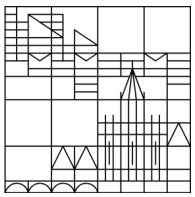
Universität Konstanz
Fachbereich Mathematik und Statsitik
Johannes Schropp
Jan Rohleff

Wintersemester 2024/2025

Universität Konstanz



Numerik partieller Differentialgleichungen I

https://www.math.uni-konstanz.de/~rohleff/numpde1.html

3. Übungsblatt

Ausgabe: 09.01.2025, Abgabe: 16.01.2025

Aufgabe 3.1 (Theorie - 6 Punkte)

Es sei $A \in \mathbb{R}^{N\times N}$ symmetrisch und positiv definit. Die Vektoren $d_0, d_1, \dots, d_{N-1} \in \mathbb{R}^N$ heißen *konjugiert* oder *A-orthogonal*, falls $d_i^T A d_j = 0$ für $i \neq j$, $0 \leq i, j \leq N-1$ und $d_i^T A d_i \neq 0$ für $0 \leq i \leq N-1$.
a) Verifizieren Sie, dass $d_0, d_1, \dots, d_{N-1}$ eine Basis des $\mathbb{R}^N$ bilden.
b) Gegeben sei das Gleichungssystem $Ax = b$. Zeigen Sie: Zerlegt man die Lösung $x = A^{-1}b$ nach der Basis $d_0, \dots, d_{N-1}$, so gilt

$$x \;=\; \sum_{k=0}^{N-1} \alpha_k d_k \quad \text{mit} \quad \alpha_k \;=\; \frac{d_k^T A x}{d_k^T A d_k},\; k = 0, \dots, N-1.$$

c) Zeigen Sie: Für jedes $x_0 \in \mathbb{R}^N$ liefert die durch

$$x_{k+1} \;=\; x_k + \alpha_k d_k \quad \text{mit} \quad \alpha_k \;=\; \frac{-d_k^T (A x_k - b)}{d_k^T A d_k}$$

für $k \geq 0$ erzeugte Folge nach höchstens N Schritten die Lösung $x_N = A^{-1}b$.

Aufgabe 3.2 (Theorie - 6 Punkte)

a) Zu $e > 0$, $e \in \mathbb{R}^N$ wird durch

$$\| u \|_e \;=\; \max \left\{ \frac{|u_i|}{e_i};\; i = 1, \dots, N \right\}$$

eine gewichtete Maximumsnorm auf $\mathbb{R}^N$ erklärt. Es sei nun $P \in \mathbb{R}^{N,N}$. Zeigen Sie: Mit der $\| \cdot \|_e$ zugeordneten Matrixnorm gilt

$$\| P \|_e \;=\; \| Pe \|_e, \;\; \text{falls } P \geq 0.$$

b) Es sei $v \in C^3([-a, a])$ für ein $a > 0$. Zeigen Sie: Für $\mu_0, \mu_1 \in]0, 1]$ und $h < a$ gilt

$$\left| \frac{2}{\mu_0 \mu_1 (\mu_0 + \mu_1) h^2} \Big(\mu_0 v(\mu_1 h) - (\mu_0 + \mu_1) v(0) + \mu_1 v(-\mu_0 h) \Big) - v''(0) \right| \;\leq$$

$$\frac{2h}{3} \max\{ |v^{(3)}(x)|;\; x \in [-a, a] \}$$

Aufgabe 3.3 (Programm - 8 Punkte)

Wir betrachten eine numerisch effiziente Methode zur Lösung von großen linearen Gleichungssystem der Form $Ax = b$, das *konjugierte Gradienten Verfahren* (siehe Algorithmus 1). Dieses stellt insbesondere als *iteratives* Verfahren eine Alternative zu klassischen direkten Lösungsmethoden linearer Gleichungssysteme dar, die z.B. aus der FD-Diskretisierung elliptischer Randwertprobleme resultieren.

Schreiben Sie eine Matlab/Python Funktion

```
function [x,normr,niter] = mycg(A,b,x0,tol)
```

die als Übergabewerte **ausschließlich und in dieser Reihenfolge** die Matrix `A`, die rechte Seite `b`, einen Startwert `x0` und eine Toleranz `tol` akzeptiert. Als Rückgabe erhält man die Lösung `x`, sowie die Gesamtanzahl an Iterationen `niter` und einen Residuumsvektor `normr` der Länge `niter`, der als Einträge die Normen der Residuen pro Iteration enthält.

Betrachten Sie nun hierzu die Problemstellung auf Blatt 8, Aufgabe 3, mit $f(x, y) = 20$ und $g(x, y) = y\cos(4\pi x)$ und lösen Sie das lineare Gleichungssystem, indem Sie das CG-Verfahren in einem `main.m`-File für verschiedene $M = 10, 20, 40, 80$ und Toleranzen $\tau = 10^{-1}, 10^{-2}, 10^{-4}, 10^{-6}$ aufrufen. Wählen Sie zunächst als Startwert den Nullvektor $x_0 = \mathbf{0}$.

1. Plotten Sie für jeden Aufruf die erhaltenen Normen der Residuen im Iterationsverlauf (Hinweis: hier könnte der Plotbefehl `semilogy` hilfreich sein).

2. Schreiben Sie die erhaltene Anzahl an Iterationen in eine Tabelle:

	M			
τ	10	20	40	80
10^{-1}				
10^{-2}				
10^{-4}				
10^{-6}				

3. Lösen Sie das Gleichungssystem $Ax = b$ mittels LU-Zerlegung und dem Backslash-Operator "\". Vergleichen Sie die Differenz der Lösung mit der aus dem CG-Verfahren in der 2-Norm sowie die Rechenzeit unter Verwendung von `tic` und `toc` für verschiedene M und τ. Was beobachten Sie?

4. Was stellen Sie fest, wenn Sie als Startvektor x_0 nicht den Nullvektor verwenden?

Algorithm 1 Conjugate Gradient Method

Require: $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $x^0 \in \mathbb{R}^n$, Tolerance $\tau \geq 0$
Ensure: A is positive definite
  $k = 0$;
  $r^0 = b - Ax^0$;
  $p^0 = r^0$;
  while $\|r_k\| \geq \tau$ **do**
    $\alpha_k = \frac{(p^k)^T r^k}{(p^k)^T A p^k}$;
    $x^{k+1} = x^k + \alpha_k p^k$;
    $r^{k+1} = r^k - \alpha_k A p^k$;
    $\beta_k = \frac{(Ap^k)^T r^{k+1}}{(Ap^k)^T p^k}$;
    $p^{k+1} = r^{k+1} - \beta_k p^k$;
    $k = k + 1$;
  end while