Numerische Mathematik 2, Übungen, WS15/16, Blatt 7

Bearbeitung: Hausaufgaben bis 19.11.2015, Programmieraufgabe bis 26.11.2015

Hausaufgaben

1. Zeigen Sie, der Arnoldi Algorithmus (links) ist äquivalent zum modifizierten Arnoldi Algorithmus (rechts):

$\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$
for $j = 1, \ldots, m$
  $\boldsymbol{w}_j = A\boldsymbol{v}_j$
  for $i = 1, \ldots, j$
    $h_{ij} = (\boldsymbol{v}_i, A\boldsymbol{v}_j)_2$
    $\boldsymbol{w}_j = \boldsymbol{w}_j - h_{ij}\boldsymbol{v}_i$
  end
  $h_{j+1,j} = \|\boldsymbol{w}_j\|_2$
  if $h_{j+1,j} = 0$
    $\boldsymbol{v}_{j+1} = \boldsymbol{0}$
    stop
  else
    $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j/h_{j+1,j}$
  end
end

$\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$
for $j = 1, \ldots, m$
  $\boldsymbol{w}_j = A\boldsymbol{v}_j$
  for $i = 1, \ldots, j$
    $h_{ij} = (\boldsymbol{v}_i, \boldsymbol{w}_j)_2$
    $\boldsymbol{w}_j = \boldsymbol{w}_j - h_{ij}\boldsymbol{v}_i$
  end
  $h_{j+1,j} = \|\boldsymbol{w}_j\|_2$
  if $h_{j+1,j} = 0$
    $\boldsymbol{v}_{j+1} = \boldsymbol{0}$
    stop
  else
    $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j/h_{j+1,j}$
  end
end

2. Zeigen Sie, die obigen Algorithmen sind äquivalent zum folgenden Lanczos Algorithmus, falls A symmetrisch ist:

$\boldsymbol{v}_0 = \boldsymbol{0}$
$\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$
$c_1 = 0$
for $j = 1, \ldots, m$
  $\boldsymbol{w}_j = A\boldsymbol{v_j} - c_j\boldsymbol{v}_{j-1}$
  $a_j = (\boldsymbol{w}_j, \boldsymbol{v}_j)_2$
  $\boldsymbol{w}_j = \boldsymbol{w}_j - a_j\boldsymbol{v_j}$
  $c_{j+1} = \|\boldsymbol{w}_j\|_2$
  if $c_{j+1} = 0$
    $\boldsymbol{v}_{j+1} = \boldsymbol{0}$
    stop
  else
    $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j/c_{j+1}$
  end
end

wobei $H = \begin{bmatrix} a_1 & c_2 & & & \\ c_2 & a_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & a_{m-1} & c_m \\ & & & c_m & a_m \\ & & & & c_{m+1} \end{bmatrix}$

3. Bemerken Sie dass die Matrix $A = I + \tau B$ für die Programmieraufgabe des 3. Blatts nicht symmetrisch ist, wenn Konvektion dabei ist, d.h. wenn $k > 0$ gilt. Die Systeme $A\boldsymbol{V}^n = \boldsymbol{V}^{n-1}$, $n = 1, \ldots, M$, sind für diese Programmieraufgabe mit der Symmetrischen Gauß-Seidel Methode gelöst worden. Ändern Sie Ihren Code, um zwei andere Methoden zu implementieren. Erstens sollen Systeme $A\boldsymbol{V}^n = \boldsymbol{V}^{n-1}$ mit der GMRES Methode gelöst werden. Zweitens sollen Systeme

$A^\top A\boldsymbol{V}^n = A^\top \boldsymbol{V}^{n-1}$ mit der Methode der Konjugierten Gradienten gelöst werden, wobei zu merken ist, die Matrix $A^\top A$ ist SPD wie vorausgesetzt. Vergleichen Sie die jeweiligen Schnelligkeiten der zwei Methoden. Wählen Sie z.B. `N=51`, `M=51`, `r=0.001`, `k=0.1`, `d=0.01`.

Programmieraufgabe

Alle Codes sollen an
`stephen.keeling@uni-graz.at`
mit Betreff
Num2 Programmieraufgabe
per Email bis zum 26.11.2015 geschickt werden.

Das folgende Problem der Membrandynamik (mit Dämpfung $s \geq 0$ und Diffusion $r \geq 0$)

$$\left\{\begin{array}{rcll} v_{tt} + sv_t &=& r(v_{xx} + v_{yy}), & (x,y) \in (0,1)^2, \qquad\qquad t \in (0,T] \\ v &=& 0, & x \in \{0,1\} \;\mid\; y \in \{0,1\} \quad t \in (0,T] \\ v &=& v_0, & (x,y) \in (0,1), \qquad\qquad t = 0 \\ v_t &=& v_1, & (x,y) \in (0,1), \qquad\qquad t = 0 \end{array}\right.$$

soll gelöst werden. Zuerst wird das Problem in erste Ordnung so umgeschrieben:

$$\left\{\begin{array}{ll} \left[\begin{array}{c} v \\ v_t \end{array}\right]_t = \left[\begin{array}{cc} 0 & I \\ r(\partial_x^2 + \partial_y^2) & -sI \end{array}\right] \left[\begin{array}{c} v \\ v_t \end{array}\right], & (x,y) \in (0,1)^2, \qquad\qquad t \in (0,T] \\ \left[\begin{array}{c} v \\ v_t \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \end{array}\right], & x \in \{0,1\} \;\mid\; y \in \{0,1\}, \quad t \in (0,T] \\ \left[\begin{array}{c} v \\ v_t \end{array}\right] = \left[\begin{array}{c} v_0 \\ v_1 \end{array}\right], & (x,y) \in (0,1)^2, \qquad\qquad t = 0. \end{array}\right.$$

Das Problem sei auf dem räumlichen Gitter $\{(x_i, y_j) : x_i = ih, y_j = jh, i, j = 0, \ldots, N\}$, $h = 1/N$, in $\mathbb{R}^2$ so diskretisiert, dass

$$v(x_i, y_j, t) = 0, \quad i \in \{0, N\} \quad \text{oder} \quad j \in \{0, N\}$$

und für $1 \leq i, j \leq N-1$,

$$\begin{aligned} (\partial_x^2 + \partial_y^2)v(x_i, y_j, t) \;\approx\; & \frac{1}{h^2}\left[\underbrace{v(x_{i+1}, y_j, t)}_{i \leq N-2} - 2v(x_i, y_j, t) + \underbrace{v(x_{i-1}, y_j, t)}_{i \geq 2}\right] \\ + \; & \frac{1}{h^2}\left[\underbrace{v(x_i, y_{j+1}, t)}_{j \leq N-2} - 2v(x_i, y_j, t) + \underbrace{v(x_i, y_{j-1}, t)}_{j \geq 2}\right] \end{aligned}$$

wobei ein Term ausgelassen werden soll, wenn die untergeklammerte Bedingung nicht erfüllt wird. Das Problem sei auf dem zeitlichen Gitter $\{t^n = n\tau : n = 0, \ldots, M\}$, $\tau = T/M$, so diskretisiert:

$$(I - \tau B)\boldsymbol{V}^n = \boldsymbol{V}^{n-1}, \quad n = 1, \ldots, M$$

wobei im $n$ten Zeitschritt die exakte Lösung so approximiert wird:

$$\left\{\{v(x_i, y_j, t^n)\}_{i,j=1}^{N-1}, \{v_t(x_i, y_j, t^n)\}_{i,j=1}^{N-1}\right\} \approx \boldsymbol{V}^n.$$

Bemerken Sie, wegen der Randbedingung gelten $v(x_i, y_j, t) = 0$ und $v_t(x_i, y_j, t) = 0$ für $i \in \{0, N\}$ oder $j \in \{0, N\}$. Da diese Null-Werte nicht approximiert werden müssen, erscheinen sie in $\boldsymbol{V}$ nicht. Zur grafischen Darstellung wird der Membran-Zustand $\mathtt{v}^n \approx \{v(x_i, y_j, t^n)\}_{i,j=0}^N$ vom langen Vektor $\mathtt{V}^n$ (lexikografische Reihenfolge) so extrahiert:

```
Nm = N-1; NmNm = Nm*Nm; Np = N+1;
v  = reshape(V(1:NmNm),Nm,Nm);
v  = [zeros(Nm,1),v,zeros(Nm,1)];  % Randwerte
v  = [zeros(1,Np);v;zeros(1,Np)];  % zufuegen
```

Wenn die Membran-Zustände $\mathtt{v0}^n \approx \{v(x_i, y_j, t^n)\}_{i,j=0}^N$ und $\mathtt{v1}^n \approx \{v_t(x_i, y_j, t^n)\}_{i,j=0}^N$ gegeben sind, werden sie in den langen Vektor $\mathtt{V}^n$ (lexikografische Reihenfolge) so verkettet:

```
V0 = [reshape(v0(2:N,2:N),NmNm,1);reshape(v1(2:N,2:N),NmNm,1)];
```

Die Anfangsbedingung ist

```
x  = linspace(0,1,Np);
y  = linspace(0,1,Np);
v0 = (x.*(1-x))'*(y.*(1-y));
v0 = v0/max(v0(:));
v1 = zeros(Np,Np);
V0 = [reshape(v0(2:N,2:N),NmNm,1);reshape(v1(2:N,2:N),NmNm,1)];
```

Der Differentialoperator $(\partial_x^2 + \partial_y^2)$ wird approximiert durch $D \in \mathbb{R}^{(N-1)^2 \times (N-1)^2}$,

$$D = \frac{1}{h^2} \begin{bmatrix} D_1 & D_0 & & & \\ D_0 & D_1 & D_0 & & \\ & \ddots & \ddots & \ddots & \\ & & D_0 & D_1 & D_0 \\ & & & D_0 & D_1 \end{bmatrix}, \quad D_1 = \begin{bmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ & & & 1 & -4 \end{bmatrix}, \quad \begin{array}{l} D_0 = -I \\ \\ D_0, D_1 \in \mathbb{R}^{(N-1)\times(N-1)} \end{array}$$

und der Operator $[0, I; r(\partial_x^2 + \partial_y^2), -sI]$ wird approximiert durch

$$B = \begin{bmatrix} 0 & B_0 \\ rD & -sB_0 \end{bmatrix}, \quad B_0 = I \in \mathbb{R}^{(N-1)^2 \times (N-1)^2}.$$

Schreiben Sie einen Matlab Code der Form ($\mathtt{N}= N$, $\mathtt{M}= M$, $\mathtt{T}= T$, $\mathtt{s}= s$, $\mathtt{r}= r$, $\mathtt{V}= \boldsymbol{V}^M$)

```
V = familienv(N,M,T,s,r,gon)
```

zur Lösung der obigen Systeme $(I - \tau B)\boldsymbol{V}^n = \boldsymbol{V}^{n-1}$, $n = 1, \ldots, M$, wobei für jedes n die GMRES Methode implementiert werden soll. Mit `gon=true` soll das Ergebnis für jedes $n = 0, \ldots, M$ so grafisch dargestellt werden:

```
v = reshape(V(1:NmNm),Nm,Nm);
v = [zeros(Nm,1),v,zeros(Nm,1)];   % Randwerte
v = [zeros(1,Np);v;zeros(1,Np)];   % zufuegen
surf(x,y,v); axis([0 1 0 1 -1 1]);
view([1 1 1]); drawnow; pause(0.1);
```

In jedem Zeitschritt soll die Iteration mit dem aktuellsten `V` (d.h. $\boldsymbol{V}^{n-1}$) gestartet werden. Die Lösung $\boldsymbol{V}^M$ zur Zeit $t^M = T$ soll zum Schluss der Rechnungen ausgegeben werden. Falls Sie noch weitere Parameter im Befehlsfenster ausgeben wollen, soll dies nur mit `gon=true` geschehen.

Achtung: Getestet werden die Ergebnisse z.B. mit dem folgenden Code:

```
N=20; M=100; T=10; r=0.0025; s=0.1;
Nm = N-1; Np = N+1; NmNm = Nm*Nm; h = 1/N; tau = T/M;

D1 = 4*spdiags(ones(Nm,1), 0,Nm,Nm) ...
   -   spdiags(ones(Nm,1),+1,Nm,Nm) ...
   -   spdiags(ones(Nm,1),-1,Nm,Nm);
D0 = -speye(Nm);
D  = [D1,D0,sparse(Nm,NmNm-2*Nm)];
for j=2:(N-2)
    D  = [D;sparse(Nm,(j-2)*Nm),D0,D1,D0,sparse(Nm,(Nm-j-1)*Nm)];
end
D  = [D;sparse(Nm,NmNm-2*Nm),D0,D1];
B  = [sparse(NmNm,NmNm),speye(NmNm);-r*D/h^2,-s*speye(NmNm)];
A  = speye(2*NmNm) - tau*B;
[L,U] = lu(A);

surf(x,y,v0); axis([0 1 0 1 -1 1]); pause(0.01); drawnow;     % v0 oben
V = V0;                                                       % V0 oben
for n=1:M
    V = L\V;
    V = U\V;
    v = reshape(V(1:NmNm),Nm,Nm);
    v = [zeros(Nm,1),v,zeros(Nm,1)];
    v = [zeros(1,Np);v;zeros(1,Np)];
    surf(x,y,v); axis([0 1 0 1 -1 1]); pause(0.01); drawnow;  % x,y oben
end
```