TENTAMEN I GRUNDKURS I NUMERISKA METODER - DEL 20

Lösningsförslag

Uppgift 1 Man vill lösa ekvationssystemet

$$\begin{aligned} 10\sin(x+y)+xy &= 5 \\ \cos(xy)+10y &= 0 \end{aligned}$$

med Newtons metod.

(a) **3p** Inför lämpliga beteckningar och formulera Newtons metod för detta ekvationssystem

(b) **4p** Skriv ett program som bestämmer lösningen till den första ekvationen när $y = 0$.

(c) **5p** Använd (b) som startgissning (dvs vi antar att en lösning har ett litet y-värde) och skriv ett MATLAB-program som löser det olinjära ekvationssystemet.

Lösning:

(a)

$$\mathbf{z} = \begin{bmatrix} x \\ y \end{bmatrix}, \ \mathbf{f}(\mathbf{z}) = \begin{bmatrix} 10\sin(x+y)+xy-5 \\ \cos(xy)+10y \end{bmatrix}$$

och Jacobimatris

$$J(\mathbf{z}) = \begin{bmatrix} 10\cos(x+y)+y & 10\cos(x+y)+x \\ -y\sin(xy) & -x\sin(xy)+10 \end{bmatrix}$$

Newtons metod:

$$\mathbf{z}_{n+1} = \mathbf{z}_n - J(\mathbf{z})^{-1}\mathbf{f}(\mathbf{z}_n).$$

(b) Man kan antingen lösa ekvationen analytiskt när $y = 0$:

$$10\sin(x) = 5 \ \Rightarrow \ \sin(x) = 1/2 \ \Rightarrow \ x = \pi/3;$$

eller med Newtons metod:

```
g=@(x) 10*sin(x)-5
gp=@(x) 10*cos(x)
x=0;
dx=inf;
TOL=1e-10;
while (abs(dx)>TOL)
   dx=gp(x)\g(x);
   x=x-dx;
end
```

(c) Vi fortsätter programmet i (b) kan fortsättas:

```
f=@(z) [10*sin(z(1)+z(2))+z(1)*z(2)-5];
J=@(z) [10*cos(z(1)+z(2))+z(2) 10*cos(z(1)+z(2))+z(2)
         -z(2)*sin(z(1)*z(2)),-z(1)*sin(z(1)*z(2))+10]
z=[x;0];
dz=inf;
while (norm(dz)>TOL)
   dz=J(z)\f(x);
   z=z-dz;
end
```

Uppgift 2 Följande differentialekvation ska lösas

$$y'''(x)+\alpha y(x)^2=\beta x,\ \ 1\le x\le 4.$$

där $\alpha=0.1$. Antag att begynnelsevärdena är givna som $y(1)=1, y'(1)=0.5$, $y''(1)=4$.

(a) **6p** Antag att $\beta=3$. Skriv ett MATLAB-program som löser differentialekvationen på det givna intervallet. Basera programmet på framåt Euler med N steg och steglängd h. Skriv programmet i den här funktionen:

```
function [y4]=solve_ode(y1,yp1,ypp1,beta,N)
# Löser differentialekvationen med givna begynnelsedata
# y1 är begynnelsevillkor y(1)=y1
# yp1 är begynnelsevillkor y'(1)=yp1
# ypp1 är begynnelsevillkor y''(1)=ypp1
# beta är parameter i differentialekvationen
# y4 är approximationen vid y(4)
# N är antalet diskretiseringsintervall
```

(b) **4p** Skriv ett MATLAB-program som med hjälp av `solve_ode` bestämmer β så att $y(4)=0$. Bestäm startgissningar/startgissning genom att lösa problemet exakt för $\alpha=0$.

(c) **5p** Med vilken noggrannhetsordning approximerar metoden i (a) värdet $y(4)$? (Ingen härledning krävs.) Skriv ett program som anropar `solve_ode` och använder Richardsonextrapolation för att öka noggrannheten med $h=0.4$ och $h=0.2$.

Lösning:

(a) Vi behöver först skriva om differentialekvationen till ett ODE-system. Vi ansätter

$$\mathbf{z}(x)=\begin{bmatrix}z_1(x)\\ z_2(x)\\ z_3(x)\end{bmatrix}=\begin{bmatrix}y(x)\\ y'(x)\\ y''(x)\end{bmatrix}.$$

Du uppfyller $\mathbf{z}$ ett ODE-system:

$$\mathbf{z}'(x)=\mathbf{f}(x,\mathbf{z}(x))$$

med

$$\mathbf{f}(x,\mathbf{z}) := \begin{bmatrix} z_2 \\ z_3 \\ \beta x - \alpha z_1^2 \end{bmatrix}$$

Programmet blir således

```
function [y4]=solve_ode(y1,yp1,ypp1,beta,N)
alpha=0.1;
f=@(x,z) [z(2); z(3); beta*x-alpha*z(1)^2];
a=1; b=4;
h=(b-a)/N;
x=a;
for i=1:N
  z=z+h*f(x,z)
  x=x+h;
end
y4=z(1);
```

Vi anropar programmet med:

```
y4=solve_ode(1,0.5,4,3,N);
```

(b) Enligt uppgiftslydelse ska startgissning bestämmas utifrån ekvationen

$$y'''(x) = \beta x.$$

Genom att integrera funktionen får vi att y uppfyller

$$y(x) = \frac{1}{24}\beta x^4 + c_1 x^2 + c_2 x + c_3.$$

Konstanterna c_1, c_2 och c_3 bestäms utifrån begynnelsevillkor: $1 = y(1) = \frac{\beta}{24} + c_1 + c_2 + c_3$, $0.5 = y'(1) = \frac{\beta}{12} + 2c_1 + c_2$, $4 = y''(1) = \frac{\beta}{6} + 2c_1$ och randvärdet $0 = y(4) = \frac{\beta}{24}4^4 + c_1 4^4 + c_2 4 + c_3$. Problemet motsvara linjär interpolation och kan lösas med Vandermonde-matrisen:

$$\begin{bmatrix} 1 \\ 0.4 \\ 4 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{24} & 1 & 1 & 1 \\ \frac{1}{12} & 2 & 1 & 0 \\ \frac{1}{6} & 2 & 0 & 0 \\ \frac{4^4}{24} & 4^2 & 4 & 1 \end{bmatrix} \begin{bmatrix} \beta \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

I princip kan startgissning bestämmas genom att lösa 4×4-problemet för hand, men det är lättare att lösa som en dela av MATLAB-programmet. Vi använder sekantmetoden eftersom derivatan är komplicerad.

```
A=[1/24, 1, 1, 1;
   1/12, 2, 1, 0;
   1/6, 2, 0, 0;
   4^4/24, 4^2,4,1]
```

```
b=[1;0.4;4;0];
x=A\b;
x0=x(end);
x1=x(end)*(1.01);
f=@(x) solve_ode(1,0.5,4,x,N);
f0=f(x0);
f1=f(x1);
TOL=1e-5;
h=inf;
while (abs(h)>TOL)
  h=((x1-x0)/(f1-f0))*f1;
  x2=x1-h;
  x0=x1;
  f0=f1;
  x1=x2;
  f1=f(x1);
end
x
```

(c) Noggrannhetsordning för Euler framåt är 1. Richardsonextrapolation

```
N1=10; N2=20;
ny_approx=2*solve_ode(1,0.5,4,N2)-solve_ode(1,0.5,4,N1);
```

Uppgift 3 Genom mätningar har vi fått följande datavärden

t	-1	-0.5	-0.25	0	0.25	1
y	9.3	5.3	5.1	4.9	5.4	11.4

(a) **6p** Skriv ett MATLAB-program som bestämmer minstakvadratanpassningen av

$$g(t) = \alpha_1 + \alpha_2 t + e^{\alpha_1 t^2}$$

OBS: Om du använder MATLAB-operatorn \ måste du ange vilket matematiskt problem som MATLAB löser i detta steg.

(b) **5p** Utvidga ditt MATLAB-program i (a) så att det bestämmer minimum av minstrakvadratanpassningen. Ange tydligt hur du kommit fram till startvärden. Denna uppgift kan lösas även om (a) inte lösts.

Lösning:

(a) Vi låter $\mathbf{x} = [\alpha_1, \alpha_2]$ och

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \alpha_1 + \alpha_2 t_1 + e^{\alpha_1 t_1^2} - y_1 \\ \vdots \\ \alpha_1 + \alpha_2 t_m + e^{\alpha_1 t_m^2} - y_m \end{bmatrix}.$$

Vi vill bestämma

$$\mathbf{f}(\mathbf{x}) \approx 0,$$

som är ett olinjärt överbestämt ekvationssystem. Vi använder Gauss-Newton. Jacobimatrisen är

$$J(\mathbf{x}) = \begin{bmatrix} 1+t_1^2 e^{\alpha_1 t_1^2} & t_1 \\ \vdots & \vdots \\ 1+t_m^2 e^{\alpha_1 t_m^2} & t_m \end{bmatrix}$$

I MATLAB

```
tv=[-1;...1];
yv=[9.3;5.3... ];
m=length(tv);
J=@(x)  [ones(m,1)+(tv.^2).*exp(x(1)*tv.^2), tv];
f=@(x)  x(1)+x(2)*tv+exp(x(1)*tv.^2)-y;
h=inf;
x=[1;0];
while (norm(h)>TOL)
  h=J(x)\f(x);
  x=x-h;
end
```

(b) Eftersom α_1 och α_2 är bestämda från (a) behöver vi endast bestämma nollställe av derivatan funktion i en variabel: $g'(t) = 0$.

$$\begin{aligned} g'(t) &= \alpha_2 + 2t\alpha_1 e^{\alpha_1 t^2} \\ g''(t) &= (2\alpha_1 + (2t\alpha_1)^2) e^{\alpha_1 t^2} \end{aligned}$$

Ekvationen $g'(t) = 0$ går inte att lösa analytiskt. Vi använder Newtons metod (fortsättning av programmet i a):

```
alpha1=x(1);
alpha2=x(2);
f=@(t) alpha2+2*t*alpha1*exp(alpha1*t^2);
fp=@(t) (2*alpha1+(2*t*alpha1)^2)*exp(alpha1*t^2);
h=inf;
while (abs(h)>TOL)
  h=f(t)/fp(t);
  t=t-h;
end
```

Uppgift 4 Ingenjören Ingvar har utvecklat en numerisk metod för att lösa linjära ekvationssystem med n obekanta när matrisen är tridiagonal, och har gjort det tillgängligt i MATLAB med programmet:

```
function x=ingvars_metod(T,b,n)
% T är en tridiagonal matris
% b är högerledvektorn i det linjära ekvationssystemet  T x = b
% n är antalet obekanta i stystemet
% x är lösningen till ekvationssystemet
```

På en viss dator kommer Ingvar fram till att beräkningstiden för hans metod kan väl uppskattas till $\alpha n + \beta$ millisekunder där $\alpha = 1$ och $\beta = 10$. Han vill använda det för att lösa differentialekvationen

$$\mathbf{y}'(t) = T\mathbf{y}(t), \quad \mathbf{y}(0) = \mathbf{y}_0,$$

vid tidpunkt $t = 1$ där T är en tridiagonal $n \times n$ matris och $\mathbf{y}_0$ en given vektor. Matrisen T och vektorn $\mathbf{y}_0$ är tillgängliga i MATLAB. Vi antar att beräkningstiden för att multiplicera T med en vektor är $\alpha n/3$ och att addera två vektorer är $\alpha n/10$. Ett steg av bakåt Euler innebär lösningen av ett linjärt ekvationssystem.

(a) **4p** Skriv ett MATLAB program som beräknar lösningen med framåt Euler med $N = 10$ steg och $h = 1/N$.

(b) **4p** Skriv ett MATLAB program som beräknar lösningen med bakåt Euler med hjälp av Ingvars metod med $N = 10$ steg och $h = 1/N$.

(c) **2p** Vad är beräkningstiden för (a) respektive (b)?

(d) **2p** Ingvar upptäcker att problemet är styvt och att man behöver mycket finare diskretisering med framåt Euler för samma noggrannhet med bakåt Euler. Om man med framåt Euler väljer $N = 1000$ behöver man för samma noggrannhet endast $N = 10$ med bakåt Euler. Vilken metod är snabbast för att uppnå den noggrannheten för stora n-värden? Ge ett tydligt resonemang med tydliga antaganden om beräkningstider.

Lösning:

(a) Framåt Euler:

```
N=10;
h=1/N;
y=y0; n=length(y0);
for i=1:N
   y=y+h*T*b;
end
```

(b) Bakåt Euler för detta problem blir

$$\mathbf{y}_{n+1} = \mathbf{y}_n + hT\mathbf{y}_{n+1}$$

dvs

$$\mathbf{y}_{n+1} = (I - hT)^{-1}\mathbf{y}_n$$

Matrisen $(I - hT)$ är tridiagonal eftersom T är tridiagonal.

```
N=10;
h=1/N;
y=y0; n=length(y0);
T2=-h*T;
for i=1:n
    T2(i,i)=T2(i,i)+1;
end
for i=1:N
   y=invars_metod(T2,b);
end
```

(c) Summera ihop stegen i del olika delarna av algoritmerna. Det blir uppskattningsvis:

- Framåt Euler: $N(\alpha n/3 + N\alpha n/10)$
- Bakåt Euler: $N(n+10)$ (om man bortser från skapandet av T2)

(d) Jämför matrix-vektor produkter för $N = 1000$ och $N = 10$. Bakåt Euler är betydligt snabbare.