BHYWI-22: Semester-Fahrplan: 2019

Vorlesungen & Übungen

Kolditz		*Hydromechanik und numerische Methoden*
05 Apr 2019	V2-01	Einführungin die Veranstaltung
12 Apr 2019	V2-02	Hydromechanik / Numerische Methoden: Wiederholung
19 Apr 2019		Ostern
		Numerische Methoden
26 Apr 2019	V2-03	Einzugsgebiet: Übung
03 Mai 2019	V2-04	Numerik: Finite-Differenzen-Verfahren (2D): Vorlesung
10 Mai 2019	V2-05	Numerik: Finite-Differenzen-Verfahren (2D): Übung
17 Mai 2019	V2-06	Software: Objekt-Orientierte FDM
24 Mai 2019	V2-07	Selke-Modell: Übung
31 Mai 2019	V2-08	Numerik: Finite-Elemente-Verfahren: Vorlesung
07 Jun 2019	V2-09	Numerik: Finite-Elemente-Verfahren: Übung
14 Jun 2019	V2-10	Pfingsten
		Datenbasierte Methoden
21 Jun 2019	V1+2	*VISLAB Exkursion*
28 Jun 2019	V2-11	Hydrologeologische Modellierung: Datenbasierte Verfahren I
05 Jul 2019	V2-12	Hydrologeologische Modellierung: Datenbasierte Verfahren II
12 Jul 2019	V2-13	Klausurvorbereitung

ECHNISCHE
NIVERSITÄT
RESDEN

Modellierung von Hydrosystemen

"Numerische und daten-basierte Methoden"

BHYWI-22-04 @ 2019

Finite-Differenzen-Methode (FDM) 2D

Olaf Kolditz

*Helmholtz Centre for Environmental Research – UFZ

[1]Technische Universität Dresden – TUDD

[2]Centre for Advanced Water Research – CAWR

03/10.05.2019 - Dresden

TECHNISCHE UNIVERSITÄT DRESDEN

Fahrplan für heute ...

- Übersicht Übungen
- Grundlagen - GWE
- Grundlagen - TSE
- Übungen zur expliziten FDM (page 12) [BHYWI-22-E2]
- TSE double-check
- Übung zur expliziten FDM [BHYWI-22-E2]
- Übung zu OOP-FDM [BHYWI-22-E4]

TECHNISCHE UNIVERSITÄT DRESDEN

Grundlagen - GWE

- 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

Grundlagen - TSE

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

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

TECHNISCHE UNIVERSITÄT DRESDEN

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

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

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

TECHNISCHE UNIVERSITÄT DRESDEN

FDM - explizites Schema

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

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

TECHNISCHE UNIVERSITÄT DRESDEN

Grundlagen - FDM

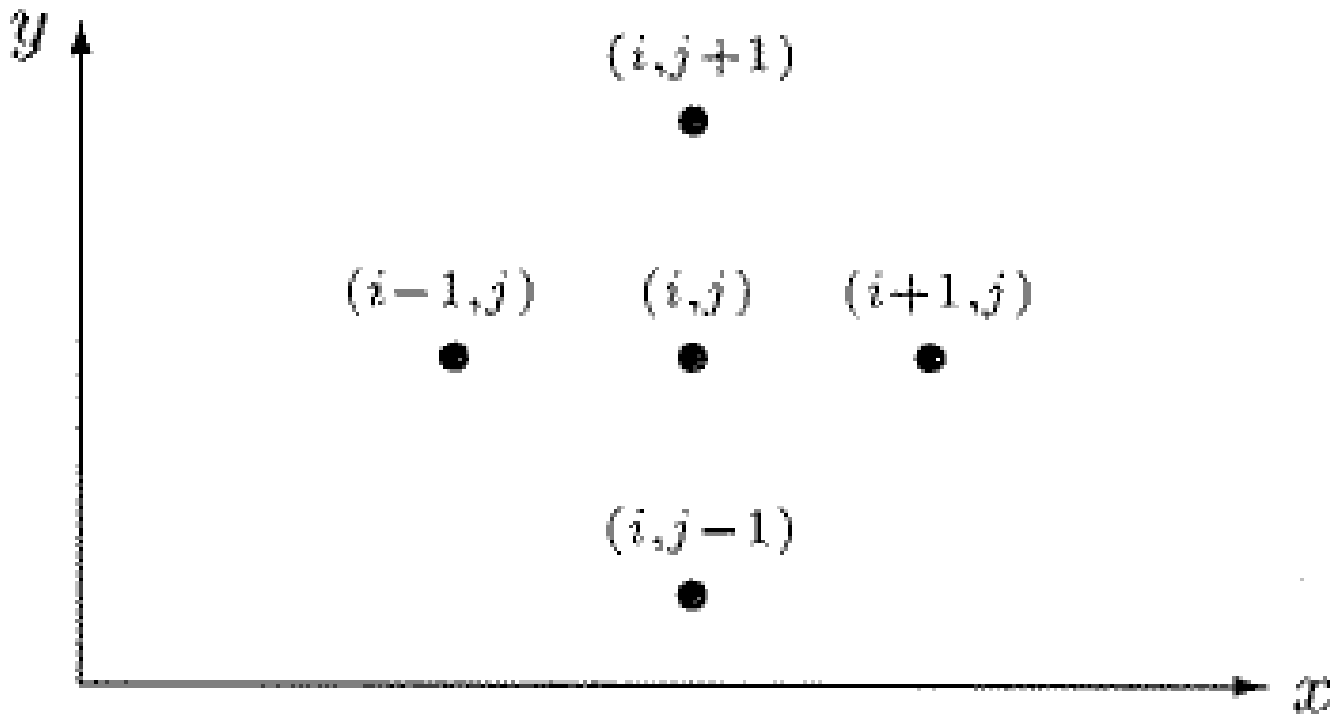


Figure: 5-Punkte-Stern (Knabner und Angermann 2000)

Übung - BHYWI-22-E2 - QAD

- Datenstrukturen

Table:

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

TECHNISCHE UNIVERSITÄT DRESDEN

Übung - BHYWI-22-E2 - QAD

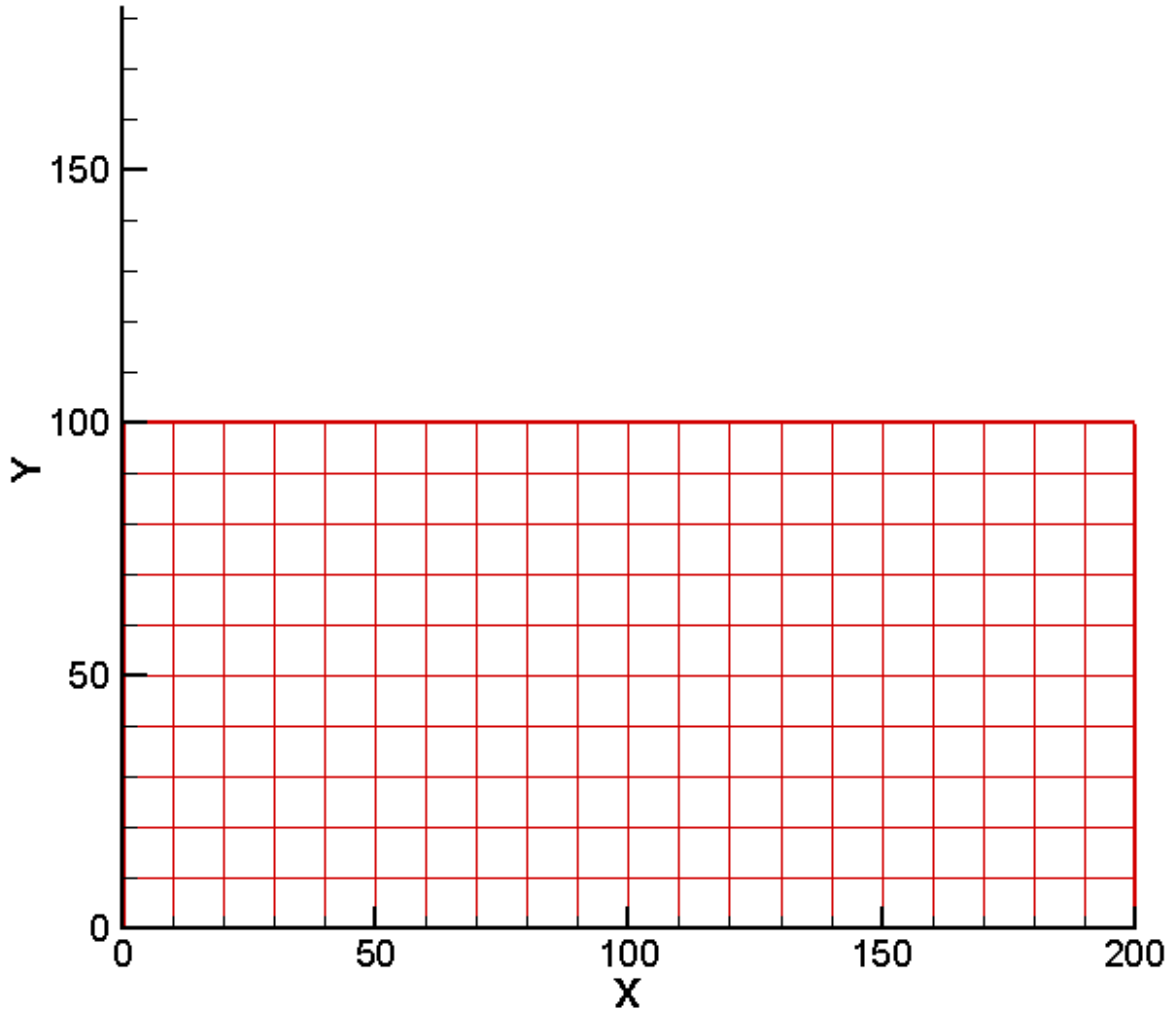
Die Minimal-Datenstrukturen für die Programmierung der Gleichung (9) sind damit:

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

Restart

TECHNISCHE UNIVERSITÄT DRESDEN

Übung - BHYWI-22-E2 - QAD



TECHNISCHE UNIVERSITÄT DRESDEN

Übung - BHYWI-22-E2 - QAD - V1

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

Übung - BHYWI-22-E2 - QAD - V2

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

Übung - BHYWI-22-E2 - QAD

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

Übung - BHYWI-22-E2 - QAD

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

Übung - BHYWI-22-E2 - QAD

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

Übung - BHYWI-22-E2 - QAD

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;
}
```

TECHNISCHE UNIVERSITÄT DRESDEN

Übung - BHYWI-22-E2 - QAD

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.

TECHNISCHE UNIVERSITÄT DRESDEN

Übung - BHYWI-22-E2 - QAD

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

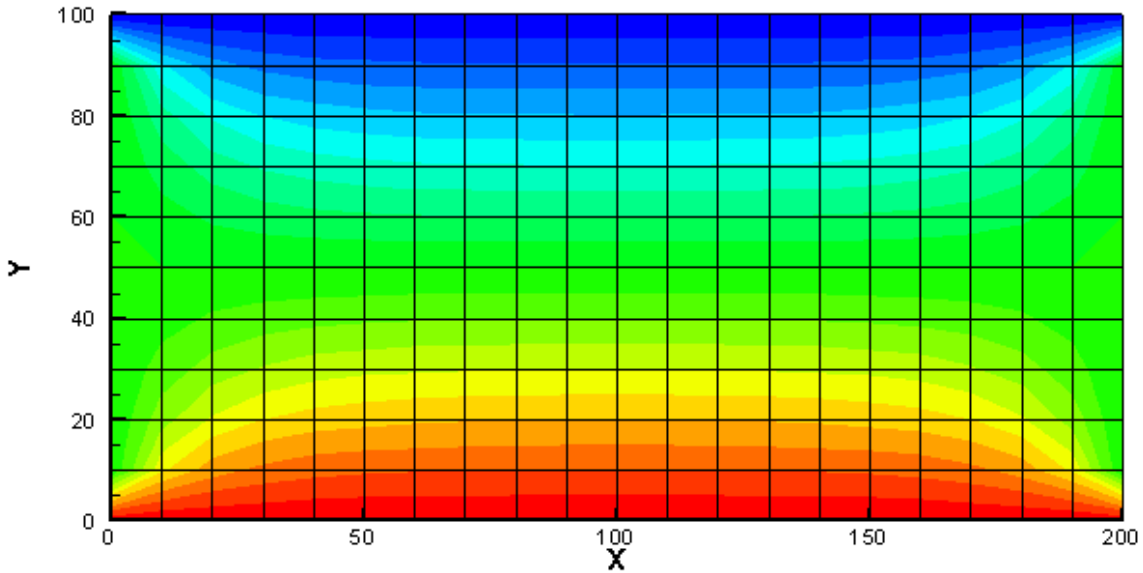


Figure: Berechnete Druckverteilung im Rechteck-Aquifer nach 100 Zeitschritten $\Delta t = 25$ sec

TECHNISCHE UNIVERSITÄT DRESDEN

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?

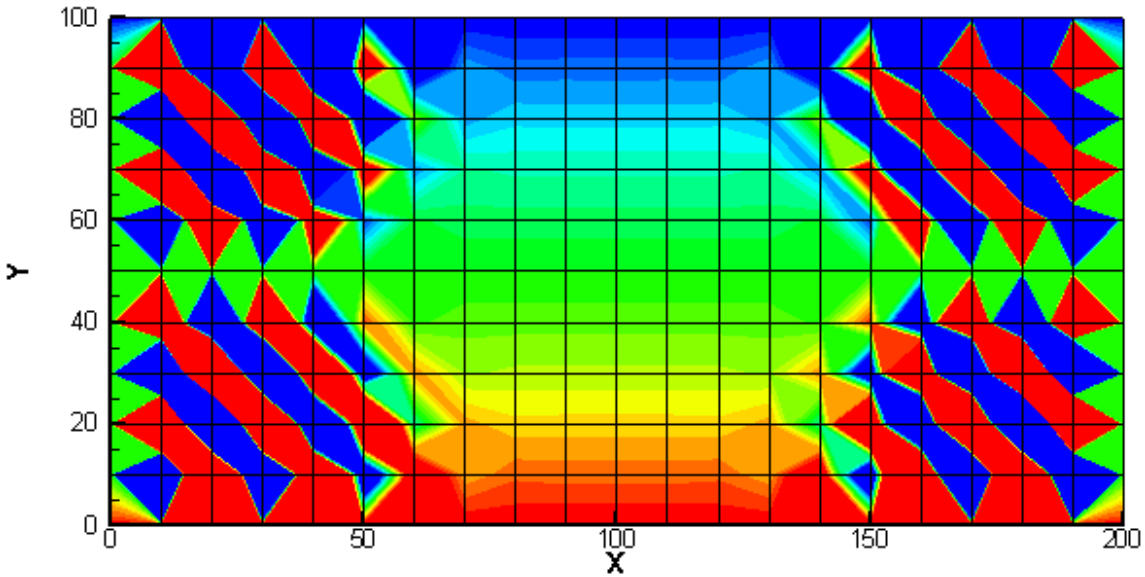


Figure: Berechnete Druckverteilung im Rechteck-Aquifer nach 100 Zeitschritten $\Delta t = 50$ sec

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

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.

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

Übung - BHYWI-22-E2 - QAD

Wir sehen, dass es eigentlich zwei α-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-α ?

Übung - BHYWI-22-E2 - QAD

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

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

TECHNISCHE UNIVERSITÄT DRESDEN

Übung - BHYWI-22-E2 - QAD

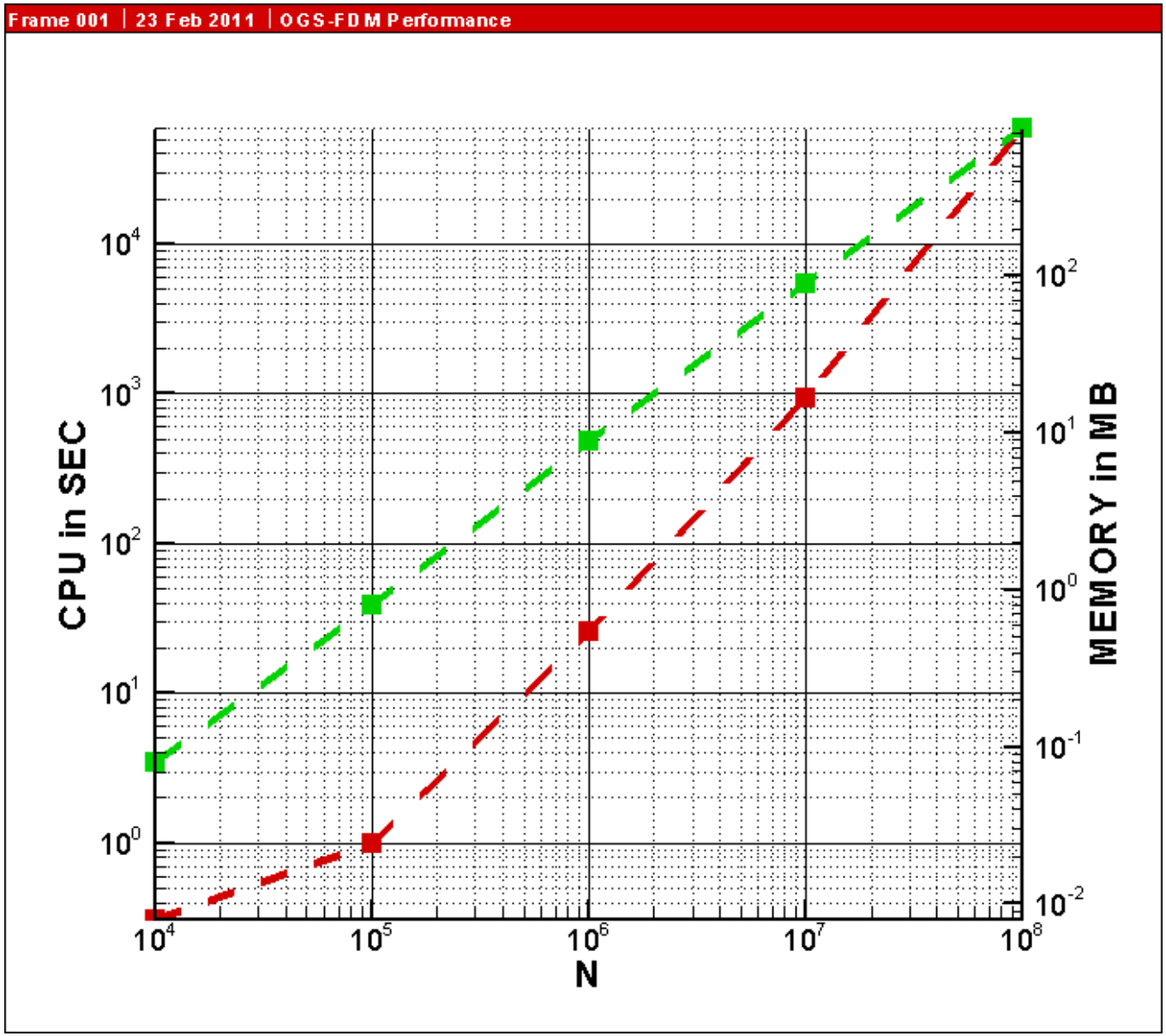
Die numerischen und hydraulischen Parameter sind in der nachfolgenden Tabelle **??** zu finden. Glücklicherweise sind die Ortdiskretisierungen und die hydraulischen Leitfähigkeiten in beide Koordinatenrichtungen gleich (isotropes Problem).

TECHNISCHE UNIVERSITÄT DRESDEN

Übung - BHYWI-22-E2 - QAD

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

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
```

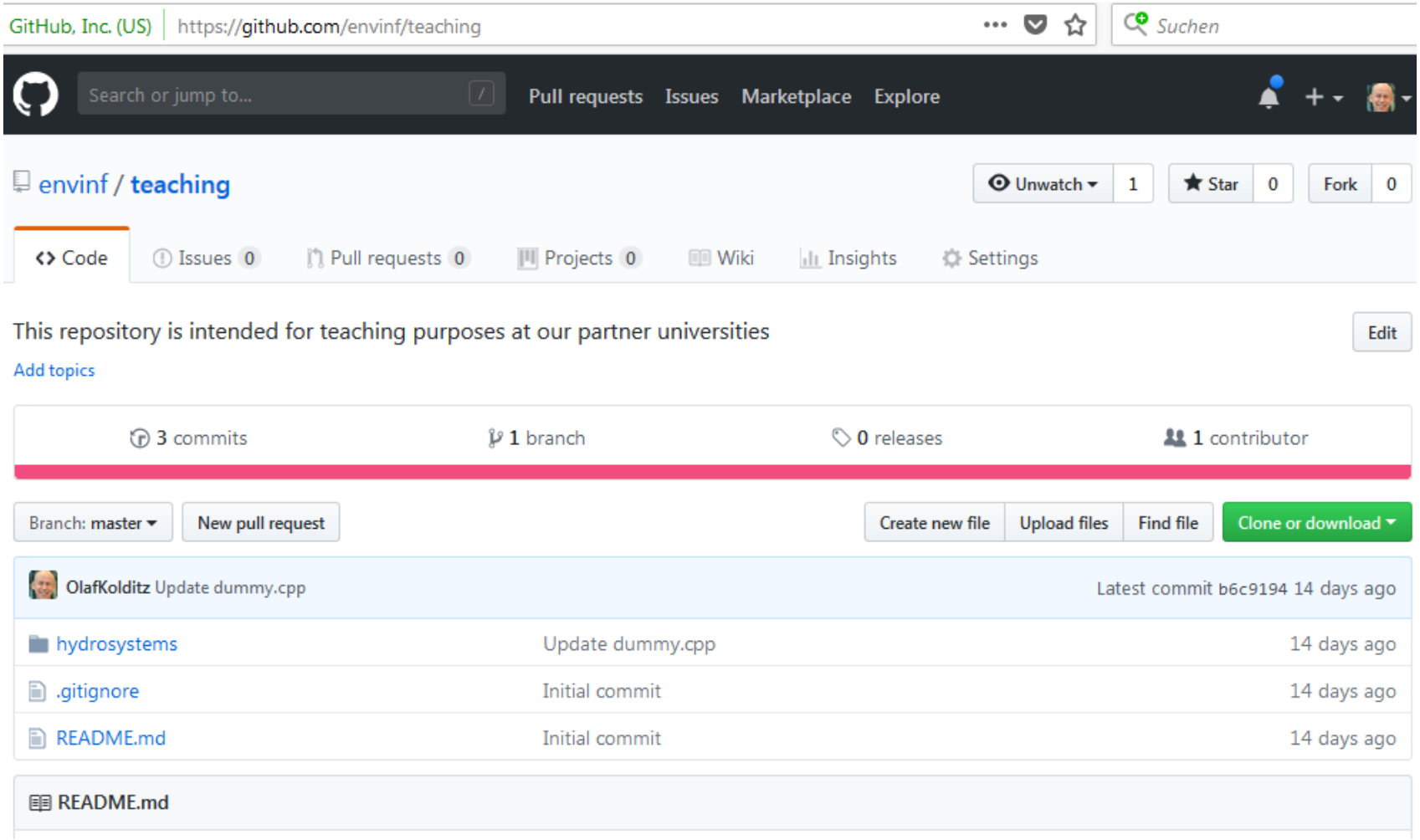
Übung E2 Der Quelltext für diese Übung befindet sich in `EXERCISES`.

TECHNISCHE UNIVERSITÄT DRESDEN

Software-Engineering
GitHub

TECHNISCHE UNIVERSITÄT DRESDEN

GitHub



TECHNISCHE UNIVERSITÄT DRESDEN