Numerische Mathematik 2, Übungen, WS15/16, Blatt 5

Bearbeitung: Hausaufgaben bis 5.11.2015, Programmieraufgabe bis 12.11.2015

Hausaufgaben

1. Für die Tschebyscheff-Polynome, $T_m(x) = \cos(m \cdot \arccos(x))$, $m \in \mathbb{N}_0$, zeigen Sie mittels einer Induktion unter Verwendung von $T_0(x) = 1$, $T_1(x) = x$ und dem Kosinus-Additionstheorem die Rekursionsformel
$$T_{m+1}(x) = 2xT_m(x) - T_{m-1}(x), \quad m \in \mathbb{N}.$$
Weiters zeigen Sie, es gilt
$$T_m\left(\frac{1}{2}\left(x + \frac{1}{x}\right)\right) = \frac{1}{2}\left(x^m + \frac{1}{x^m}\right).$$

2. Sei $A \in \mathbb{R}^{n\times n}$ SPD mit Eigenvektoren $\{\boldsymbol{u}_i\}_{i=1}^n$ und entsprechenden Eigenwerten $\{\lambda_i\}_{i=1}^n$. Sei $\boldsymbol{b} \in \text{span}\{\boldsymbol{u}_{i_l}\}_{l=1}^k$, $1 \le k \le n$. Zeigen Sie, zur Lösung des Gleichungssystems $A\boldsymbol{x} = \boldsymbol{b}$ konvergiert die Methode der Konjugierten Gradienten mit $\boldsymbol{x}_0 = 0$ nach höchstens k Iterationen. (Hinweis: Stellen Sie die Lösung $\boldsymbol{x}^\star = A^{-1}\boldsymbol{b}$ bezüglich $\{\boldsymbol{u}_{i_l}\}_{l=1}^k$ dar. Verwenden Sie das Polynom $p(z) = \prod_{l=1}^k (1 - z/\lambda_{i_l})$, um $\|\boldsymbol{x}_k - \boldsymbol{x}^\star\|_A$ bezüglich $\|p(A)\boldsymbol{x}^\star\|_A$ abzuschätzen.)

3. Zur Lösung des Gleichungssystems $A\boldsymbol{x} = \boldsymbol{b}$, $A \in \mathbb{R}^{n\times n}$ regulär, sei $\boldsymbol{x}_0$ ein Anfangsvektor für ein iteratives Verfahren. Mit $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$ sei $\{\boldsymbol{u}_j = A^{j-1}\boldsymbol{r}_0\}_{j=1}^i$ eine Basis für den Krylov-Unterraum $\mathcal{K}^i(A, \boldsymbol{r}_0) = \text{span}\{A^j\boldsymbol{r}_0\}_{j=0}^{i-1}$, $1 \le i \le n$. Für die Matrix $U_i \in \mathbb{R}^{n\times i}$ mit Spalten $\{\boldsymbol{u}_j\}_{j=1}^i$ zeigen Sie,
$$AU_i = U_iB_i + \boldsymbol{u}_{i+1}\boldsymbol{e}_i^\top$$
wobei $\boldsymbol{e}_i \in \mathbb{R}^i$ erfüllt $(\boldsymbol{e}_i)_j = \delta_{i,j}$ und $B_i \in \mathbb{R}^{i\times i}$ erfüllt $(B_i)_{kj} = \delta_{k,j+1}$.
Sei $U_i = Q_iR_i$, $Q_i \in \mathbb{R}^{n\times i}$, $R_i \in \mathbb{R}^{i\times i}$, so zerlegt, dass $Q_i^\top Q_i = I$ und $R_i = \{r_{kj}\}_{k,j=1}^i$ mit $r_{kj} = 0$, $k > j$. Zeigen Sie mit der obigen Gleichung, es gelten $AQ_iR_i = Q_iR_iB_i + \boldsymbol{u}_{i+1}\boldsymbol{e}_i^\top$ und
$$AQ_i = Q_iR_iB_iR_i^{-1} + \boldsymbol{u}_{i+1}\boldsymbol{e}_i^\top R_i^{-1} = Q_i\tilde{H}_i + \frac{1}{r_{ii}}\boldsymbol{u}_{i+1}\boldsymbol{e}_i^\top.$$
Zeigen Sie, $\tilde{H}_i$ ist eine obere Hessenberg-Matrix.
(Bemerkung: Mit $U_{i+1} = Q_{i+1}R_{i+1}$, $Q_{i+1} = [Q_i, \boldsymbol{q}_{i+1}]$ und $R_{i+1} = [[R_i; 0], [\tilde{\boldsymbol{r}}; r_{i+1,i+1}]]$ folgen $\boldsymbol{u}_{i+1} = Q_i\tilde{\boldsymbol{r}} + r_{i+1,i+1}\boldsymbol{q}_{i+1}$ und $AQ_i = Q_iH_i + (r_{i+1,i+1}/r_{i,i})\boldsymbol{q}_{i+1}\boldsymbol{e}_i^\top$ mit der oberen Hessenberg-Matrix $H_i = \tilde{H}_i + \tilde{\boldsymbol{r}}\boldsymbol{e}_i^\top/r_{i,i}$. Daher gilt $Q_iAQ_i^\top = H_i$, und falls A symmetrisch ist, ist H_i eine tridiagonale Matrix.)

Programmieraufgabe

Alle Codes sollen an
`stephen.keeling@uni-graz.at`
mit Betreff
Num2 Programmieraufgabe
per Email bis zum 12.11.2015 geschickt werden.

Seien verrauschte Daten $v : [0,1]^2 \to \mathbb{R}$ gegeben, d.h. $v = u^\star$+Rauschen, und $u^\star$ ist eine zu rekonstrierende Funktion. Die Rekonstruktion erfolgt durch die folgende anisotropische Diffusion:

$$u^\star(\boldsymbol{x}) \approx u(\boldsymbol{x},T): \quad \begin{cases} u_t(\boldsymbol{x},t) = \nabla \cdot \left[\dfrac{\nabla u(\boldsymbol{x},t)}{1+|\nabla u(\boldsymbol{x},t)|^2}\right], \quad \boldsymbol{x} = (x,y) \in [0,1]^2, \quad t \in [0,T] \\ u_x(\boldsymbol{x},t) = 0, \quad x \in \{0,1\}, \quad u_y(\boldsymbol{x},t) = 0, \quad y \in \{0,1\} \\ u(\boldsymbol{x},0) = v(\boldsymbol{x}). \end{cases}$$

Diese Evolution sei auf dem räumlichen Gitter $\{(x_i,y_j) : x_i = ih, y_j = jh, i,j = 0,\ldots,N\}$, $h = 1/N$, in $\mathbb{R}^2$ und dem zeitlichen Gitter $\{t^n = n \cdot dt : n = 0,\ldots,M\}$, $dt = T/M$, so diskretisiert:

$$(I + dt \cdot B)\boldsymbol{U}^n = \boldsymbol{U}^{n-1}, \quad n = 1,\ldots,M$$

wobei im $n$ten Zeitschritt die exakte Lösung so approximiert wird: $\{u(x_i,y_j,t^n)\}_{i,j=0}^N \approx \boldsymbol{U}^n$. Diese Daten werden als Bilder betrachtet, und die Einträge eines Bildes, z.B. $\boldsymbol{U}$, werden als $(N+1)\times(N+1)$-Feld oder als $(N+1)^2$-Vektor mit der lexikografischen Reihenfolge der Einträge gespeichert:

$$\langle U_{0,0},\ldots,U_{N,0},U_{0,1},\ldots,U_{N,1},\ldots,U_{0,N},\ldots,U_{N,N}\rangle^{\mathrm{T}}$$

Ein $(N+1)\times(N+1)$ Feld `U` wird in einen $(N+1)^2$-Vektor so umgewandelt (überschrieben):

```
N1 = N+1; N1N1 = N1*N1;
U  = reshape(U,N1N1,1);
```

und ein $(N+1)^2$-Vektor wird in ein $(N+1)\times(N+1)$ Feld `U` so umgewandelt (überschrieben):

```
U = reshape(U,N1,N1);
```

Der räumliche Differentialoperator $\nabla \cdot [\nabla u/(1+|\nabla u|^2)]$ wird so approximiert, dass das Produkt $B\boldsymbol{U}$ Komponenten hat, die folgendermaßen gegeben werden:

$$|\nabla U|^2_{i+1/2,j} = \left[\frac{U_{i+1,j}-U_{i,j}}{h}\right]^2 + \left[\frac{1}{2}\left(\frac{U_{i,j^+}-U_{i,j^-}}{(j^+-j^-)h} + \frac{U_{i+1,j^+}-U_{i+1,j^-}}{(j^+-j^-)h}\right)\right]^2$$

$$j^+ = \min(j+1,N), \quad j^- = \max(j-1,0), \quad i = 0,\ldots,N-1, \quad j = 0,\ldots,N$$

$$|\nabla U|^2_{i,j+1/2} = \left[\frac{U_{i,j+1}-U_{i,j}}{h}\right]^2 + \left[\frac{1}{2}\left(\frac{U_{i^+,j}-U_{i^-,j}}{(i^+-i^-)h} + \frac{U_{i^+,j+1}-U_{i^-,j+1}}{(i^+-i^-)h}\right)\right]^2$$

$$i^+ = \min(i+1,N), \quad i^- = \max(i-1,0), \quad i = 0,\ldots,N, \quad j = 0,\ldots,N-1$$

und

$$i,j = 0,\ldots,N: \quad \left(\nabla \cdot \left[\frac{\nabla U}{1+|\nabla U|^2}\right]\right)_{i,j} =$$

$$\underbrace{\frac{1}{1+|\nabla_h U|^2_{i+1/2,j}}\left[\frac{U_{i+1,j}-U_{i,j}}{h}\right]}_{0\le i\le N-1} - \underbrace{\frac{1}{1+|\nabla_h U|^2_{i-1/2,j}}\left[\frac{U_{i,j}-U_{i-1,j}}{h}\right]}_{1\le i\le N} +$$

$$\underbrace{\frac{1}{1+|\nabla_h U|^2_{i,j+1/2}}\left[\frac{U_{i,j+1}-U_{i,j}}{h}\right]}_{0\le j\le N-1} - \underbrace{\frac{1}{1+|\nabla_h U|^2_{i,j-1/2}}\left[\frac{U_{i,j}-U_{i,j-1}}{h}\right]}_{1\le j\le N}$$

Diese Diskretisierung lässt sich in Matlab so umschreiben:

```
    NN1 = N*N1;

% Vorwaerts-Differenzen in x
    Dx = spdiags(ones(N1,1),0,N1,N1) ...
       - spdiags(ones(N1,1),-1,N1,N1);
    Dx(find(sum(abs(Dx),2)~=2),:) = [];
    Dx = kron(speye(N1),Dx);
    Dx = Dx/h;

% Vorwaerts-Differenzen in y
    Dy = spdiags(ones(N1,1),0,N1,N1) ...
       - spdiags(ones(N1,1),-1,N1,N1);
    Dy(find(sum(abs(Dy),2)~=2),:) = [];
    Dy = kron(Dy,speye(N1));
    Dy = Dy/h;

% Zentrale Differenzen in x
    Z1 = spdiags(ones(N1,1),+1,N1,N1) ...
       - spdiags(ones(N1,1),-1,N1,N1);
    Z1( 1,   2) =  2.0;
    Z1( 1,   1) = -2.0;
    Z1(N1,N1-1) = -2.0;
    Z1(N1,N1  ) =  2.0;
    Z2 = spdiags(ones(N1,2),[0,1],N,N1);
    Z2(find(sum(abs(Z2),2)~=2),:) = [];
    Zx = kron(Z2,Z1);
    Zx = Zx/(4*h);

% Zentrale Differenzen in y
    Z1 = spdiags(ones(N1,1),+1,N1,N1) ...
       - spdiags(ones(N1,1),-1,N1,N1);
    Z1( 1,   2) =  2.0;
    Z1( 1,   1) = -2.0;
    Z1(N1,N1-1) = -2.0;
    Z1(N1,N1  ) =  2.0;
    Z2 = spdiags(ones(N1,2),[0,1],N,N1);
    Z2(find(sum(abs(Z2),2)~=2),:) = [];
    Zy = kron(Z1,Z2);
    Zy = Zy/(4*h);

% Koeffizienten 1/(1+|Du|^2) in Schnittstellen (i+1/2,j)
    Xu = 1./(1 + (Dx*u).^2 + (Zy*u).^2);
% Koeffizienten 1/(1+|Du|^2) in Schnittstellen (i,j+1/2)
    Yu = 1./(1 + (Zx*u).^2 + (Dy*u).^2);

% Diskreter Differentialoperator
    B  = Dx'*spdiags(Xu,0,NN1,NN1)*Dx ...
       + Dy'*spdiags(Yu,0,NN1,NN1)*Dy;
```

Schreiben Sie einen Matlab Code der Form (`N`= N, `M`= M, `T`= T, `Ustar`= $u^\star$, `V`= $\boldsymbol{U}^0$, `U`= $\boldsymbol{U}^M$)

```
U = familienv(N,M,T,Ustar,V,gon)
```

zur Lösung der obigen Systeme $(I + dt \cdot B)\boldsymbol{U}^n = \boldsymbol{U}^{n-1}$, $n = 1, \ldots, M$, wobei für jedes n die Methode der Konjugierten Gradienten möglichst effizient implementiert werden soll. Mit `gon = true` soll das Ergebnis für jedes $n = 0, \ldots, M$ so grafisch dargestellt werden:

```
subplot(1,3,1); imagesc(Ustar); colormap('gray'); axis image; axis off;
title('exakt');
subplot(1,3,2); imagesc(reshape(V,N1,N1)); colormap('gray'); axis image; axis off;
title('verrauscht');
subplot(1,3,3); imagesc(reshape(U,N1,N1)); colormap('gray'); axis image; axis off;
title('entrauscht');
drawnow;
```

In jedem Zeitschritt soll die Iteration mit dem aktuellsten `U` (d.h. $\boldsymbol{U}^{n-1}$) gestartet werden. Die Lösung $\boldsymbol{U}^M$ 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 den folgenden Daten

```
N = 101; N1 = N+1; NN1 = N*N1; N1N1 = N1*N1; N4 = round(N/4);
M = 101; T = 1;
Ustar = [zeros(N4,1);ones(N1-2*N4,1);zeros(N4,1)];
Ustar = kron(Ustar,Ustar');
V = Ustar + 0.5*randn(N1,N1);
V = reshape(V,N1N1,1);
```

und mit dem folgenden Code:

```
U = V;
for n=1:M
    Xu = 1./(1 + (Dx*U).^2 + (Zy*U).^2);
    Yu = 1./(1 + (Zx*U).^2 + (Dy*U).^2);
    B  = Dx'*spdiags(Xu,0,NN1,NN1)*Dx ...
       + Dy'*spdiags(Yu,0,NN1,NN1)*Dy;
    A  = speye(N1N1) + dt*B;
    U  = A\U; % grafisch darstellen wenn gon = true
end
```