat

```
1 set PATH=%PATH%;C:\MinGW\bin  
2 g++ main.cpp fdm.cpp solver.cpp  
3 a.exe  
4 python isolines.py
```

## Listing 5: Skript