Hydroinformatik II - SoSe 2024

HyBHW-S2-01-V06: Einführung in Näherungsverfahren

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, 28.06.2024

TECHNISCHE UNIVERSITÄT DRESDEN

Zeitplan: Hydroinformatik II - SoSe 2024

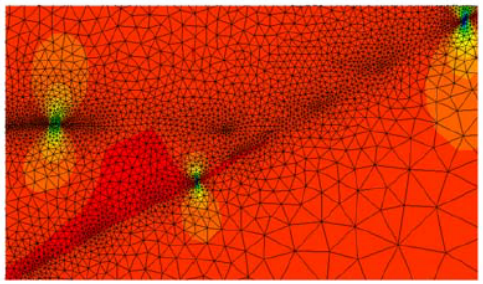
Datum	HI	II	Thema	Typ
14.06.2024	14	2-01	Einführung in die Lehrveranstaltung - Teil 2	L
14.06.2024	15	2-02	Werkzeuge | Tools	L
14.06.2024	16	2-03	Grundlagen: Kontinuumsmechanik	L
21.06.2024	17	2-04	Grundlagen: Hydromechanik	L
21.06.2024	18	2-05	Grundlagen: Partielle Partialgleichungen	L
21.06.2024	19	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

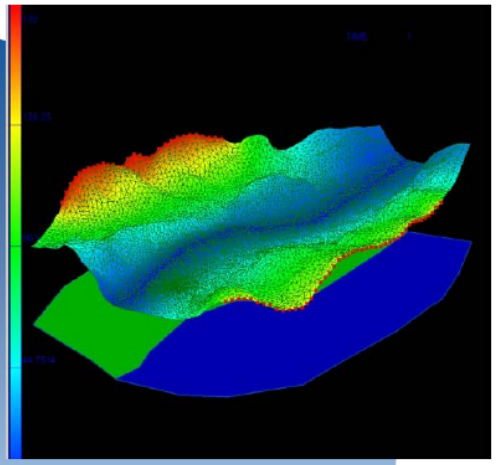
Konzept

$$\frac{d\psi}{dt} = \frac{\partial\psi}{\partial t} + \mathbf{v}^E \nabla\psi$$

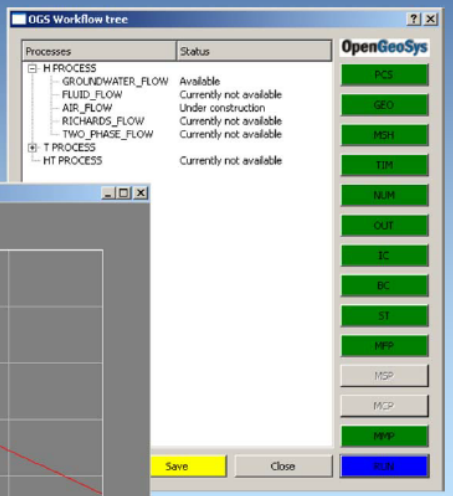


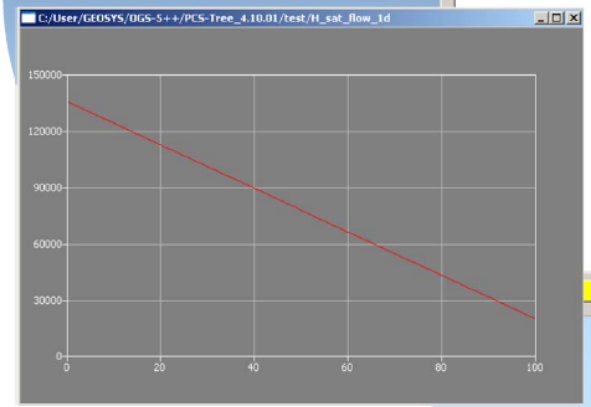
Basics
Mechanik

Anwendung



Numerische
Methoden





Programmierung
Visual C++

Prozessverständnis

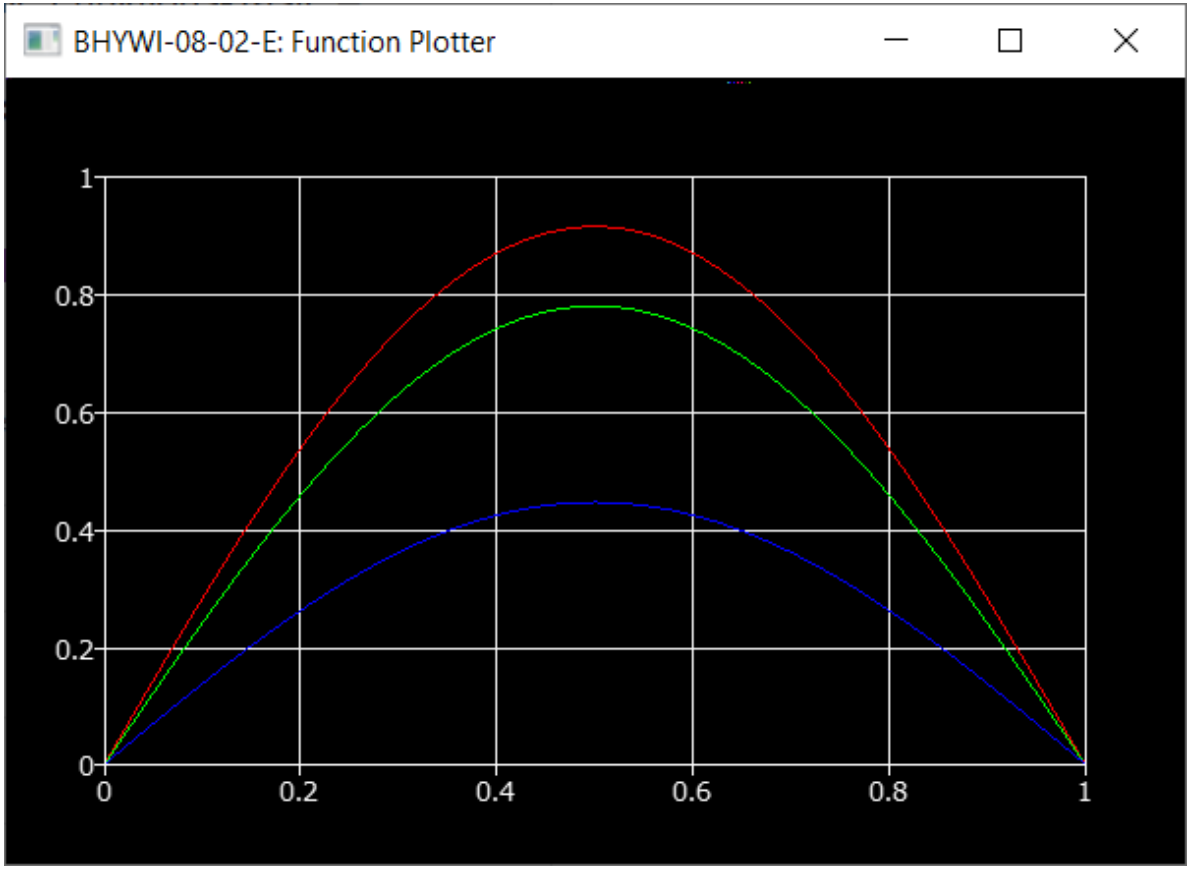
Inhalte

- Übung: BHYWI-08-02-E: Funktionsrechner

- Konzept
- Näherungsverfahren
- Lösungsverfahren
- Definitionen
- Fehler
- Kriterien
- Lösen von Gleichungssystemen

Demo: Funktionsrechner

Übung: BHYWI-08-02-E: Funktionsrechner

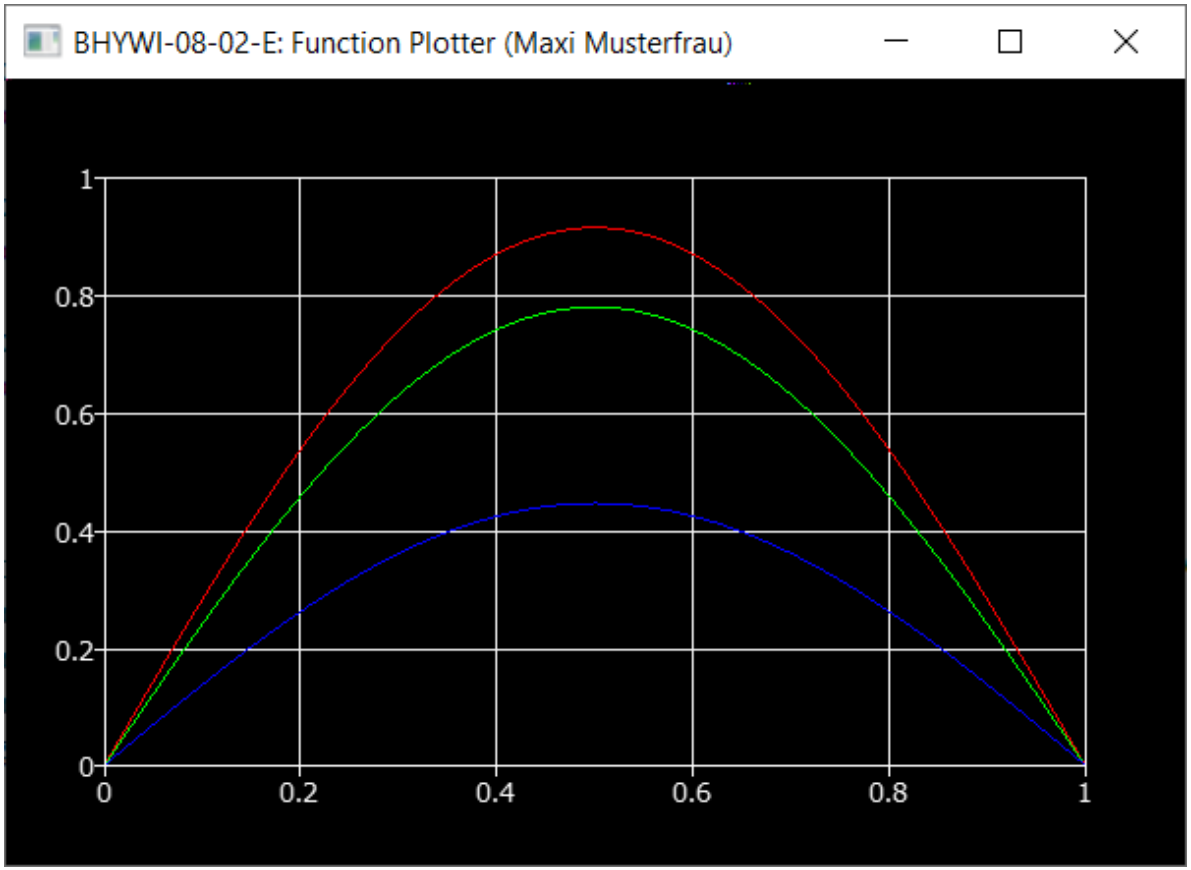


Parabolic equation

$$\frac{\partial \psi}{\partial t} = \alpha \frac{\partial^2 \psi}{\partial x^2} \qquad (1)$$

$$\psi(t, x) = sin(\pi x) exp(-\alpha \pi^2 t) \qquad (2)$$

Hausaufgabe: BHYWI-08-02-E: Funktionsrechner



Parabolic equation

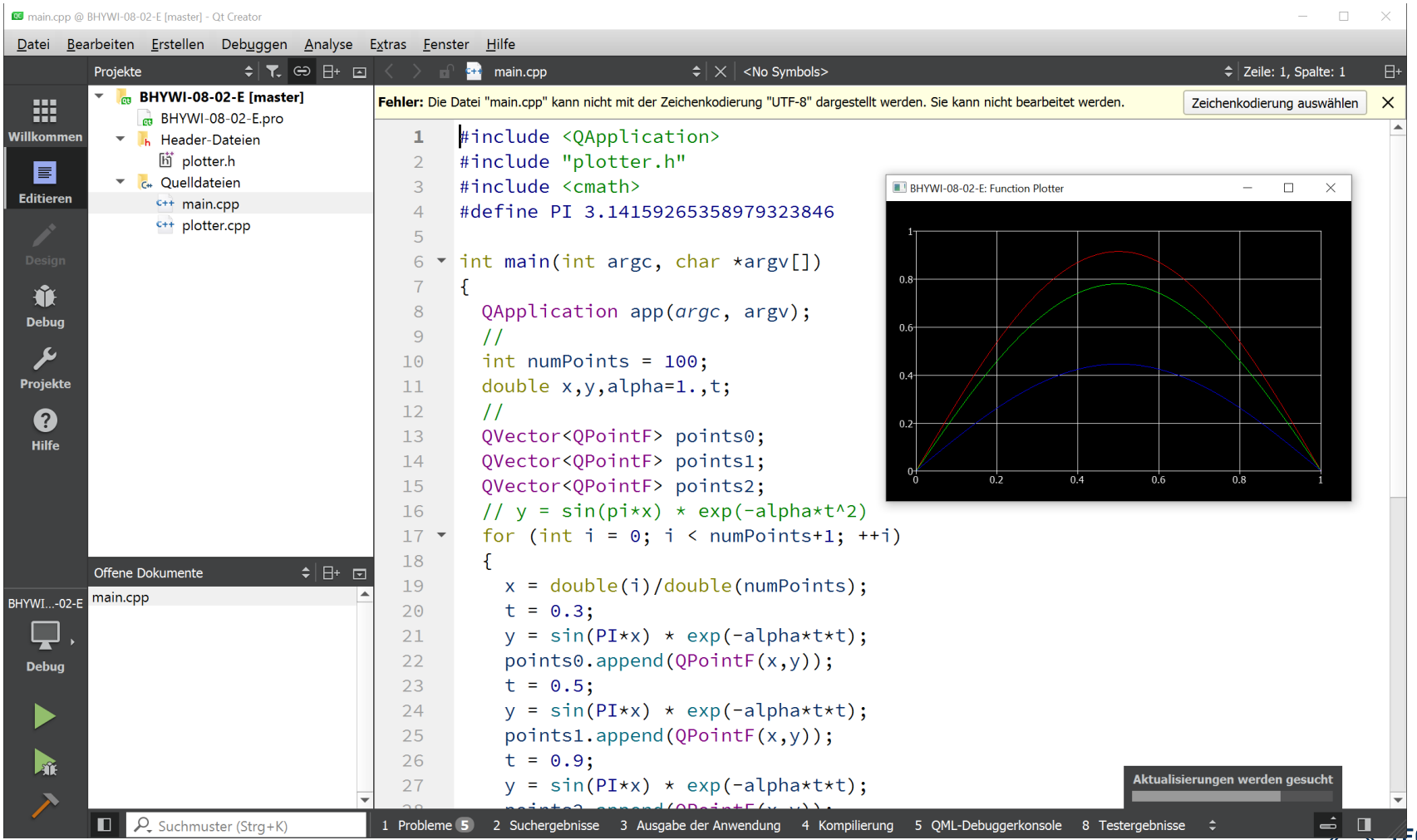
$$\frac{\partial \psi}{\partial t} = \alpha \frac{\partial^2 \psi}{\partial x^2} \qquad (3)$$

$$\psi(t,x) = sin(\frac{\pi}{\sqrt{\alpha}}x)exp(-\pi^2 t) \qquad (4)$$

TECHNISCHE UNIVERSITÄT DRESDEN

Qt: BHYWI-08-02-E: Funktionsrechner

https://github.com/OlafKolditz/Hydroinformatik-II/tree/master/BHYWI-08-02-E



Python: EX06-parabolische-Gleichung

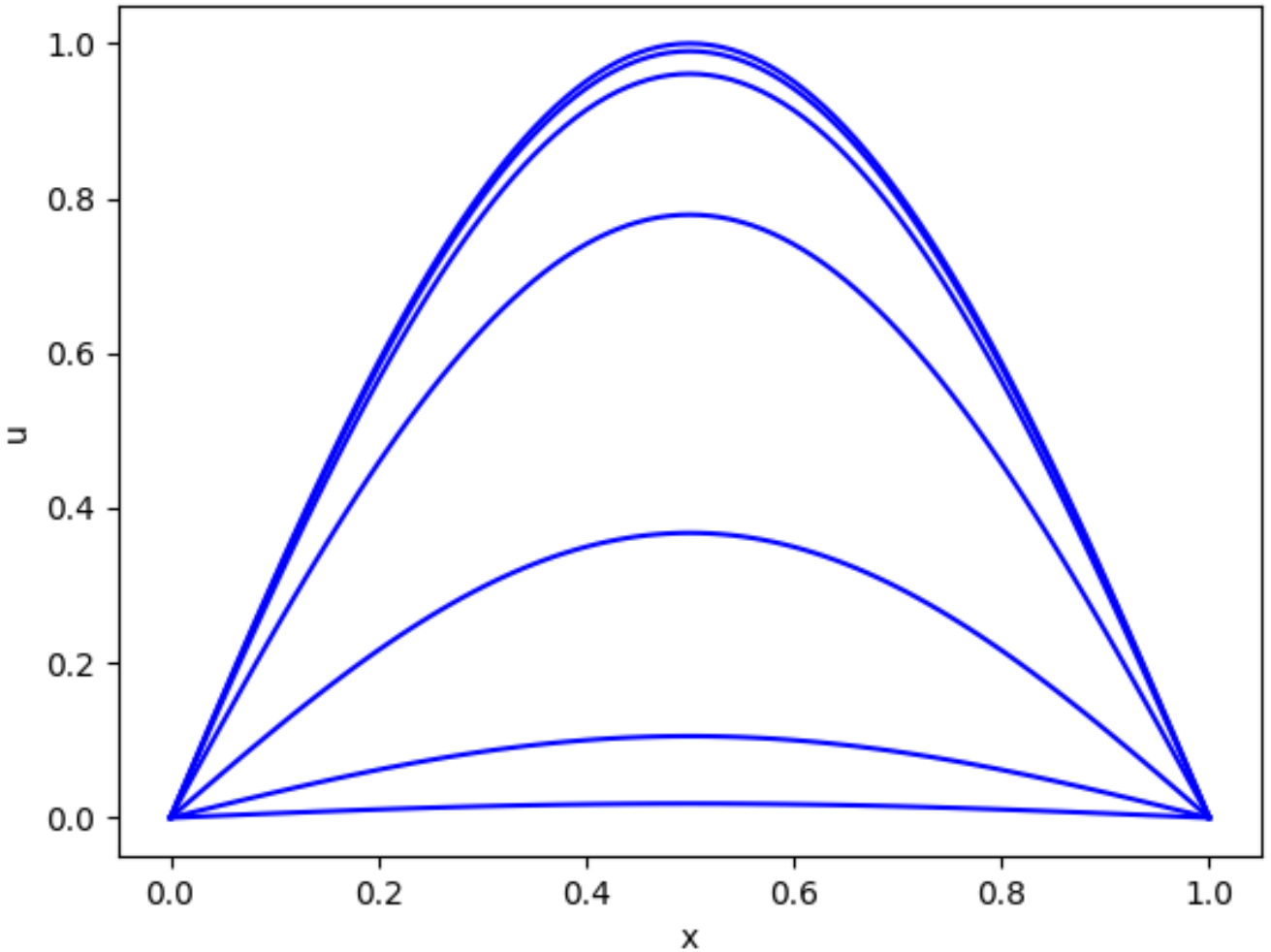
https://github.com/OlafKolditz/Hydroinformatik-II/

```
1  import matplotlib.pylab as plt
2  PI = 3.14159265358979323846
3  numPoints = 100
4  alpha = 1.0
5  t = [0.01,0.1,0.2,0.5,1.0,1.5,2.0]
6  x = []
7  y = []
8  for n in t:
9      for i in range(0,numPoints+1):
10         x.append(float(i)/float(numPoints))
11         #y.append(math.sin(PI*x[i]) * math.exp(-alpha*n*n))
12         y.append(math.sin(math.pi*x[i]) * math.exp(-alpha*n*n))
13     plt.plot(x,y,color='blue')
14     x = []
15     y = []
16 plt.xlabel('x')
17 plt.ylabel('u')
18 plt.axis('tight')
19 plt.savefig("diffusion-equation.png")
20 plt.show()
```

Listing: Python code for parabolic equation

Python: EX06-parabolische-Gleichung: Funktionsrechner

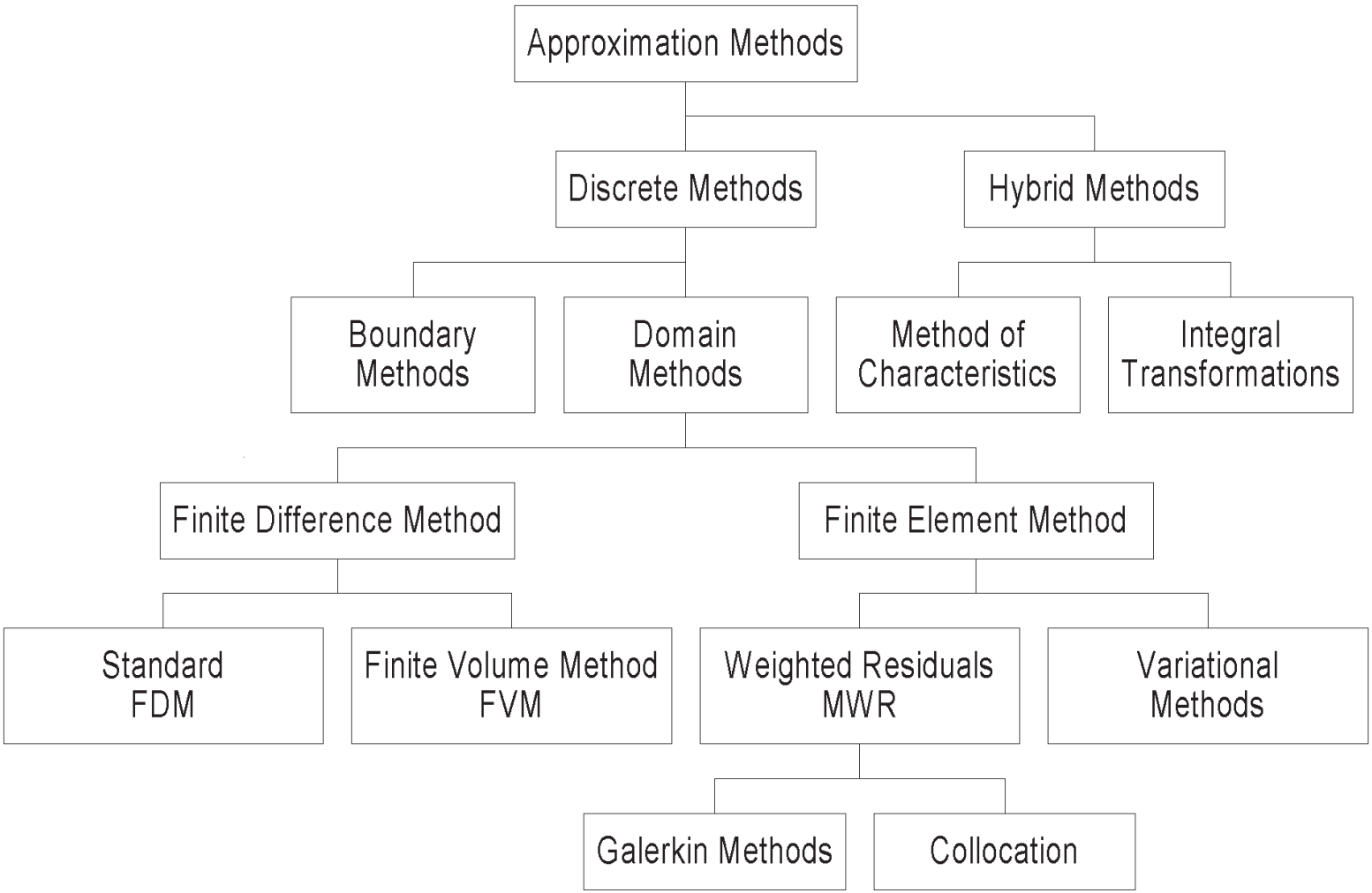
https://github.com/OlafKolditz/Hydroinformatik-II/



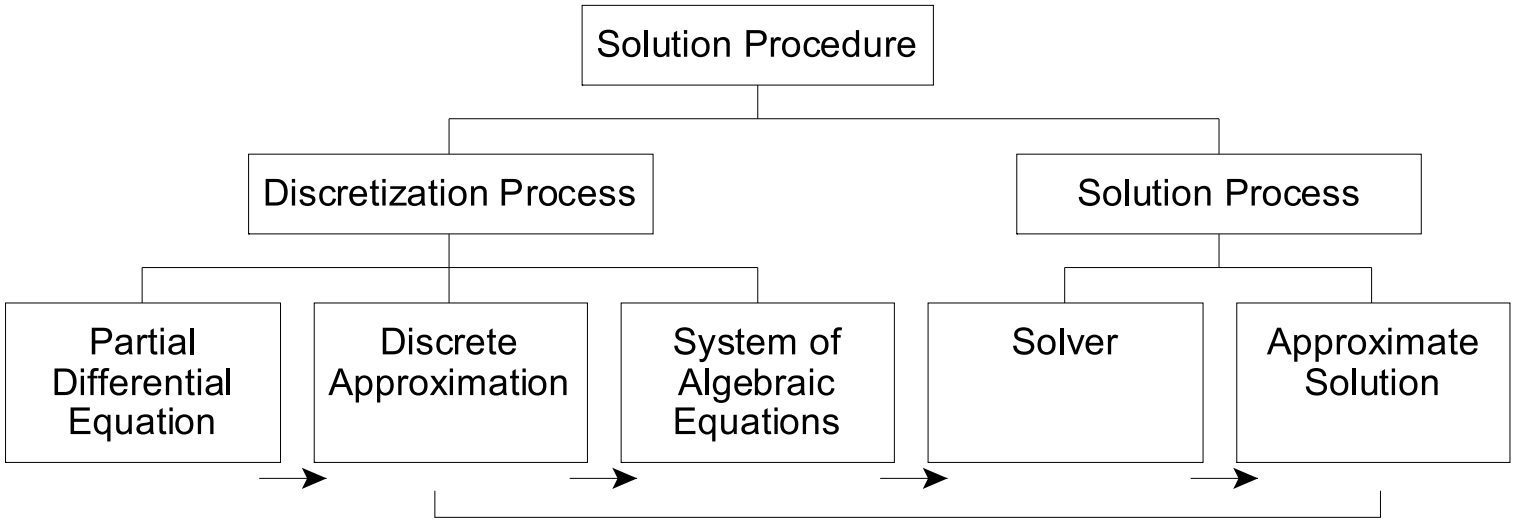
Näherungsverfahren

TECHNISCHE UNIVERSITÄT DRESDEN

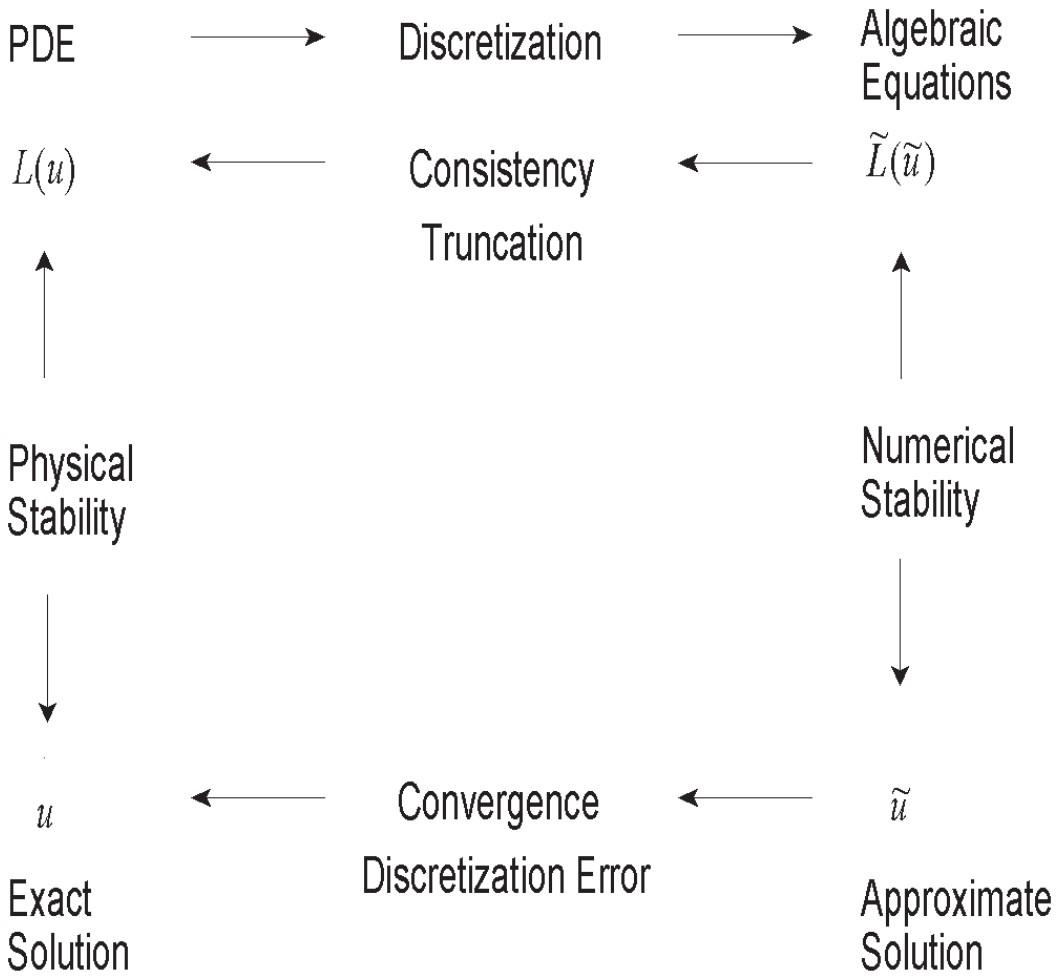
Näherungsverfahren



Lösungsverfahren



Definitionen



Konvergenz

Definition: A solution of the algebraic equations which approximate a given PDE is said to be convergent if the approximate solution approaches the exact solution of the PDE for each value of the independent variable as the grid spacing tends to zero. Thus we require

$$\lim_{\Delta t, \Delta x \to 0} \mid u_j^n - u(t_n, x_j) \mid = 0 \tag{5}$$

Or in other words, the approximate solution converges to the exact one as the grid sizes becomes infinitely small. The difference between exact and approximate solution is the solution error, denoted by

$$\varepsilon_j^n = \mid u_j^n - u(t_n, x_j) \mid \tag{6}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Konsistenz

Definition: The system of algebraic equations (SAE) generated by the discretization process is said to be consistent with the original partial differential equation (PDE) if, in the limit that the grid spacing tends to zero, the SAE is equivalent to the PDE at each grid point. Thus we require

$$\lim_{\Delta t, \Delta x \to 0} \mid \tilde{L}(u_j^n) - L(u[t_n, x_j]) \mid = 0 \tag{7}$$

Stabilität

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} - \alpha\frac{\partial^2 u}{\partial x^2} = 0 \tag{8}$$

- Courant-Zahl

$$Cr = \frac{v\Delta t}{\Delta x} \leqslant 1 \tag{9}$$

- Peclet-Zahl

$$Pe = \frac{v\Delta x}{\alpha} \leqslant 2 \tag{10}$$

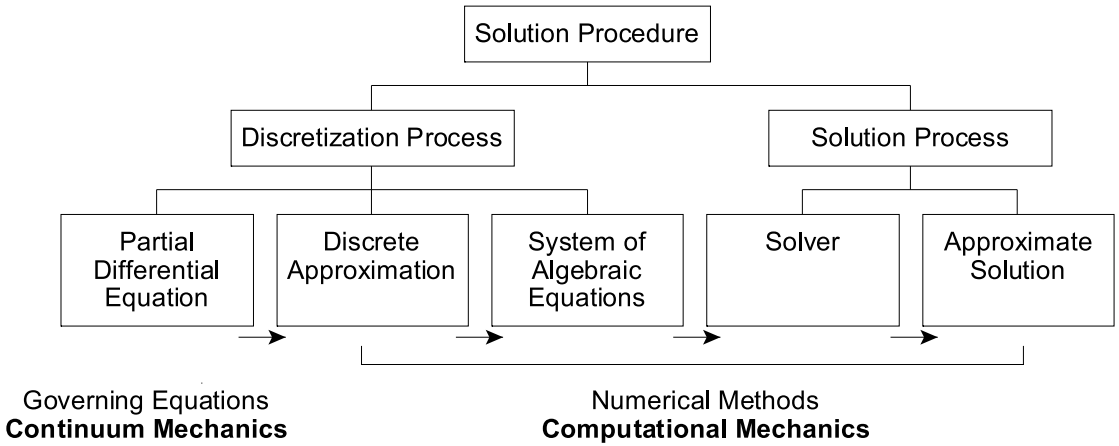
- Neumann-Zahl

$$Ne = \frac{\alpha\Delta t}{\Delta x^2} \leqslant \frac{1}{2} \tag{11}$$

$$0 < Cr^2 < Ne < 1 \tag{12}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Gleichungssysteme



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

Gleichungslöser

The following list reveals an overview on existing methods for solving linear algebraic equation systems.

- Direct methods
  - **Gaussian elimination**
  - Block elimination (to reduce memory requirements for large problems)
  - Cholesky decomposition
  - Frontal solver
- Iterative methods
  - Linear steady methods (Jacobian, **Gauss-Seidel**, Richardson and block iteration methods)
  - Gradient methods (CG) (also denoted as Krylov subspace methods)

Lösen linearer Gleichungen

Application of direct methods to determine the solution of equation

$$\mathbf{x} = \mathbf{A}^{-1}\,\mathbf{b} \tag{14}$$

requires an efficient techniques to invert the system matrix. As a first example we consider the Gaussian elimination technique. If matrix $\mathbf{A}$ is not singular (i.e. $\det \mathbf{A} \neq 0$), can be composed in following way.

$$\mathbf{P}\,\mathbf{A} = \mathbf{L}\,\mathbf{U} \tag{15}$$

with a permutation matrix $\mathbf{P}$ and the lower $\mathbf{L}$ as well as the upper matrices $\mathbf{U}$ in triangle forms.

$$\mathbf{L} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ l_{n1} & \cdots & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & u_{nn} \end{bmatrix} \tag{16}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Gauss-Verfahren (Eliminierungsverfahren)

- 1

$$a_{11}u_1 + a_{12}u_2 = b_1 \tag{17}$$
$$a_{21}u_1 + a_{22}u_2 = b_2 \tag{18}$$

- 2

$$a_{21}\frac{a_{11}}{a_{21}}u_1 + a_{22}\frac{a_{11}}{a_{21}}u_2 = \frac{a_{11}}{a_{21}}b_2 \tag{19}$$

- 3: (19) - (17)

$$\left(\frac{a_{22}a_{11}}{a_{21}} - a_{12}\right)u_2 = \frac{a_{11}}{a_{21}}b_2 - b_1 \tag{20}$$

- 4

$$u_2 = \frac{\frac{a_{11}}{a_{21}}b_2 - b_1}{\frac{a_{22}a_{11}}{a_{21}} - a_{12}} \tag{21}$$

Lösen linearer Gleichungen - Iterative Verfahren

High resolution FEM leads to large equation systems with sparse system matrices. For this type of problems iterative equation solver are much more efficient than direct solvers. Concerning the application of iterative solver we have to distinguish between symmetrical and non-symmetrical system matrices with different solution methods. The efficiency of iterative algorithms, i.e. the reduction of iteration numbers, can be improved by the use of pre-conditioning techniques).

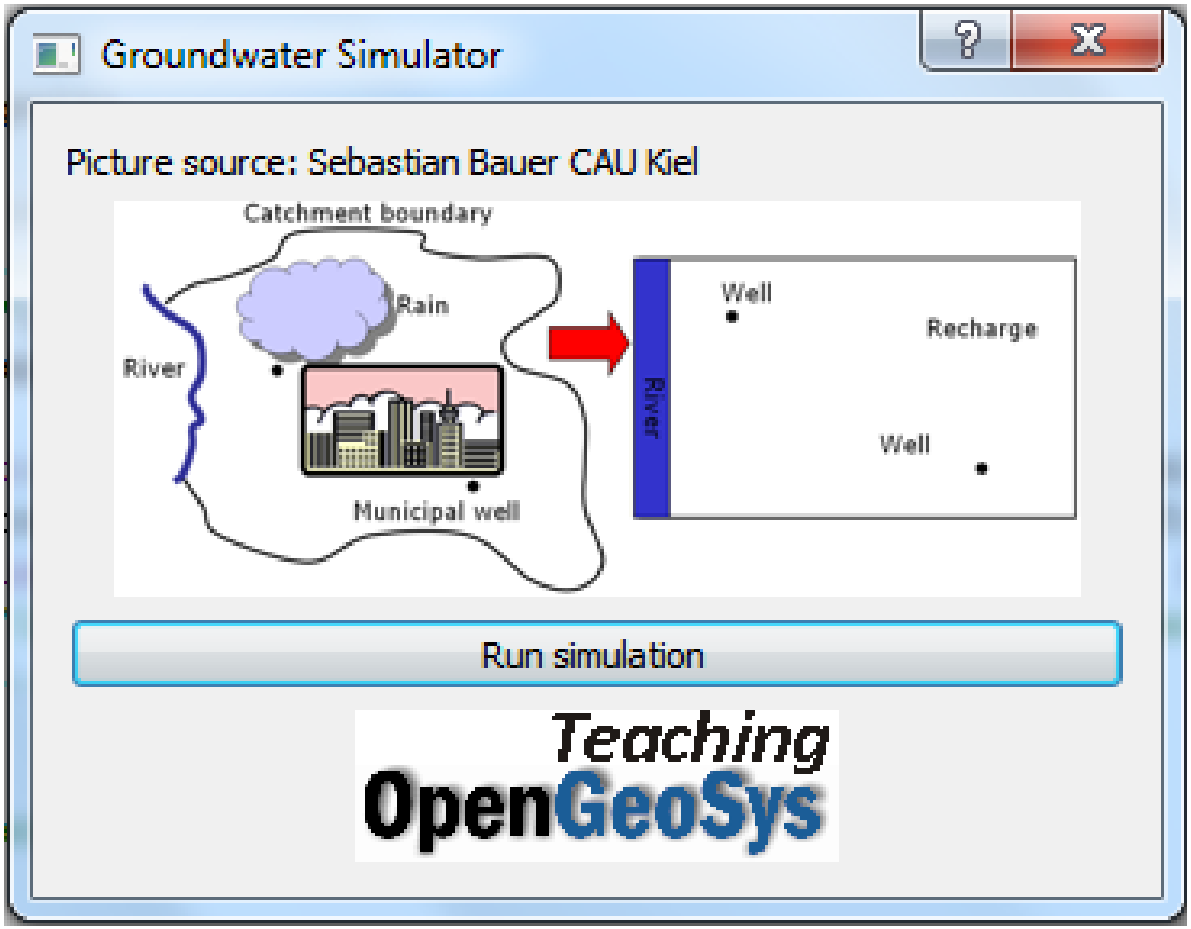
Symmetric Matrices	Non-symmetric Matrices
CG	BiCG
Lanczos	CGStab
Gauss-Seidel, Jacobian, Richards	GMRES
SOR and block-iteration	CGNR

The last two rows of solver for symmetric problems belong to the linear steady iteration methods. The algorithms for solving non-symmetrical systems are also denoted as Krylov subspace methods.

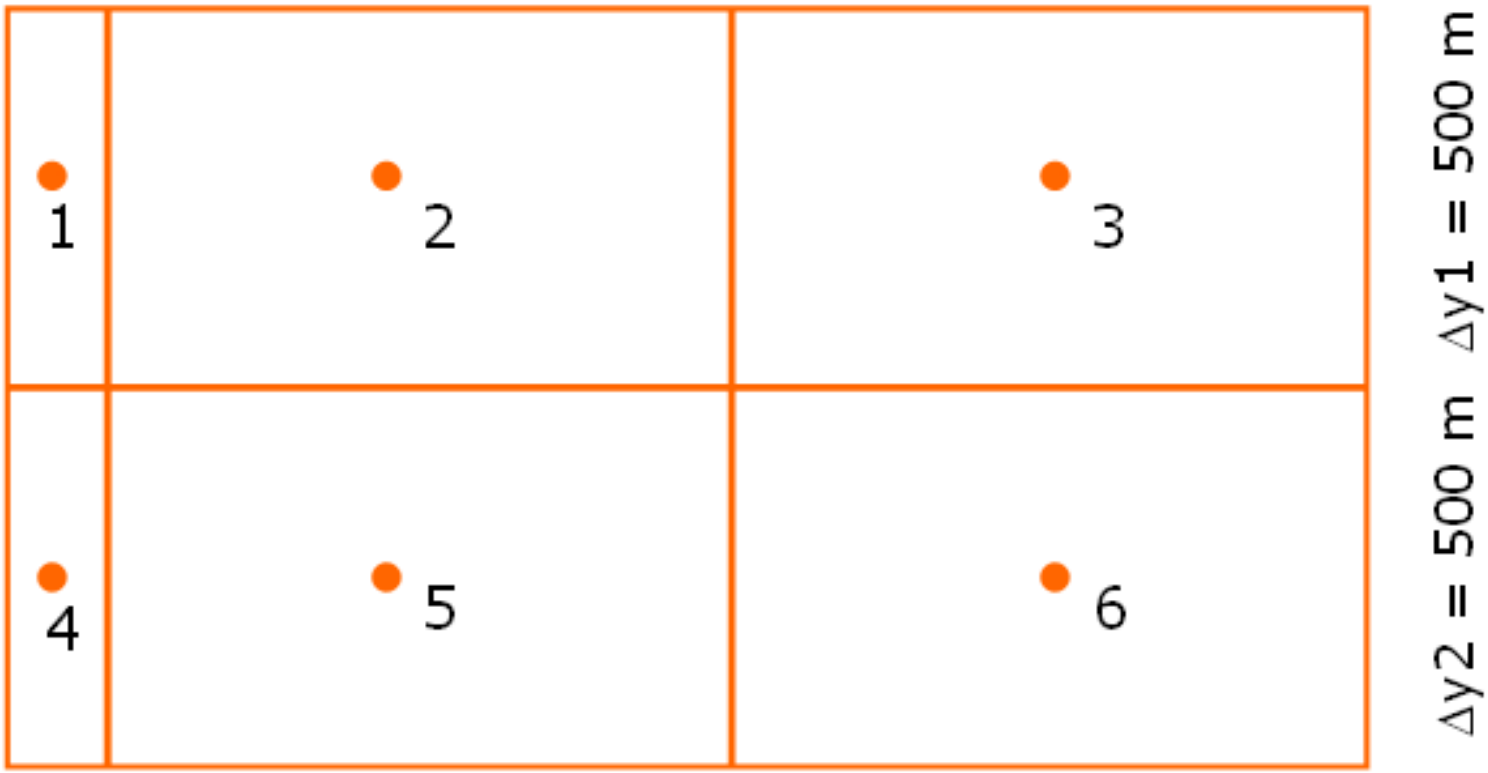
TECHNISCHE UNIVERSITÄT DRESDEN

Lösen linearer Gleichungen - Iterative Verfahren

- Gauss-Seidel Verfahren (Hydrosystemanalyse)



Prinzip-Beispiel



Quelle: Sebastian Bauer (Uni Kiel)

Übung USA#1 - Alternatives Lösungsverfahren

- Gauss-Seidel Verfahren
- Umstellung des Gleichungssystems

$$
\begin{aligned}
h_2 &= 0.2408h_3 + 0.3211h_5 + 4.4181 \\
h_3 &= 0.4285h_2 + 0.5714h_6 + 0.0857 \\
h_5 &= 0.7028h_2 + 0.1054h_6 + 2.0223 \\
h_6 &= 0.8695h_3 + 0.1304h_5
\end{aligned}
$$

TECHNISCHE UNIVERSITÄT DRESDEN

Übung USA#1 - Alternatives Lösungsverfahren

- Konstruktion eines iterativen Lösungsverfahrens
- Pro: Es muss kein Gleichungssystem gelöst werden.
- Con: Es kann auch mal nicht klappen (keine Konvergenz).

$$
\begin{aligned}
h_{2,i+1} &= 0.2408h_{3,i} + 0.3211h_{5,i} + 4.4181 \\
h_{3,i+1} &= 0.4285h_{2,i} + 0.5714h_{6,i} + 0.0857 \\
h_{5,i+1} &= 0.7028h_{2,i} + 0.1054h_{6,i} + 2.0223 \\
h_{6,i+1} &= 0.8695h_{3,i} + 0.1304h_{5,i}
\end{aligned}
$$

TECHNISCHE UNIVERSITÄT DRESDEN

Übung USA#1

```
1  void Dialog::GaussSeidel()
2  {
3    for(int k=0;k<solver_iterations;k++)
4    {
5      x[1] = 0.2408 * x[2] + 0.3211 * x[4] + 4.4181;
6      x[2] = 0.4285 * x[1] + 0.5714 * x[5] + 0.0857;
7      x[4] = 0.7028 * x[1] + 0.1054 * x[5] + 2.0223 ;
8      x[5] = 0.8695 * x[2] + 0.1304 * x[4];
9      TestOutput(x);
10   }
11 }
```

Listing: Implementation of Gauss-Seidel method

Lösen nichtlinearer Gleichungen

In this section we present a description of selected iterative methods that are commonly applied to solve non-linear problems.

- **Picard method** (fixpoint iteration)
- **Newton methods**
- Cord slope method
- Dynamic relaxation method

All methods call for an initial guess of the solution to start but each algorithm uses a different scheme to produce a new (and hopefully closer) estimate to the exact solution. The general idea is to construct a sequence of linear sub-problems which can be solved with ordinary linear solver

Lösen nichtlinearer Gleichungen

The general algorithm of the Picard method can be described as follows. We consider a non-linear equation written in the form

$$\mathbf{A}(\mathbf{x})\,\mathbf{x} - \mathbf{b}(\mathbf{x}) = 0 \tag{22}$$

We start the iteration by assuming an initial guess $\mathbf{x}_0$ and we use this to evaluate the system matrix $\mathbf{A}(\mathbf{x}_0)$ as well as the right-hand-side vector $\mathbf{b}(\mathbf{x}_0)$. Thus this equation becomes linear and it can be solved for the next set of $\mathbf{x}$ values.

$$\begin{aligned} \mathbf{A}(\mathbf{x}_{k-1})\,\mathbf{x}_k - \mathbf{b}(\mathbf{x}_{k-1}) &= 0 \\ \mathbf{x}_k &= \mathbf{A}^{-1}(\mathbf{x}_{k-1})\,\mathbf{b}(\mathbf{x}_{k-1}) \end{aligned} \tag{23}$$

Lösen nichtlinearer Gleichungen

Repeating this procedure we obtain a sequence of successive solutions for $\mathbf{x}_k$. During each iteration loop the system matrix and the right-hand-side vector must be updated with the previous solution. The iteration is performed until satisfactory convergence is achieved. A typical criterion is e.g.

$$\varepsilon \geqslant \frac{\| \mathbf{x}_k - \mathbf{x}_{k-1} \|}{\| \mathbf{x}_k \|} \tag{24}$$

where ε is a user-defined tolerance criterion. For the simple case of a non-linear equation $\mathbf{x} = \mathbf{b}(\mathbf{x})$ (i.e. $\mathbf{A} = \mathbf{I}$), the iteration procedure is graphically illustrated in Fig. 1. To achieve convergence of the scheme it has to be guaranteed that the iteration error

Lösen nichtlinearer Gleichungen

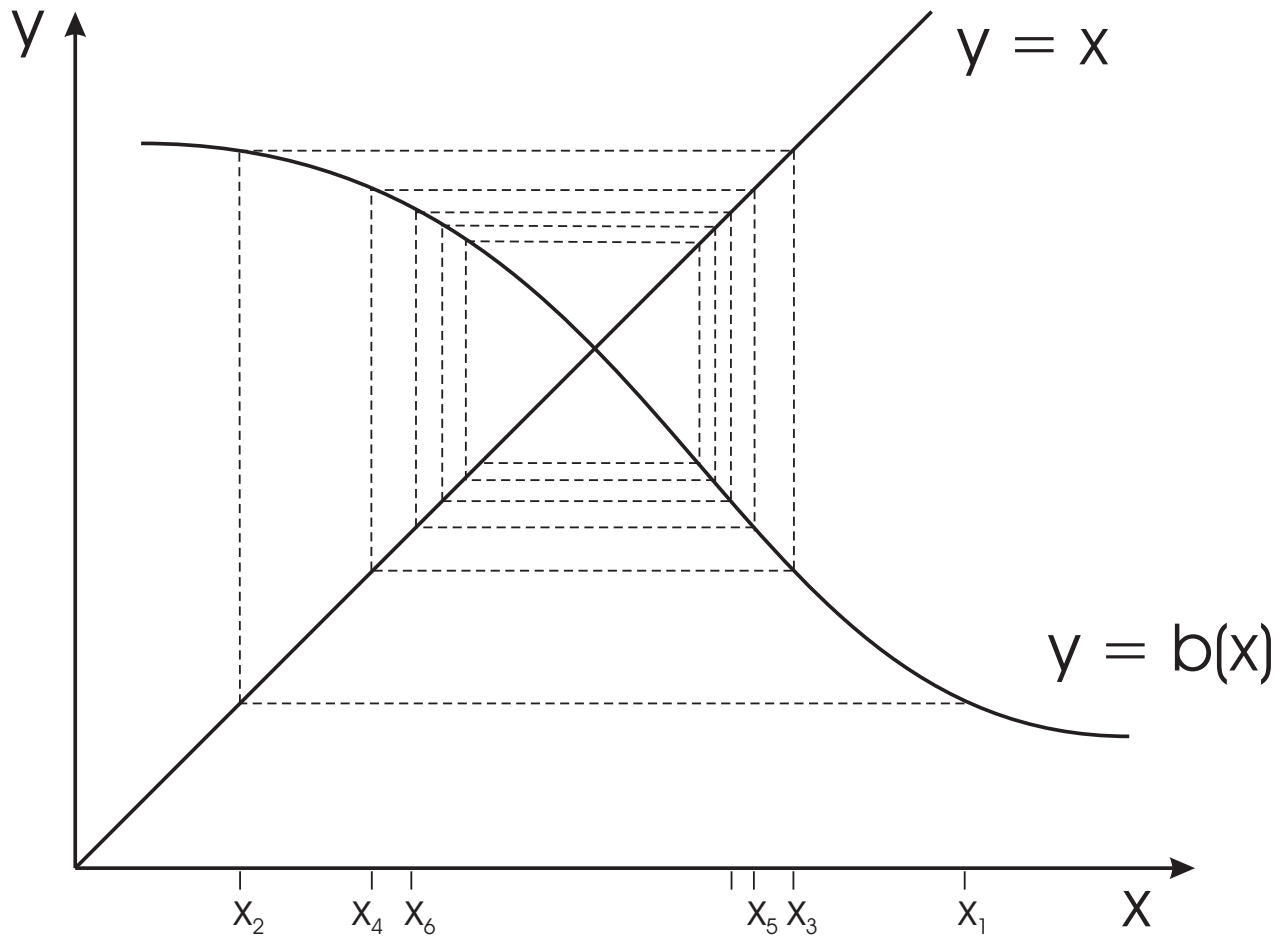


Fig.: Graphical illustration of the Picard iteration method

Lösen nichtlinearer Gleichungen - Newton-Verfahren

In order to improve the convergence order of non-linear iteration methods, i.e. derive higher-order schemes, the Newton-Raphson method can be employed. To describe this approach, we consider once again the non-linear equation

$$\mathbf{R}(\mathbf{x}) = \mathbf{A}(\mathbf{x})\,\mathbf{x} - \mathbf{b}(\mathbf{x}) = 0 \tag{25}$$

Assuming that the residuum $\mathbf{R}(\mathbf{x})$ is a continuous function, we can develop a Taylor series expansion about any known approximate solution $\mathbf{x}_k$.

$$\mathbf{R}_{k+1} = \mathbf{R}_k + \left[\frac{\partial \mathbf{R}}{\partial \mathbf{x}}\right]_k \Delta\mathbf{x}_{k+1} + 0(\Delta\mathbf{x}_{k+1}^2) \tag{26}$$

Lösen nichtlinearer Gleichungen - Newton-Verfahren

Second- and higher-order terms are truncated in the following. The term $\partial \mathbf{R}/\partial \mathbf{x}$ represents tangential slopes of $\mathbf{R}$ with respect to the solution vector and it is denoted as the Jacobian matrix $\mathbf{J}$. As a first approximation we can assume $\mathbf{R}_{k+1} = 0$. Then the solution increment can be immediately calculated from the remaining terms in equation (26).

$$\Delta \mathbf{x}_{k+1} = -\mathbf{J}_k^{-1} \mathbf{R}_k \tag{27}$$

where we have to cope with the inverse of the Jacobian. The iterative approximation of the solution vector can be computed now from the increment.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \mathbf{x}_{k+1} \tag{28}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Lösen nichtlinearer Gleichungen - Newton-Verfahren

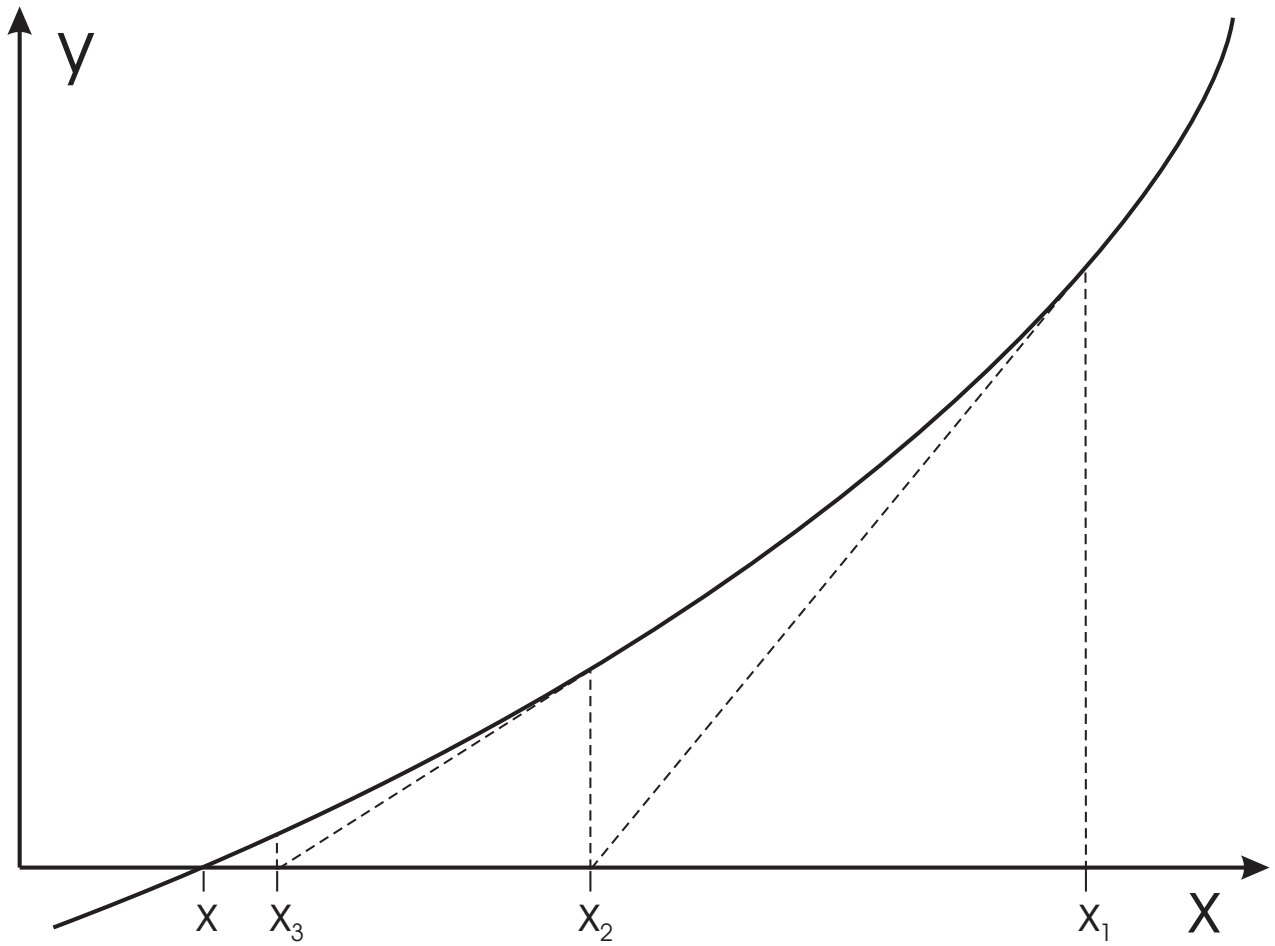


Fig.: Graphical illustration of the Newton-Raphson iteration method

Lösen nichtlinearer Gleichungen - Newton-Verfahren

Once an initial guess is provided, successive solutions of $\mathbf{x}_{k+1}$ can be determined using equations (27) and (28) (Fig. 2). The Jacobian has to re-evaluated and inversed at every iteration step, which is a very time-consuming procedure in fact. At the expense of slower convergence, the initial Jacobian $\mathbf{J}_0$ may be kept and used in the subsequent iterations. Alternatively, the Jacobian can be updated in certain iteration intervals. This procedure is denoted as modified or 'initial slope' Newton method (Fig. 3).
The convergence velocity of the Newton-Raphson method is second-order. It is characterized by the expression.

$$\| \mathbf{x}_{k+1} - \mathbf{x} \| \leqslant C \| \mathbf{x}_k - \mathbf{x} \|^2 \tag{29}$$

TECHNISCHE UNIVERSITÄT DRESDEN

Lösen nichtlinearer Gleichungen - Newton-Verfahren

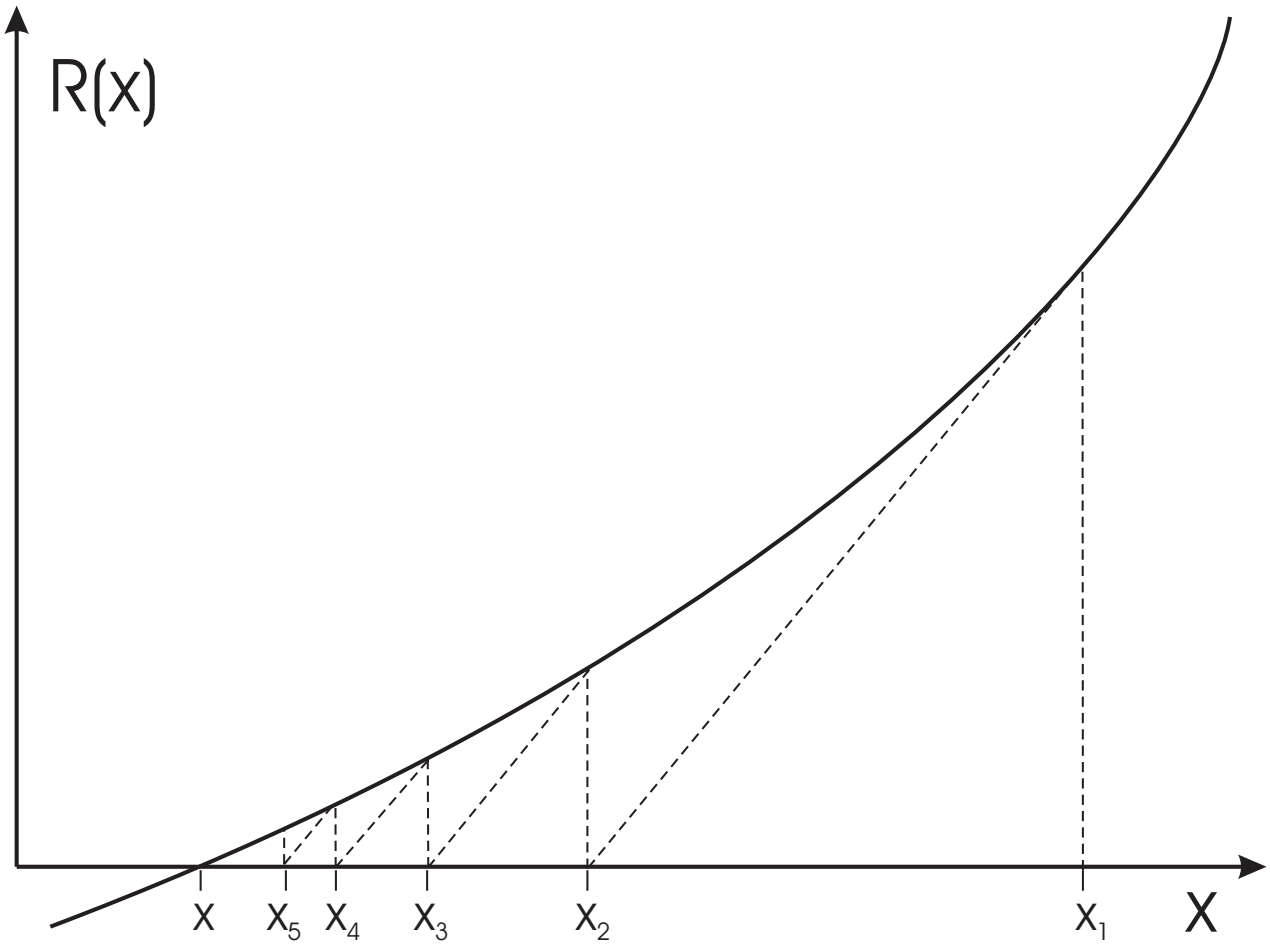


Fig.: Graphical illustration of the modified Newton-Raphson iteration method