Numerische Mathematik für StudentInnen
mit Ziel Diplom in Mathematik

Gerhard Opfer — WS 1997/1998 (1. Teil) — SS 1998 (2. Teil)

Bogen 19: Übungen am 26. Mai 1998

Lösungshinweise

Aufgabe P55. (a) ja, (b) i. a. nein, denn $\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} = \mathbf{x}^{\mathrm{T}}\frac{1}{2}(\mathbf{A}+\mathbf{A}^{\mathrm{T}})\mathbf{x}$ und $\mathbf{A} \neq \frac{1}{2}(\mathbf{A}+\mathbf{A}^{\mathrm{T}})$.

Aufgabe P56. (a) Buch, Beispiel 6.1, S. 158–159, (b) Sind x_{jk} die unbekannten vom Lager k zur Fabrik j zu transportierenden Mengen, so muß gelten:

$$\text{(L19.1)} \quad \text{Kosten} := S := \sum_{j,k=1}^{m,n} c_{jk}x_{jk}, \quad z_j = \sum_{k=1}^{n} x_{jk}, \quad l_k = \sum_{j}^{m} x_{jk}, \quad x_{jk} \geq 0, \quad 1 \leq j \leq m, \quad 1 \leq k \leq n.$$

Aufgabe P57. Wir betrachten quadratische Teilgleichungssysteme $(\mathbf{a}^{j_1}, \mathbf{a}^{j_2})\,\mathbf{x} = \mathbf{b}$ und setzen $I = \{j_1, j_2\}$ mit $1 \leq j_1 < j_2 \leq 5$. Ist die Lösung $\mathbf{x}$ eindeutig und zulässig, so heißt $\mathbf{x}$ Basisvektor zur Indexmenge I. Sei beispielsweise $I = \{1, 3\}$, dann ist das Teilgleichungssystem $\begin{pmatrix} 1 & 3 \\ -1 & 5 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 3 \\ -1 \end{pmatrix}$ eindeutig lösbar mit zulässiger Lösung $\mathbf{x}^{\mathrm{T}} = (\,2.25, \quad 0.25\,)$. Die Wahl $I = \{1, 2\}$ führt jedoch zu einem singulären System.

Aufgabe 37. Das benutzte matlab-cg-Programm mit den entsprechenden Hilfsprogrammen ist unten angegeben. Die Rechenergebnisse sind in das Hauptprogramm eingeblendet.

```
%Es soll das cg-Verfahren ausprobiert werden am Modellproblem Delta u=0
%auf dem Quadrat [0 1]x[0 1]. Die Randwerte sind Eins auf den Quadratseiten
%parallel zur x-Achse und Null auf den Quadratseiten parallel zur y-Achse.
%Das Residuum r wird mit r=residuum(u) und das Produkt  A*h mit
%z=Amalh(h) berechnet, zwei separat formulierten Prozuduren.
%Wir verwenden die Formeln und Bezeichnungen aus dem Buch, S. 148.
%Die Lsung x hat formal die Form einer Matrix x(j,k), j,k=0,...,n.
%Nur die (n-1)^2 inneren Punke 0<j,k<n werden berechnet. Die Randpunkte x(j,k)
%mit j,k in {0,n} werden vorbesetzt und spielen die Rolle der rechten Seite.
%Da von Matrizen, interpretiert als Vektoren nicht ad hoc Skalarprodukte
%gebildet werden knnen, mu dafr auch ein separates Programm z=skalarpr(x,y)
%geschrieben werden. Aus matlab-technischen Grnden laufen die Indizes
%von 1 bis n+1, und nicht von Null bis n.
n=33;%n=100 kommt bei norm(r,inf)<1e-10 mit 213 Schritten zum Ende.
x=zeros(n+1,n+1);r=x;h=x;hilfs=x;        %Startwert Nullvektor
x(1,:)=ones(size(x(1,:)));x(n+1,:)=ones(size(x(1,:))); %Randwerte
h=residuum(x);   r=h;   step=0;          %(5.53)
tic  %Stoppuhr beginnt.
y=norm(r,inf);Y=[y];
while y>=1e-10 & step <=(n-1)^2
  step=step+1;           hilfs=Amalh(h);
  alpha=skalarpr(r,r)/skalarpr(h,hilfs); %(5.54)
  x=x+alpha*h;    r2hilfs=skalarpr(r,r); %(5.55)
  r=r-alpha*hilfs;                       %(5.56)
  beta=skalarpr(r,r)/r2hilfs;            %(5.57)
  h_alt=h;
  h=r+beta*h;                            %(5.58)
  h0=skalarpr(h_alt,Amalh(h));           %Test auf Orthogonalitt
  y=norm(r,inf);Y=[Y,y];
end; %while
toc %Stoppuhr stoppt.
semilogy(Y); print -dps cgfehlera.ps %Y protokolliert Fehlerabfall
    %=========================Ergebnisse====================================
    %n=33: elapsed_time = 6.613817 Sekunden in 68 Schritten.
    %Fehler des Residuums: 0.00000000008149 = 8.149e-11.
    %Das SOR-Verfahren braucht zum Vergleich bei einem optimalen Parameter
    %omega=1.82 auf demselben Rechner zur Erreichung derselben Genauigkeit
    %161 Schritte und 353.641601 Sekunden, also etwa 53-mal so lange.
    %auf Sun-ultra1: SOR: 161 Schritte bei 22.6717 Sekunden,
    %cg: 68 Schritte in 0.374341 Sekunden, also wieder hnlicher Faktor 61.
    %n=10: elapsed_time=0.070138 Sekunden (Sun-ultra1, ohne Bildschirmanzeige),
    %13 Iterationsschritte, Fehler des Residuums: nicht erkennbar.
```

```
%Das zu lsende lin. Glsys. r:=b-Ax=0 hat hier die Besonderheit, da
%nur r selbst einfach ausgerechnet werden kann. x ist hier formal eine Matrix,
%die auch die Randwerte enthlt. Das Ergebnis r hat dieselbe Gre wie x.
%r(j,k)=u(j,k)-0.25(u(j+1,k)+u(j-1,k)+u(j,k+1)+u(j,k-1)), (j,k) innerer Pkt,
%die Randpunkte bleiben bei r immer Null. Fr x=0 ist offensichtlich r=b.
function r=residuum(x);
  [z s]=size(x);
  r=zeros(size(x));
  hhh=x(1:z-2,2:s-1)+x(3:z,2:s-1)+x(2:z-1,1:s-2)+x(2:z-1,3:s);
  r(2:z-1,2:s-1)=0.25*hhh-x(2:z-1,2:s-1);
return;
```

```
%Es soll das Produkt A*u ausgerechnet werden, wobei A i.w. den Fnfpunkte-
%differenzenstern darstellt. A wird nicht gebraucht, die Randwerte von u
%werden ungendert auf y bertragen.
function y=Amalh(u);
  [z s]=size(u);    y=u;
  hhh=u(1:z-2,2:s-1)+u(3:z,2:s-1)+u(2:z-1,1:s-2)+u(2:z-1,3:s);
  y(2:z-1,2:s-1)=u(2:z-1,2:s-1)-0.25*hhh; %Rand(y)=Rand(u)
return
```

```
%Zwei gleich grosse Matrizen x,y werden als Vektoren interpretiert
%und davon wird das Skalarprodukt gebildet.
function s=skalarpr(x,y);
  [zx sx]=size(x);
  [zy sy]=size(y);
  if zx==zy & sx==sy
    n=zx*sx;
    xv=reshape(x,n,1);
    yv=reshape(y,n,1);
    s=xv'*yv; else
   disp('skalarpr: matrices must have the same size');
  end; %if
return;
```

In der folgenden Tabelle sind alle Zwischenergebnisse für das Modellproblem mit $n = 3$ und für ein System in gewöhnlicher Schreibweise protokolliert.

```
Wir protokollieren alle Zwischenergebnisse beim Modellproblem mit n=3.
Das entspricht einem 4x4-Gleichungssystem.
Es ist zu beachten, dass in
unserer Matrixschreibweise fr die Unbekannten, tatschlich nur
die inneren 2x2 Gren Unbekannte sind.
Das Gleichungssystem, konventionell geschrieben, she so aus:

x_11-(x_10+x_12+x_01+x_21)/4=0 ==> x_11-(     x_12     +x_21)/4=1/4
x_12-(x_11+x_13+x_02+x_22)/4=0 ==> x_12-(x_11          +x_22)/4=1/4
x_21-(x_20+x_22+x_11+x_31)/4=0 ==> x_21-(     x_22+x_11     )/4=1/4
x_22-(x_21+x_23+x_12+x_32)/4=0 ==> x_22-(x_21     +x_12     )/4=1/4

Vor Beginn der Rechnung:
x =      1    1        1          1
         0    0        0          0
         0    0        0          0
         1    1        1          1

h =      0         0         0      0
         0    0.2500    0.2500      0
         0    0.2500    0.2500      0
         0         0         0      0

r =      0         0         0      0
         0    0.2500    0.2500      0
         0    0.2500    0.2500      0
         0         0         0      0
Erster Iterationsschritt:
hilfs =  0         0         0          0
         0    0.1250    0.1250          0
         0    0.1250    0.1250          0
         0         0         0          0
alpha =     2
x = 1.0000    1.0000    1.0000    1.0000
         0    0.5000    0.5000         0
         0    0.5000    0.5000         0
    1.0000    1.0000    1.0000    1.0000
r2hilfs =   0.2500
r =  0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0  Die Lsung wird bereits in einem Schritt erreicht
```

```
beta =      0
h =  0     0     0     0  Die Richtung vergrert den Krylow-Raum nicht mehr.
     0     0     0     0
     0     0     0     0
     0     0     0     0
----------------------------------------------------------------------
----------------------------------------------------------------------
Wir demonstrieren das cg-Verfahren auch an einem konventionellen Gleichungssystem. Wir whlen
              A              |    b
  16   -120    240   -140 |   -4
-120   1200  -2700   1680 |   60
 240  -2700   6480  -4200 | -180
-140   1680  -4200   2800 |  140
Die rechte Seite ist so gewhlt, da x=(1 1 1 1)^T die Lsung ist.
=====================================================================
0. Schritt:
  x    h    r
  0   -4   -4
  0   60   60
  0 -180 -180
  0  140  140
=====================================================================
1. Schritt:
      hilfs
       -70064
       793680
     -1917360
      1249360
alpha =  9.792648279175893e-05
          x
  -0.00039170593117
   0.00587558896751
  -0.01762676690252
   0.01370970759085
r2hilfs =      55616
          r
   2.86112109032180
 -17.72229086216323
   7.76032104560690
  17.65456945928806
beta =  0.01248151634099
          h
   2.81119502495785
 -16.97339988170403
   5.51364810422930
  19.40198174702619
Norm Residuum = 17.72229086216323, Orthogonalittstest = 0.00000004470348
========================================================================
2.Schritt:
     hilfs
   1.0e+03 *
   0.68878520663518
  -2.99694380745490
   0.74298286448664
   2.25934774915339
alpha =  0.00689093359858
          x
   0.01898005231849
  -0.11108698255952
   0.02036741606968
   0.14740747549054
r2hilfs =    6.941720128203145e+02
          r
  -1.88525203228789
   2.92944991369484
   2.64047546154404
   2.08555414376271
beta =  0.03379205490635
          h
  -1.79025597565206
   2.35588385294489
   2.82679296101644
   2.74118697625019
Norm Residuum=  2.92944991369484, Orthogonalittstest = 0.00000000056480
=======================================================================
3. Schritt:
      hilfs
 -16.68602399490049
  14.74446596804319
  14.08525002804345
```

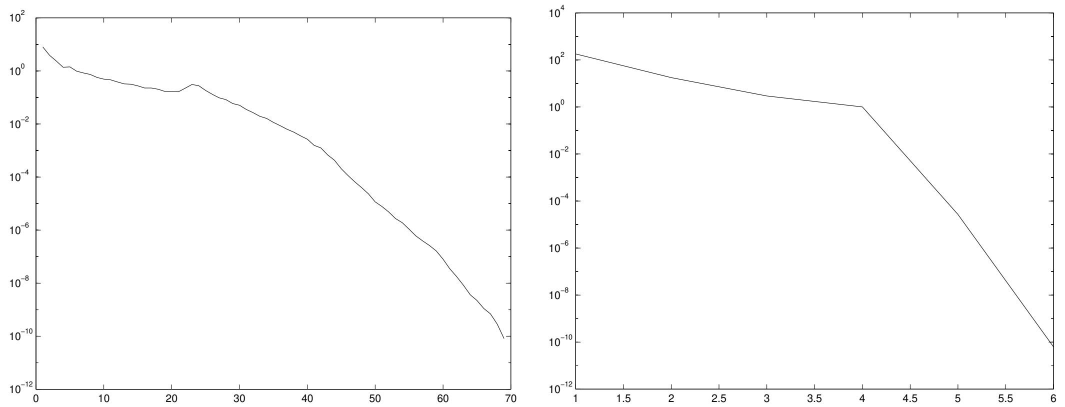
```
  11.31380677018296
alpha =  0.17319751434540
        x
  -0.29108783270645
   0.29694624485699
   0.50996093048680
   0.62217424613304
r2hilfs =  23.45749877167435
         r
   1.00472584793654
   0.37574505767945
   0.20094516775347
   0.12603093338289
beta =   0.05145139924093
         h
   0.91261467298979
   0.49695857836259
   0.34638762096219
   0.26706883889199
Norm Residuum=   1.00472584793654, Orthogonalittstest = 0.00000000002108
=========================================================================
4. Schritt:
      hilfs
   0.71019695037391
   0.26560601695229
   0.14202042721536
   0.08909828693857
alpha =  1.41471298985382
         x
   0.99999999990341
   1.00000000108583
   0.99999999738657
   1.00000000169872
r2hilfs =  1.20692113449516
         r
   1.0e-04 *
   0.00996888001703
  -0.11224686292499
   0.27024547317345
  -0.17570522819366
beta =   9.661250968100366e-10
         h
   1.0e-04 *
   0.00997769701642
  -0.11224206168345
   0.27024881971119
  -0.17570264797458
Norm Residuum=  0.00002702454732, Orthogonalittstest = 0.00000000000109
========================================================================
5. Schritt:
      hilfs
   0.01030867780014
  -0.11607400594796
   0.27946116870510
  -0.18169759983263
alpha =  9.670230402318969e-05
         x
   0.99999999999990
   1.00000000000042
   0.99999999999994
   0.99999999999964
r2hilfs =   1.166036797906211e-09
         r
   1.0e-10 *
   0.15106996794465
  -0.62480130041221
   0.08418548657800
   0.53719933049720
beta =   6.079303143103908e-12
         h
   1.0e-10 *
   0.15107002860209
  -0.62480198276573
   0.08418712950249
   0.53719826234754
Norm Residuum=  0.00000000006248, Orthogonalittstest = 0.00000000000000
========================================================================
Rechenzeit (Sun: ultra1) = 0.060326 Sekunden

Wie man sieht, ist der Orthogonalittstest nur bedingt erfolgreich.
```

```
Das fhrt hier dazu, da ein Schritt mehr als theoretisch voerhergesagt
gerechnet werden mu, um die geforderte Genauigkeit des
Residuums von 1e-10 zu erreichen.
```

Zur Lösung des Systems in gewöhnlicher Schreibweise wurde das folgende Programm verwendet.

```
%Es soll das cg-Verfahren ausprobiert werden an einem konventionellen Beispiel.
%Gehrt zu Numerik, Aufgabenblatt 19, SS 1998
n=4;
A=invhilb(n);      %schwieriges Beispiel, erzeugt groe Rundungsfehler.
%A(1,1)=-A(1,1);  %Wir verderben die positive Definitheit: Verfahren geht aber durch
%A(2,1)=2*A(2,1); %Wir verderben auch die Symmetrie, fhrt zum Chaos
b=sum(A)';
x=zeros(n,1);r=x;h=x;hilfs=x;           %Startwert Nullvektor
h=b-A*x; r=h;                            %(5.53)
step=0;
%x,h,r
tic  %Stoppuhr beginnt.
y=norm(r,inf);Y=[y];
while y>=1e-10 & step <=50
  step=step+1;
  hilfs=A*h;
  alpha=r'*r/(h'*hilfs);                 %(5.54)
  x=x+alpha*h;                           %(5.55)
  r2hilfs=r'*r;
  r=r-alpha*hilfs;                       %(5.56)
  beta=r'*r/r2hilfs;                     %(5.57)
  halt=h;
  h=r+beta*h;                            %(5.58)
  test0=halt'*A*h;                       %Test auf Konjugiertheit
  y=norm(r,inf);Y=[Y,y];
  [step y test0]
end; %while
toc %Stoppuhr stoppt.
%In dem Beispiel mit n=4 protokollieren wir den Fehler
%und zeichnen den Abfall auf einer halblogarithmischen Skala.
semilogy(Y);
```

Der Fehlerverlauf ergibt sich aus der Figur L19.1.



Figur L19.1: Fehlerabfall beim cg-Verfahren, links Modellproblem mit $n = 33$, rechts 4×4-System

Aufgabe 38. Die drei Indexmengen

$$I = \{3, 4, 5\},\ I = \{1, 3, 5\},\ I = \{2, 3, 5\}$$

definieren Basisvektoren und weitere gibt es nicht. Die dazugehörenden Basisvektoren sind

$$(0.3241,\ 0.0764,\ 0.5069)^{\mathrm{T}},\ (0.2095,\ 0.4841,\ 0.6619)^{\mathrm{T}},\ (0.0435,\ 0.3333,\ 0.5217)^{\mathrm{T}}.$$

Nach den Regeln der Kombinatorik gibt es höchstens $\binom{5}{3} = 10$ Basisvektoren.