Universität Konstanz
FB Mathematik & Statistik
Prof. Dr. M. Junk
Dr. M. Rheinländer

Ausgabe: 06. Dez., WS 07/08

# Numerik von Gleichungssystemen

**5. Übungsblatt**

## Aufgabe 5.1: Nachtrag zum Jacobi-Verfahren

a) Es sei $J$ die Iterationsmatrix des Jacobi-Verfahrens zur Lösung des Gleichungssystems $Ax = b$. Falls $A$ *block-tridiagonal* ist mit invertierbaren Diagonalblöcken, ist das Spektrum von $J$ symmetrisch bezüglich Spiegelung am Nullpunkt, d.h. $\mathrm{spec}(J) = -\mathrm{spec}(J)$.

b) Bestimmen Sie für das relaxierte Jacobi-Verfahren einen optimalen Wert für den Parameter $\omega$ unter der Annahme, daß die Iterationsmatrix des Jacobi-Verfahrens ausschließlich reelle Eigenwerte besitzt.

## Aufgabe 5.2: "Ohne Glättung keine Rettung"

Es sei $V_h$ ein endlichdimensionaler Vektorraum (mit Skalarprodukt); ferner besitze der lineare Operator $A_h : V_h \to V_h$ eine Quadratwurzel, d.h. es existiert ein Operator $A_h^{1/2} : V_h \to V_h$ mit $A_h = A_h^{1/2} A_h^{1/2}$. Konkret mag man sich $A$ z.B. durch eine symmetrische, positiv definite Matrix repräsentiert denken. Wir betrachten nun die Gleichung

$$A_h(x_h + c_h) = b_h \qquad b_h, c_h, x_h \in V_h. \tag{1}$$

Dabei stellt $x_h$ eine bekannte approximative Lösung von $Az = b$ dar, die durch den unbekannten Korrekturterm $c_h$ verbessert werden soll, so daß (1) gilt. Offensichtlich ergibt sich $c_h$ als Lösung von

$$A_h c_h = b_h - A_h x_h =: r_h, \tag{2}$$

wobei das Residuum $r_h$ bzw. der Defekt als rechte Seite fungiert.
Um das aufwendige Invertieren von $A_h$ zu vermeiden, möchte man (2) durch ein "gröberes" und somit leichter zu invertierendes System ersetzen. Dazu wählt man einen Vektorraum $V_H$, $\dim V_H < \dim V_h$, ausgestattet mit dem Prolongationsoperator $P : V_H \to V_h$ bzw. dem Restriktionsoperator $R : V_h \to V_H$. Setzt man $r_H := Rr_h$ und $A_H := RA_hP : V_H \to V_H$, so ergibt sich folgenden Gleichung

$$A_H c_H = r_H.$$

Unter Annahme der Invertierbarkeit von $A_H$, wird man erwarten, daß $\tilde{c}_h := Pc_H$ zwar nicht die genaue aber doch eine ungefähre Korrektur liefert. Diese Vorgehensweise läßt sich wiederholen und motiviert folgende Iteration:

$$x_h(n+1) = (I_h - PA_H^{-1}RA_h)x_h(n) + PA_H^{-1}Rb_h.$$

Es stellt sich die Frage, ob $x_h(n)$ für $n \to \infty$ gegen die exakte Lösung des Gleichungssystems konvergiert.

Zeigen Sie, daß die Iterationsmatrix nur die Eigenwerte 0 und 1 annehmen kann, d.h.

$$\mathrm{spec}(I_h - PA_H^{-1}RA_h) \subset \{0, 1\},$$

und deshalb die Konvergenz des Verfahrens für beliebige $b_h \in V_h$ ausgeschlossen ist.

## Aufgabe 5.3: "Vorsicht geboten!"

Es sei $A_h x_h = b_h$ ein Gleichungssystem (genauer eine durch die Gitterweite $h$ parametrisierte Familie von Gleichungssystemen), welches aus einer finiten Differenzen Diskretisierung hervorgeht. Die Elemente der Matrix $A_h$ bzw. des Vektors $b_h$ seien von der Größenordnung $\mathrm{O}(h^{-2})$. Um große Zahlen zu vermeiden, mag es sinnvoll erscheinen, das skalierte Gleichungssystem $h^2 A_h x_h = h^2 b_h$ zu betrachten, so daß sich die Einträge in $A_h$ und $b_h$ wie $\mathrm{O}(1)$ verhalten.
Klären Sie, ob diese Vorgehensweise im Zusammenhang mit den klassischen iterativen Lösungsverfahren (Jacobi, Gauß-Seidel) und dem Mehrgitterverfahren wirklich ratsam ist und geben Sie Hinweise, was möglicherweise beachtet werden sollte.

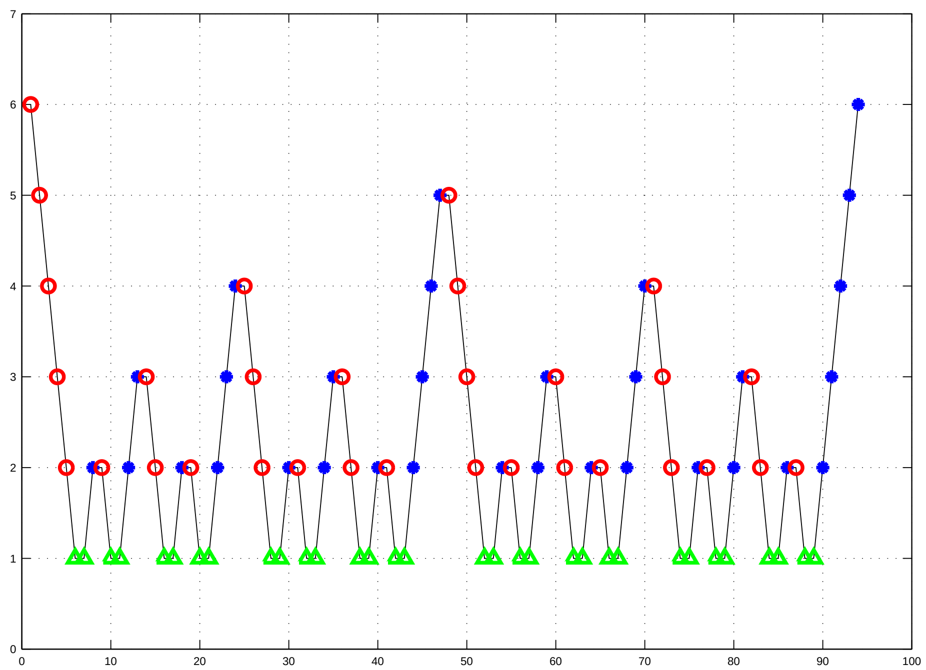
**Aufgabe 5.4: Zur Implementierung des Mehrgitteralgorithmus**

Die folgenden Aufgaben sollen dazu dienen, sich noch einmal etwas behutsamer an eine Matlab-Implementierung des Mehrgitteralgorithmus für unser Modellproblem heranzutasten, da Aufgabe 4.3 doch etwas happig war.

a) Versuchen Sie den zweidimensionalen Interpolationsoperator (auf der bilinearen Interpolation basierend) für die Prolongation als Verkettung von eindimensionalen Interpolationsoperatoren darzustellen. Spielt es eine Rolle, ob man erst in $x$- und dann in $y$-Richtung interpoliert oder umgekehrt?

b) Schreiben Sie zwei Matlab Programme, welche die Prolongationsmatrix automatisch erzeugen (ohne `for`-Schleife). Tragen Sie die Matrixelemente für das erste Programm mit dem Matlab-Kommando `sub2ind` direkt in die Prolongationsmatrix ein; verwenden Sie im zweiten Programm zur Berechnung der Indizes den Matlab-Befehl `find` und addieren sie zur zunächst leeren Prolongationsmatrix mit `sparse` erzeugte Matrizen. Vergleichen Sie die Laufzeiten bei gleicher Gittergröße. (Falls bereits vorhanden vergleichen Sie auch mit einem auf einer `for`-Schleife basierenden Programm).

c) Schreiben Sie eine “Mal”-Funktion, welche z.B. das folgende Bild eines W-Zyklus erzeugt:




Sie können die Funktion entweder mittels `for`-Schleifen definieren oder – eleganter – mittels zweier rekursiver Aufrufe (d.h. die Mal-Funktion ruft sich selbst zweimal hintereinander auf). Die Anzahl der Level (hier 6) sollte als Argument übergeben werden.