

Hydroinformatik II
"Prozesssimulation und Systemanalyse"
HyBHW-1-02-7 @ 2021
Einführung in Näherungsverfahren

Olaf Kolditz

*Helmholtz Centre for Environmental Research – UFZ

¹Technische Universität Dresden – TUDD

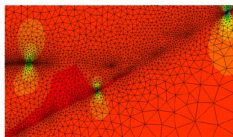
²Centre for Advanced Water Research – CAWR

04.06.2021 - Dresden

Zeitplan: Hydroinformatik II

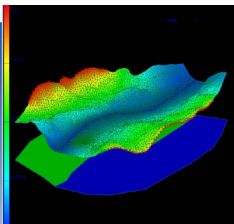
Datum	V	Thema	T
16.04.2021	01	Einführung in die Lehrveranstaltung Tools	L
23.04.2021	02	Grundlagen: Kontinuumsmechanik	L
30.04.2021	03	Grundlagen: Hydromechanik	L
07.05.2021	04	Grundlagen: Partielle Partialgleichungen	L
14.05.2021	05a	Tools: Compiler, Python, Jupyter	E
14.05.2021	05b	Übung: Elliptische PDG	E
21.05.2021	06	Übungen: Übersicht und Werkzeuge	E
28.05.2021	–	Pfingsten	
04.06.2021	07	Grundlagen: Näherungsverfahren	L
11.06.2021	08	Numerik: Finite-Differenzen-Methode (explizit)	L
18.06.2021	09	Numerik: Finite-Differenzen-Methode (implizit)	L
25.06.2021	10	Reserve	L/E
02.07.2021	11	Übung: Diffusionsprozesse	E
09.07.2021	12	Übung: Gerinnehydraulik	E
16.07.2021	13	Übung: Grundwassermodellierung	E
23.07.2021	14	Beleg: Besprechung zur Vorbereitung	L

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

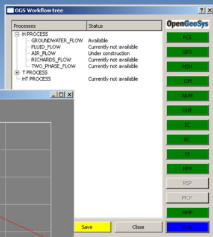


Basics
Mechanik

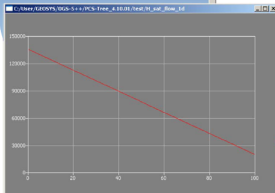
Anwendung



Numerische
Methoden



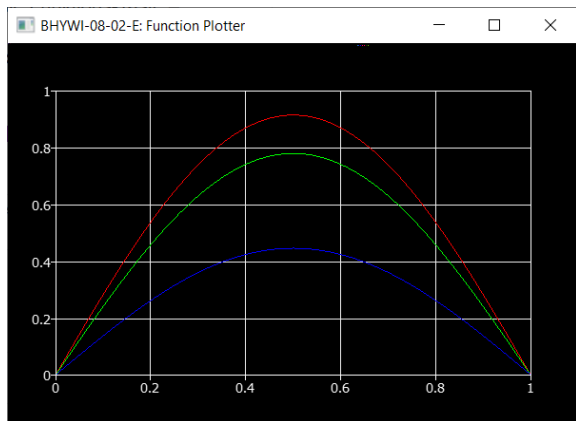
Programmierung
Visual C++



Prozessverständnis

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

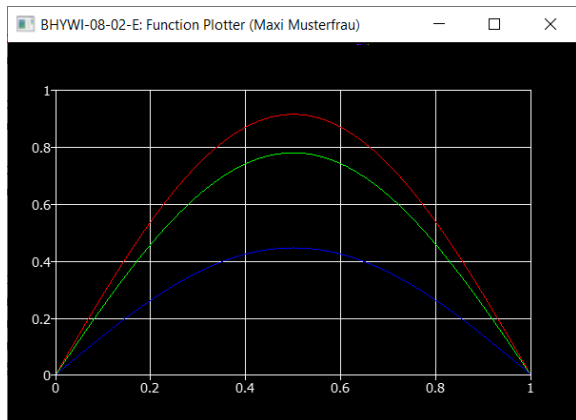
- ▶ Konzept
- ▶ Näherungsverfahren
- ▶ Lösungsverfahren
- ▶ Definitionen
- ▶ Fehler
- ▶ Kriterien
- ▶ Lösen von Gleichungssystemen



Parabolic equation

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

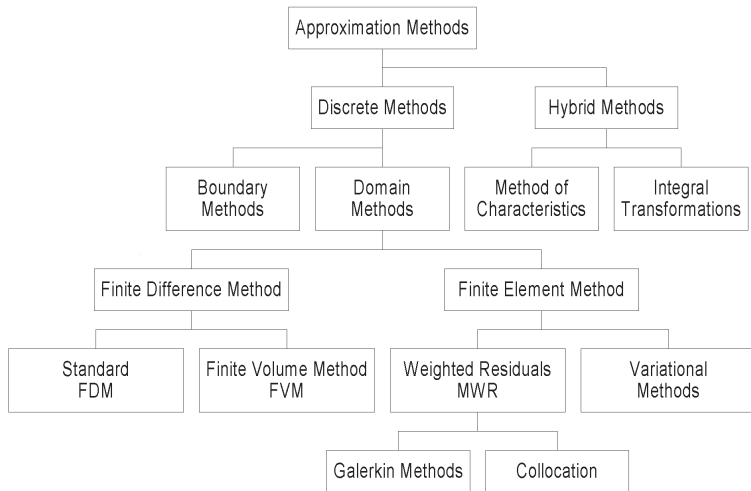
$$\psi(t, x) = \sin(\pi x) \exp(-\alpha \pi^2 t) \quad (2)$$

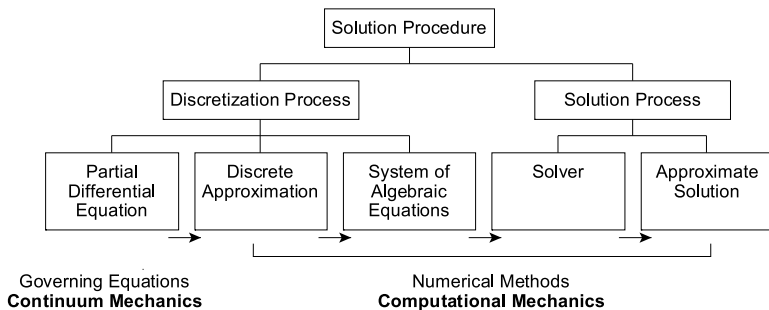


Parabolic equation

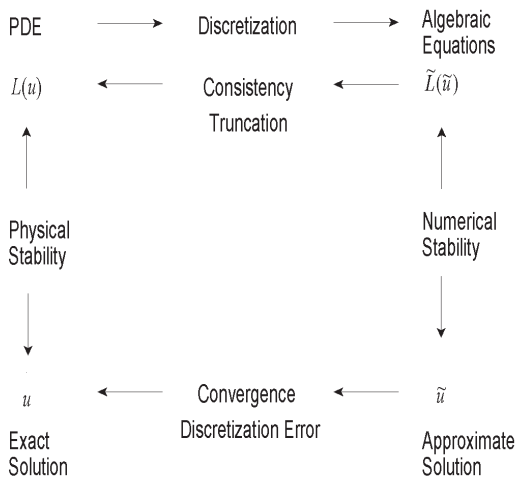
$$\frac{\partial \psi}{\partial t} = \alpha \frac{\partial^2 \psi}{\partial x^2} \quad (3)$$

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





Definitionen



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

$$\lim_{\Delta t, \Delta x \rightarrow 0} |u_j^n - u(t_n, x_j)| = 0 \quad (5)$$

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

$$\varepsilon_j^n = |u_j^n - u(t_n, x_j)| \quad (6)$$

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

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

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

- ▶ Courant-Zahl

$$Cr = \frac{v \Delta t}{\Delta x} \leq 1 \quad (9)$$

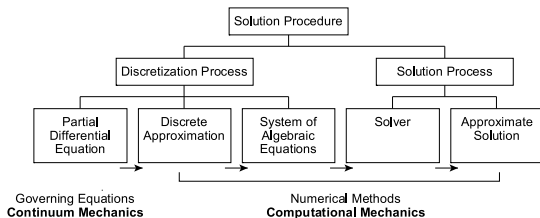
- ▶ Peclet-Zahl

$$Pe = \frac{v \Delta x}{\alpha} \leq 2 \quad (10)$$

- ▶ Neumann-Zahl

$$Ne = \frac{\alpha \Delta t}{\Delta x^2} \leq \frac{1}{2} \quad (11)$$

$$0 < Cr^2 < Ne < 1$$



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

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)

Application of direct methods to determine the solution of equation

$$\mathbf{x} = \mathbf{A}^{-1} \mathbf{b} \quad (14)$$

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.

$$\mathbf{P} \mathbf{A} = \mathbf{L} \mathbf{U} \quad (15)$$

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

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

Gauss-Verfahren (Eliminierungsverfahren)

▶ 1

$$a_{11}u_1 + a_{12}u_2 = b_1 \quad (17)$$

$$a_{21}u_1 + a_{22}u_2 = b_2 \quad (18)$$

▶ 2

$$a_{21} \frac{a_{11}}{a_{21}} u_1 + a_{22} \frac{a_{11}}{a_{21}} u_2 = \frac{a_{11}}{a_{21}} b_2 \quad (19)$$

▶ 3

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

▶ 4

$$u_2 = \frac{\frac{a_{11}}{a_{21}} b_2 - b_1}{\frac{a_{22}a_{11}}{a_{21}} - a_{12}} \quad (21)$$

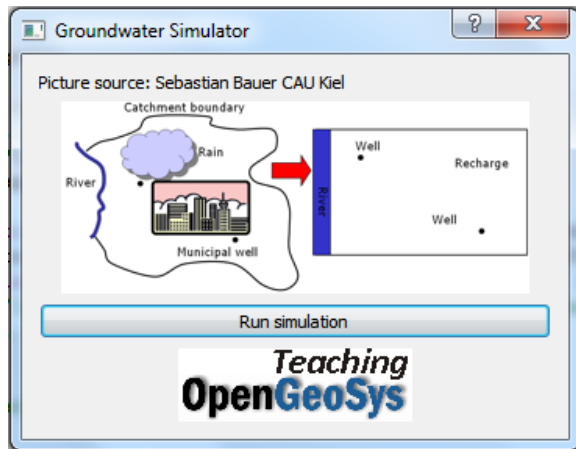
Lösen linearer Gleichungen - Iterative Verfahren

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).

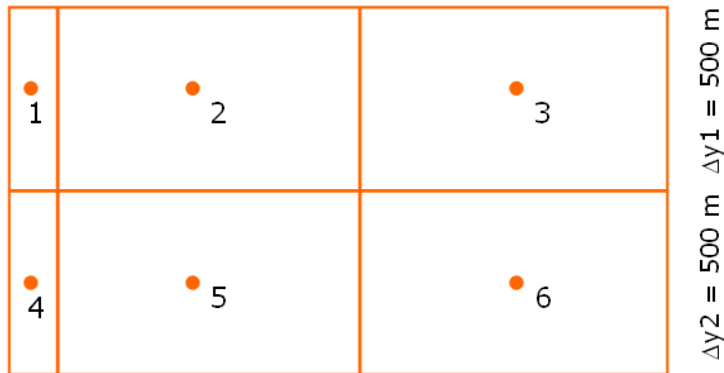
Symmetric Matrices	Non-symmetric Matrices
CG	BiCG
Lanczos	CGStab
Gauss-Seidel, Jacobian, Richards	GMRES
SOR and block-iteration	CGNR

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.

► Gauss-Seidel Verfahren (Hydrosystemanalyse)



Prinzip-Beispiel



$$\Delta x_1 = 100 \text{ m}$$

$$\Delta x_2 = 1000 \text{ m}$$

$$\Delta x_3 = 1000 \text{ m}$$

Quelle: Sebastian Bauer (Uni Kiel)

- ▶ Gauss-Seidel Verfahren
- ▶ Umstellung des Gleichungssystems

$$h_2 = 0.2408h_3 + 0.3211h_5 + 4.4181$$

$$h_3 = 0.4285h_2 + 0.5714h_6 + 0.0857$$

$$h_5 = 0.7028h_2 + 0.1054h_6 + 2.0223$$

$$h_6 = 0.8695h_3 + 0.1304h_5$$

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

$$h_{2,i+1} = 0.2408h_{3,i} + 0.3211h_{5,i} + 4.4181$$

$$h_{3,i+1} = 0.4285h_{2,i} + 0.5714h_{6,i} + 0.0857$$

$$h_{5,i+1} = 0.7028h_{2,i} + 0.1054h_{6,i} + 2.0223$$

$$h_{6,i+1} = 0.8695h_{3,i} + 0.1304h_{5,i}$$

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

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

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

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

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.

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

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.

$$\varepsilon \geq \frac{\|\mathbf{x}_k - \mathbf{x}_{k-1}\|}{\|\mathbf{x}_k\|} \quad (24)$$

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

$$e_k = \| \mathbf{x}_k - \mathbf{x} \| < C \| \mathbf{x}_{k-1} - \mathbf{x} \|^p = e_{k-1} \quad (25)$$

or, alternatively, the distance between successive solutions will reduce

$$\| \mathbf{x}_{k+1} - \mathbf{x}_k \| < \| \mathbf{x}_k - \mathbf{x}_{k-1} \|^p \quad (26)$$

where p denotes the convergence order of the iteration scheme. It can be shown that the iteration error of the Picard method decreases linearly with the error at the previous iteration step. Therefore, the Picard method is a first-order convergence scheme.

Lösen nichtlinearer Gleichungen

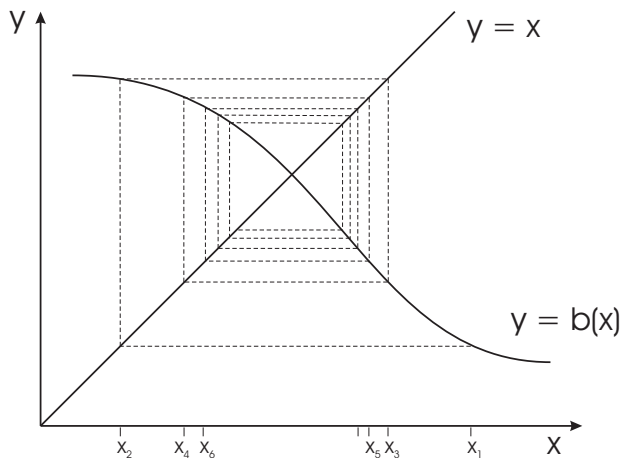


Figure: Graphical illustration of the Picard iteration method

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

$$\mathbf{R}(\mathbf{x}) = \mathbf{A}(\mathbf{x}) \mathbf{x} - \mathbf{b}(\mathbf{x}) = 0 \quad (27)$$

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 .

$$\mathbf{R}_{k+1} = \mathbf{R}_k + \left[\frac{\partial \mathbf{R}}{\partial \mathbf{x}} \right]_k \Delta \mathbf{x}_{k+1} + o(\Delta \mathbf{x}_{k+1}^2) \quad (28)$$

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 (28).

$$\Delta \mathbf{x}_{k+1} = -\mathbf{J}_k^{-1} \mathbf{R}_k \quad (29)$$

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.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \mathbf{x}_{k+1} \quad (30)$$

Lösen nichtlinearer Gleichungen - Newton-Verfahren

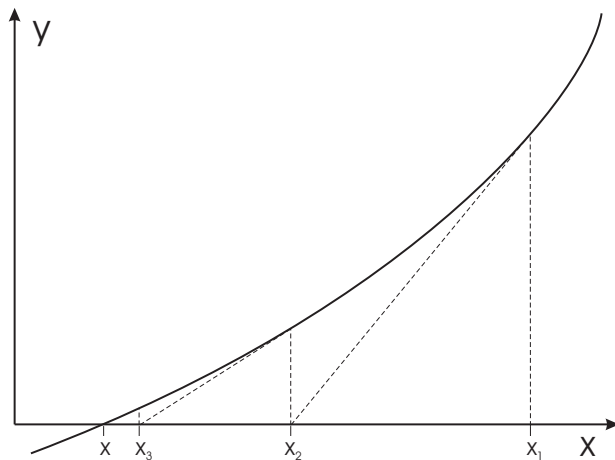


Figure: Graphical illustration of the Newton-Raphson iteration method

Once an initial guess is provided, successive solutions of \mathbf{x}_{k+1} can be determined using equations (29) and (30) (Fig. 2). The Jacobian has to be re-evaluated and inverted 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.

$$\| \mathbf{x}_{k+1} - \mathbf{x} \| \leq C \| \mathbf{x}_k - \mathbf{x} \|^2 \quad (31)$$

Lösen nichtlinearer Gleichungen - Newton-Verfahren

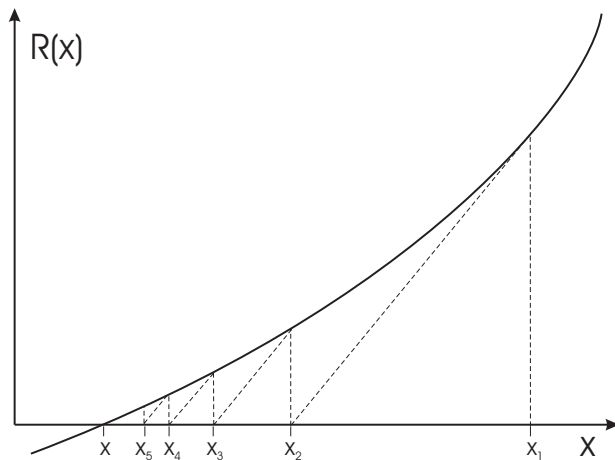


Figure: Graphical illustration of the modified Newton-Raphson iteration method