Numerische Mathematik 2, Übungen, WS15/16, Blatt 13

Bearbeitung: Hausaufgaben bis 21.1.2016, Programmieraufgabe bis 28.1.2016

Hausaufgaben

1. Man leitet eine *backward difference formula* (BDF) zur numerischen Lösung einer gewöhnlichen Differentialgleichung $y'(t) = f(t, y)$ her. Der Ansatz für BDFm ist, die Lösung wird mit einem Lagrange-Interpolierenden Polynom (siehe Skriptum von Numerik 1) approximiert,

$$y(t) \approx p(t) = \sum_{l=0}^{m} y(t_{k-l})\ell_l(t), \quad \ell_l(t_{k-l}) = \delta_{k,l}, \quad t_k = kh$$

und mit gegebenen Werten $\{y_{k-l}\}_{l=1}^m$ wird die numerische Lösung y_k implizit gegeben durch die Aktualisierung,

$$a_0 y_k + \sum_{l=1}^{m} a_l y_{k-l} = p'(t_k) = f(t_k, y_k)$$

wobei $a_l = \ell_l'(t_k)$. Zum Beispiel sind die folgenden BDF Methoden wohl bekannt:

$$\begin{array}{rcll} y_k - y_{k-1} & = & hf(t_k, y_k) + \mathcal{O}(h^2) & \text{BDF1 (implizit Euler)} \\ \frac{3}{2}y_k - 2y_{k-1} + \frac{1}{2}y_{k-2} & = & hf(t_k, y_k) + \mathcal{O}(h^3) & \text{BDF2} \\ \frac{11}{6}y_k - 3y_{k-1} + \frac{3}{2}y_{k-2} - \frac{1}{2}y_{k-3} & = & hf(t_k, y_k) + \mathcal{O}(h^4) & \text{BDF3} \end{array}$$

Leiten Sie BDF2 her, und zeigen Sie die angegebene Genauigkeitsordnung.

2. Gegeben sei ein allgemeines Mehrschritt-Verfahren,

$$\sum_{l=0}^{m} a_l y_{k-l} = h \sum_{l=0}^{m} b_l f(t_{k-l}, y_{k-l})$$

Das Mehrschritt-Verfahren ist *absolut stabil*, wenn bei der Anwendung für die Evolutionsgleichung $y'(t) = \lambda y(t)$, $\Re\{\lambda\} < 0$, gilt $y_k \to 0$ für $k \to \infty$. Für diese Evolutionsgleichung kann das Verfahren bezüglich der Rekursionsgleichung geschrieben werden,

$$\sum_{l=0}^{m} a_l y_{k-l} = z \sum_{l=0}^{m} b_l y_{k-l}$$

wobei $z = h\lambda$. Die Lösungen $y_k = w^k$, $w \in \mathbb{C}$, erfüllen $y_k \to 0$ für $k \to \infty$ genau dann wenn $p(z, w) = 0$ und $|w| < 1$, wobei das *charakterische Polynom* p definiert sei durch

$$p(z, w) = \sum_{l=0}^{m} \alpha_l w^{m-l} - z \sum_{l=0}^{m} \beta_l w^{m-l}, \quad \alpha_l = a_l/a_0, \quad \beta_l = b_l/a_0.$$

Also besteht das Gebiet G der absoluten Stabilität für ein Mehrschritt-Verfahren aus den Werten $z = h\lambda$, für die alle Lösungen w der Gleichung $p(z, w) = 0$ erfüllen $|w| < 1$. Wenn $(-\infty, 0) \subset G$ gilt, ist das Verfahren A_0-stabil. Ein Verfahren ist A-stabil, wenn $G = \{z \in \mathbb{C} : \Re\{z\} < 0\}$ gilt. Zeigen Sie, das BDF2 Verfahren ist A_0-stabil. Bonus: Zeigen Sie, BDF2 ist tatsächlich A-stabil.

3. Das Anfangswertproblem

$$\boldsymbol{y}'(t) = -A\boldsymbol{y}(t), \quad t \geq 0, \quad \boldsymbol{y}(0) = \boldsymbol{y}_0$$

soll so numerisch gelöst werden, dass die qualitativen Eigenschaften der numerischen Lösung jene der exakten Lösung wiederspiegeln.

(a) Sei A SPD. Zeigen Sie, es gilt $\|\boldsymbol{y}(t)\|_2 < \|\boldsymbol{y}_0\|_2$, $\forall t > 0$. Wählen Sie ein numerisches Verfahren aus, wobei die numerische Lösung $\{\boldsymbol{y}^k\}_{k\geq 0}$ erfüllt $\|\boldsymbol{y}^k\|_2 < \|\boldsymbol{y}_0\|_2$, $\forall k \in \mathbb{N}$, und zeigen Sie, dass Ihr Verfahren diese Eigenschaft für eine beliebige SPD Matrix A erfüllt.

(b) Sei A schief symmetrisch. Zeigen Sie, es gilt $\|\boldsymbol{y}(t)\|_2 = \|\boldsymbol{y}_0\|_2$, $\forall t > 0$. Wählen Sie ein numerisches Verfahren aus, wobei die numerische Lösung $\{\boldsymbol{y}^k\}_{k\geq 0}$ erfüllt $\|\boldsymbol{y}^k\|_2 = \|\boldsymbol{y}_0\|_2$, $\forall k \in \mathbb{N}$, und zeigen Sie, dass Ihr Verfahren diese Eigenschaft für eine beliebige schief symmetrische Matrix A erfüllt.

Programmieraufgabe

Alle Codes sollen an
`stephen.keeling@uni-graz.at`
mit Betreff
Num2 Programmieraufgabe
per Email bis zum 28.1.2015 geschickt werden.

Das Problem der Membrandynamik (mit Dämpfung $s \geq 0$ und Diffusion $r^2 \geq 0$)

$$\begin{cases} v_{tt} + sv_t &= r^2\nabla\cdot\nabla v, & (x,y) \in (0,1)^2, & t \in (0,T] \\ \hat{\boldsymbol{n}}\cdot\nabla v &= 0, & (x,y) \in \partial(0,1)^2, & t \in (0,T] \\ v &= v_0, & (x,y) \in (0,1), & t = 0 \\ v_t &= v_1, & (x,y) \in (0,1), & t = 0 \end{cases}$$

soll gelöst werden (vgl. Blatt 7). Hier ist $\hat{\boldsymbol{n}}$ der auswärtsgerichtete Einheitsvektor, d.h. $\hat{\boldsymbol{n}} = ((-1)^{x+1}, 0)^\top$, $x = 0, 1$ und $\hat{\boldsymbol{n}} = (0, (-1)^{y+1})^\top$. $y = 0, 1$. Zuerst wird das Problem in erste Ordnung so umgeschrieben:

$$\begin{cases} \begin{bmatrix} r\nabla v \\ v_t \end{bmatrix}_t = \begin{bmatrix} 0 & r\nabla \\ r\nabla\cdot & -sI \end{bmatrix} \begin{bmatrix} r\nabla v \\ v_t \end{bmatrix}, & (x,y) \in (0,1)^2, & t \in (0,T] \\ \begin{bmatrix} \hat{\boldsymbol{n}}^\top & 0 \end{bmatrix} \begin{bmatrix} r\nabla v \\ v_t \end{bmatrix} = 0, & (x,y) \in \partial(0,1)^2, & t \in (0,T] \\ \begin{bmatrix} r\nabla v \\ v_t \end{bmatrix} = \begin{bmatrix} r\nabla v_0 \\ v_1 \end{bmatrix}, & (x,y) \in (0,1)^2, & t = 0 \end{cases}$$

und die Lösung v bekommt man durch

$$v(x,y,t) = \int_0^t v_t(x,y,s)ds.$$

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

$$\begin{bmatrix} v_x(x_{i-\frac{1}{2}}, y_j, t)\}_{i=1,j=0}^N \\ v_y(x_i, y_{j-\frac{1}{2}}, t)\}_{i=0,j=1}^N \end{bmatrix} \approx D_h\{v(x_i, y_j, t)\}_{i,j=0}^N, \quad D_h = \begin{bmatrix} D_x \\ D_y \end{bmatrix}$$

wobei

$$D_x = \frac{1}{h} D_2 \otimes D_1 \in \mathbb{R}^{N(N+1)\times(N+1)^2}, \quad D_y = \frac{1}{h} D_1 \otimes D_2 \in \mathbb{R}^{(N+1)N\times(N+1)^2}$$

und

$$D_1 = \begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{N\times(N+1)}, \quad D_2 = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \in \mathbb{R}^{(N+1)\times(N+1)}.$$

Der Differentialoperator wird approximiert durch*

$$\begin{bmatrix} 0 & r\nabla \\ r\nabla\cdot & -sI \end{bmatrix} \approx \begin{bmatrix} 0 & rD_h \\ -rD_h^\top & -sD_3 \end{bmatrix} = B_h, \quad D_3 = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \in \mathbb{R}^{(N+1)^2\times(N+1)^2}$$

oder

```
N1   = N+1; NN1  = N*N1; N1N1 = N1*N1;
D1 = spdiags(ones(N1,1),1,N,N1) ...
   - spdiags(ones(N1,1),0,N,N1);
D2 = speye(N1);
Dx = kron(D2,D1)/h;
Dy = kron(D1,D2)/h;
Dh = [Dx;Dy];
Bh = [sparse(2*NN1,2*NN1),r*Dh;-r*Dh',-s*speye(N1N1)];
```

Das Problem sei auf dem zeitlichen Gitter $\{t^n = n\tau : n = 0, \ldots, M\}$, $\tau = T/M$, so diskretisiert, dass der Zustandsvektor approximiert wird mit

$$\boldsymbol{V}^n = \begin{bmatrix} r\boldsymbol{v}_x^n \\ r\boldsymbol{v}_y^n \\ \boldsymbol{v}_t^n \end{bmatrix} \approx \begin{bmatrix} r\{v_x(x_{i-\frac{1}{2}}, y_j, t^n)\}_{i=1,j=0}^N \\ r\{v_y(x_i, y_{j-\frac{1}{2}}, t^n)\}_{i=0,j=1}^N \\ \{v_t(x_i, y_j, t^n)\}_{i,j=0}^N \end{bmatrix}$$

und man bekommt $\boldsymbol{v}^n \approx \{v(x_i, y, j, t^n)\}_{i,j=0}^N$ durch

$$\boldsymbol{v}^0 = \boldsymbol{v}_0, \quad \boldsymbol{v}^n = \boldsymbol{v}^{n-1} + \tau \boldsymbol{v}_t^n, \quad n = 1, 2, \ldots, M.$$

Hier sind die Anfangsbedingungen $v_0(x, y) = xy(1-x)(1-y)$, $v_1(x, y) = 0$,

$$\boldsymbol{V}^0 = \begin{bmatrix} r\boldsymbol{v}_x^0 \\ r\boldsymbol{v}_y^0 \\ \boldsymbol{v}_t^0 \end{bmatrix} = \begin{bmatrix} rD_x\boldsymbol{v}_0 \\ rD_y\boldsymbol{v}_0 \\ \boldsymbol{v}_1 \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{v}_0 \\ \boldsymbol{v}_1 \end{bmatrix} = \begin{bmatrix} \{v_0(x_i, y_j)\}_{i,j=0}^N \\ \{v_1(x_i, y_j)\}_{i,j=0}^N \end{bmatrix}.$$

oder

```
x  = linspace(0,1,N1);
y  = linspace(0,1,N1);
v0 = 16*((x.*(1-x))'*(y.*(1-y))).^2;  % n^ . Dh v0(:) = 0 gilt am Rand
v1 = zeros(N1,N1);
V0 = [r*Dh*v0(:);v1(:)];
```

*Bemerken Sie, $-\nabla\cdot$ ist der adjungierte Operator von ∇, und daher verwendet man die Diskretisierung $-\nabla\cdot \approx D_h^\top$ wenn $\nabla \approx D_h$.

Zur Lösung des Membran-Problems verwendet man das Crank-Nicolsan Verfahren[†]

$$\boldsymbol{V}^{n+1} - \boldsymbol{V}^n = \tau B_h(\boldsymbol{V}^{n+1} + \boldsymbol{V}^n)/2, \quad n = 0, 1, \ldots, M-1.$$

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

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

zur Lösung der obigen Systeme. Mit `gon=true` soll das Ergebnis für jedes $n = 0, \ldots, M$ so grafisch dargestellt werden:

```
vt = V(l1:l2);     % l2 = length(V); l1 = l2-N1N1+1;
v  = v + tau*vt;
surf(x,y,reshape(v,N1,N1));
title(['Energie=',num2str(sum(V.^2))]); % potentielle + kinetische
axis([0 1 0 1 -0.1 0.1]); pause(0.01); drawnow;
```

wobei `v = v0(:)` anfänglich gilt. 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 Parametern

```
N=20; M=1000; T=10; r=0.1; s=0.0; % oder s=0.1;
```

und mit einem Code, der zu dem am Ende des 7. Blatts ähnlich ist.

[†]Warum? Schauen Sie sich die `Energie` an, wenn es keine Dämpfung gibt. Dies ist ein Hinweis für Hausaufgabe 3.