Serie 4

1. Gegeben sei der Vektor $b = (1, 1, 1)^T$ und die Matrizen

i) $A = \begin{pmatrix} 2 & -1 & -10 \\ -1 & 2 & 0 \\ -10 & 0 & 2 \end{pmatrix}$

ii) $A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$

Man möchte das lineare Gleichungssystem $Ax = b$ für i), ii) mit der Methode der Konjugierten Gradienten (CG) lösen.

a) Was sind die Vorausssetzungen an A, damit CG anwendbar ist? Ist CG in den Fällen i), ii), anwendbar? (Begründung!)

b) Falls CG anwendbar ist in i), ii), führen Sie einen Schritt durch.

c) Würde exakt gerechnet, nach wievielen Schritten liefert CG spätestens die Lösung von $Ax = b$?

2. Gegeben sei die Matrix A aus Aufgabe 3 mit $k = 100$ und Kondition $\varkappa(A, ||\cdot||_2) = \sin^2(\pi(k-2)/(2(k-1)))/\sin^2(\pi/(2(k-1))) \approx 3971.53$ (in der 2-Norm). Bestimmen Sie eine obere Schranke für die Anzahl Iterationen des CG-Algorithmus, so dass der (absolute) Anfangsfehler $||e^0||_A$ (in der A-Norm) um den Faktor 10^{-7} reduziert wird.

3. (**PA**) Konjugierte Gradienten (**C**onjugate **G**radients) zur Lösung von $Ax + b = 0$.

a) Implementieren Sie in MATLAB das CG-Verfahren (ohne Vorkonditionierung).

b) Testen Sie das Programm mit dem Modellproblem aus Serie 3 für $k = 40$ und $k = 100$ mit Toleranz = 10^{-7} : verwenden Sie die files `sparsemodel.m` und `rhs.m` von der Homepage.
Bestimmen Sie die Anzahl Iterationen und die CPU-Zeit (MATLAB-Funktionen `tic` und `toc`)

Bitte wenden!

c) Wenden Sie die Matlab-Funktion `pcg(A,b,TOL, MAXIT)` mit `TOL`$= 10^{-7}$ und `MAXIT`$= 1000$ (maximale Anzahl Iterationen) auf das Modellproblem mit $k = 40$ und $k = 100$ (d.h. dim(A) $= (k-2)^2$) an. Bestimmen Sie die Anzahl Iterationen und die CPU-Zeit sowie die Kondition der Matrix A mit `condest(A)` (eine untere Schranke für $\varkappa(A, ||\cdot||_1)$ (in der 1-Norm)).
Hinweise: `help pcg` erklärt obigen Aufruf und denjenigen in Aufgabenteil d)! Beachten Sie, dass `pcg` $Ax = b$ (und nicht $Ax + b = 0$) löst.
Bemerkung: `pcg(A,b,TOL,MAXIT)` ist das CG-Verfahren ohne Vorkonditionierung.

d) Wiederholen Sie c) mit der Vorkonditionierungs-Matrix $C = HH^T$, wobei H aus der unvollständigen Cholesky-Zerlegung von A stammt (MATLAB-Befehle `R=cholinc(A,10^-3); H=R';` und `pcg(A,b,TOL,MAXIT, H, H');`).
Bemerkung: `pcg(A,b,TOL,MAXIT,H,H')` ist das CG-Verfahren mit Vorkonditionierung.

e) `sparsemodel(k)` erzeugt eine dünnbesetzte (`sparse`) Matrix. Das `sparse`-Format ermöglicht es, lineare Gleichungssysteme zu lösen, die sonst wegen Speicherproblemen nicht zu behandeln wären. Versuchen Sie die maximale Grösse einer `sparse`- und nicht `sparse`-Matrix abzuschätzen, die Ihr Computer speichern kann.
Hinweis: gehen Sie folgendermassen vor:

1. `sparse`-Format
   a) Löschen Sie alle MATLAB-Variabeln mit `clear`.
   b) Erzeugen Sie die Matrix mit `A=sparsemodel(k)`.
   c) Benutzen Sie `whos` um die Speicherbelegung zu sehen.
   d) Falls keine `Out of memory` Meldung erscheint, wählen Sie ein grösseres k, und wiederholen Sie die Prozedur.
2. volles Format
   a) Löschen Sie alle MATLAB-Variabeln mit `clear`.
   b) Erzeugen Sie die Matrix im normalen Format mit `A=model(k)`.
   c) Fahren Sie wie fort für das `sparse`-Format.

Abgabe: Mittwoch 10. Mai 2006 in der Uebungsstunde.