KTH Engineering Sciences

Tentamen 2020-05-25
Del 2

SF1547 – Numeriska metoder

Max antal poäng på denna del är 50. Rättas endast om del 1 är godkänd. Betygsgränser: 10p D, 20p C, 30p B, 40p A. Svar skall motiveras och uträkningar redovisas. Korrekt svar utan motivering eller med felaktig motivering medför poängavdrag.

Inga hjälpmedel är tillåtna (ej heller miniräknare).

Varje uppgift på Del 2 ska skrivas på separata papper. (Deluppgifter kan skrivas på samma papper.)
Skriv uppgiftsnummer, sidnummer, namn och personnummer på varje papper.

P1. Givet funktionen

$$f(x) = \sin(\alpha x) + \sin(\beta x^2).$$

Konstanterna α och β ska bestämmas så att f interpolerar de två datapunkterna $(1, 0.5)$ och $(2, 1)$, dvs f skall satisfiera $f(1) = 0.5$ och $f(2) = 1$. Problemet leder till följande icke-linjära ekvationssystem för α och β,

$$\begin{aligned} \sin(\alpha) + \sin(\beta) &= 0.5, \\ \sin(2\alpha) + \sin(4\beta) &= 1. \end{aligned}$$

(a) Antag att α och β är små, så att approximationen $\sin(x) \approx x$ är giltig. Använd denna för att hitta startgissningar för α och β. (3 p)

Lösning: Approximationen ger

$$\begin{aligned} \alpha + \beta &= 0.5, \\ 2\alpha + 4\beta &= 1. \end{aligned}$$

Dra ifrån två gånger första ekvationen från den andra. Det ger $\beta = 0$ och därmed $\alpha = 0.5$, vilket blir startgissningarna.

(b) Skriv ett MATLAB-program som implementerar Newtons metod för system och beräknar värden på α, β med ett fel mindre än 10^{-10}. (8 p)

Lösning: Vi låter

$$\mathbf{F}(\alpha, \beta) = \begin{pmatrix} \sin(\alpha) + \sin(\beta) - 0.5, \\ \sin(2\alpha) + \sin(4\beta) - 1 \end{pmatrix},$$

och beräknar funktionens Jacobi-matris

$$J(\alpha,\beta)=\begin{pmatrix}\cos(\alpha) & \cos(\beta)\\ 2\cos(2\alpha) & 4\cos(4\beta)\end{pmatrix}.$$

Newtons metod blir då

$$\begin{pmatrix}\alpha_{k+1}\\ \beta_{k+1}\end{pmatrix}=\begin{pmatrix}\alpha_k\\ \beta_k\end{pmatrix}-J(\alpha_k,\beta_k)^{-1}\mathbf{F}(\alpha_k,\beta_k),\qquad \begin{pmatrix}\alpha_0\\ \beta_0\end{pmatrix}=\begin{pmatrix}0.5\\ 0\end{pmatrix}.$$

En implementation i MATLAB kan tex se ut som:

```
F = @(al,be) [sin(al)+sin(be)-0.5;...
              sin(2*al)+sin(4*be)-1];
J = @(al,be) [cos(al) cos(be);...
              2*cos(2*al) 4*cos(4*be)];
tol = 1e-10;
X = [0.5;0];
d = [1;1];

while norm(d)>tol
    jj = J(X(1),X(2));
    ff = F(X(1),X(2));
    d = -jj\ff;
    X = X+d;
end
X
```

(c) Fler datapunkter tillkommer: (6 p)

x_j	1	2	3	4	5	6
y_j	0.5	1	1.5	1.8	1.8	1.2

Konstanterna α och β ska nu istället bestämmas så att felkvadratsumman

$$\sum_{j=1}^{6}|f(x_j)-y_j|^2,$$

minimeras. Modifiera ditt MATLAB-program i deluppgift (b) så att det beräknar dessa konstanter.

Lösning: Vi får nu ett overbestämt icke-linjärt ekvationssystem $\mathbf{F}(\alpha,\beta)\approx 0$, där

$$\mathbf{F}(\alpha,\beta)=\begin{pmatrix}\sin(\alpha x_1)+\sin(\beta x_1^2)-y_1\\ \sin(\alpha x_2)+\sin(\beta x_2^2)-y_2\\ \vdots\\ \sin(\alpha x_6)+\sin(\beta x_6^2)-y_6\end{pmatrix}\in\mathbb{R}^6.$$

Dess Jacobi-matris är

$$J(\alpha,\beta)=\begin{pmatrix}x_1\cos(\alpha x_1) & x_1^2\cos(\beta x_1^2)\\ x_2\cos(\alpha x_2) & x_2^2\cos(\beta x_2^2)\\ \vdots & \vdots\\ x_6\cos(\alpha x_6) & x_6^2\cos(\beta x_6^2)\end{pmatrix}\in\mathbb{R}^{6\times 2}.$$

Vi kan lösa detta med Gauss–Newtons metod,

$$\begin{pmatrix}\alpha_{k+1}\\ \beta_{k+1}\end{pmatrix}=\begin{pmatrix}\alpha_k\\ \beta_k\end{pmatrix}-(J(\alpha_k,\beta_k)^TJ(\alpha_k,\beta_k))^{-1}J(\alpha_k,\beta_k)^T\mathbf{F}(\alpha_k,\beta_k).$$

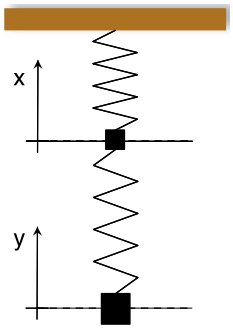
Eftersom backslash i MATLAB ger minstakvadratlösningen när man ger den ett överbestämt linjärt ekvationssystem räcker det att modifiera början av programmet i deluppgift (b). Vi tar bort definitionerna av `F` och `J` och ersätter dem med raderna:

```
xx = [1 2 3 4 5 6]';
yy = [0.5 1 1.5 1.8 1.8 1.2]';

F = @(al,be) sin(al*xx)+sin(be*xx.^2)-yy;
J = @(al,be) [xx.*cos(al*xx) xx.^2.*cos(be*xx.^2)];
```

Resten av programmet blir oförändrat.

P2. Två massor av olika storlek hänger från taket, ihopkopplade med fjädrar som i bilden till höger. Deras vertikala positioner x och y bestäms då av de ordinära differentialekvationerna



$$x'' = -2x + y - 1, \qquad y'' = -\eta y' + \frac{1}{2}(x - y) - 1,$$

där η är en friktionskoefficient för den undre fjädern. Vid $t = 0$ släpps massorna från vila i läget $x = -1.5$ och $y = 0$, varför

$$x(0) = -1.5, \quad x'(0) = y(0) = y'(0) = 0.$$

Vi vill beräkna massornas rörelse $x(t)$ och $y(t)$.

(a) Skriv om ekvationerna som ett system av första ordningens differentialekvationer på formen (3 p)

$$\mathbf{u}' = \mathbf{F}(t, \mathbf{u}), \quad \mathbf{u}(0) = \mathbf{u_0}.$$

I ditt svar ska det klart framgå vad $\mathbf{u}$, $\mathbf{F}(t, \mathbf{u})$ och $\mathbf{u_0}$ är.

Lösning: Låt $\mathbf{u}(t) = (x(t),\ x'(t),\ y(t),\ y'(t))^T$. Systemet blir då som i uppgiftstexten, med

$$\mathbf{F}(t, \mathbf{u}) = A\mathbf{u} - \mathbf{b}, \qquad A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1/2 & 0 & -1/2 & -\eta \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \qquad \mathbf{u}(0) = \begin{pmatrix} -1.5 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

(b) Skriv ett MATLAB-program som beräknar $x(t)$ och $y(t)$ fram till $t = 20$ med $\eta = 0.05$. Programmet ska använda Framåt Euler-metoden. Det ska skriva ut $x(20)$ och $y(20)$. (6 p)

Lösning: Framåt Euler för detta problem lyder

$$\mathbf{u}_{n+1} = \mathbf{u}_n + h(A\mathbf{u}_n - \mathbf{b}).$$

Programmet kan tex se ut som:

```
eta = 0.05;
A   = [0 1 0 0; -2 0 1 0; 0 0 0 1; 0.5 0 -0.5 -eta];
b   = [0 1 0 1]';
u   =  [-1.5 0 0 0]';  % Begynnelsedata
T   = 20;              % Sluttid

n = 4000;              % Antal steg
```

```
h = T/n;
U = u;
for k=1:n
    u=u+h*(A*u-b);
    U=[U u];
end
disp(['x(20) = ' num2str(u(1))])
disp(['y(20) = ' num2str(u(3))])
```

(c) Utöka ditt program med kod som räknar ut hur lång sträcka massorna (totalt) rört sig efter 20 tidsenheter, genom att beräkna integralerna (4 p)

$$L_1 = \int_0^{20} |x'(t)|dt, \qquad L_2 = \int_0^{20} |y'(t)|dt.$$

Not: Du får inte använda inbyggda MATLAB-funktioner för integralberäkning.

Lösning: Man tex lägga till följande rader till programmet ovan för att räkna ut L_1 och L_2 med trapetsregeln:

```
xpa = abs(U(2,:)); ypa = abs(U(4,:));
L1 = h*(sum(xpa(2:end-1))+xpa(1)/2+xpa(end)/2)
L2 = h*(sum(ypa(2:end-1))+ypa(1)/2+ypa(end)/2)
```

(d) Beskriv hur man kan gå tillväga för att uppskatta det numeriska felet i approximationen av L_1 och L_2. (2 p)

Lösning: Lös problemet med två olika steglängder, h och $h/2$. Det ger approximationerna L_1^h och $L_1^{h/2}$. Felet kan sedan uppskattas som

$$|L_1 - L_1^{h/2}| \leq |L_1^h - L_1^{h/2}|,$$

och likadant för L_2.

(e) När friktionskoefficienten η är stor blir systemet styvt och man behöver använda en implicit ODE-metod istället för Framåt Euler. Välj en implicit metod och modifiera ditt MATLAB-program i deluppgift (b) så att det använder denna metod för att beräkna $x(t)$ och $y(t)$ fram till $t = 20$. (6 p)

Lösning: Vi väljer Bakåt Euler, som för detta problem lyder

$$\mathbf{u}_{n+1} = \mathbf{u}_n + h(A\mathbf{u}_{n+1} - \mathbf{b}).$$

Vi kan skriva om detta som

$$(I - hA)\mathbf{u}_{n+1} = \mathbf{u}_n - h\mathbf{b}.$$

där I är identitetsmatrisen. I varje tidssteg behöver vi alltså lösa ett linjärt ekvationssystem med systemmatrisen $I - hA$. Allt annat är samma som för Framåt Euler. Det räcker därför att i ursprungsprogrammet byta ut raden "`u=u+h*(A*u-b);`" mot "`u=(I-h*A)\(u-h*b);`" samt dessförinnan definiera "`I = eye(4,4);`". (Bäst är om man också LU-faktoreriserar $I - hA$ eftersom man under tidsstegningen löser samma ekvationssystem om och om igen med olika högerled.)

P3. Betrakta följande differensformel för att approximera derivatan av en funktion f (12 p)

$$f'(x_0) \approx \frac{af(x_0-h)+bf(x_0)+cf(x_0+h)}{h}, \qquad h \ll 1.$$

Antag att parametrarna a, b och c är valda så att formeln är *exakt* för funktionerna $f(x)=1$ och $f(x)=x$. Visa att approximationen då har minst noggrannhetsordning ett. (Du får anta att f kan deriveras hur många gånger som helst.)

Lösning: När approximationen är exakt för 1 och x gäller

$$\begin{aligned} \frac{a+b+c}{h} &= 0, \\ \frac{a(x_0-h)+bx_0+c(x_0+h)}{h} &= 1. \end{aligned}$$

Detta kan skrivas om som

$$\begin{aligned} a+b+c &= 0, \\ (a+b+c)\frac{x_0}{h}+(-a+c) &= 1, \end{aligned}$$

vilket reducerar till det underbestämda linjära ekvationssystemet

$$\begin{aligned} a+b+c &= 0, \\ -a+c &= 1, \end{aligned}$$

med lösningarna

$$a = \frac{-1-b}{2}, \qquad c = \frac{1-b}{2}, \qquad b \in \mathbb{R}.$$

Om formeln är exakt för de givna funktionerna måste den alltså vara på formen

$$\begin{aligned} &\frac{af(x_0-h)+bf(x_0)+cf(x_0+h)}{h} \\ &\qquad = \frac{f(x_0+h)-f(x_0-h)}{2h} - \frac{bh}{2}\left(\frac{f(x_0-h)-2f(x_0)+f(x_0+h)}{h^2}\right). \end{aligned}$$

I uttrycket identifierar vi de kända andra ordningens approximationerna för $f'(x_0)$ och $f''(x_0)$. Det ger

$$\begin{aligned} \frac{af(x_0-h)+bf(x_0)+cf(x_0+h)}{h} &= f'(x_0)+O(h^2)-\frac{bh}{2}\left(f''(x_0)+O(h^2)\right) \\ &= f'(x_0)+O(h), \end{aligned}$$

vilket är vad vi skulle visa.

Alternativ lösning: Låt $p(x)$ vara linjäriseringen av f runt x_0, d.v.s.

$$p(x) = f(x_0)+(x-x_0)f'(x_0) \quad \Rightarrow \quad f(x) = p(x)+\frac{1}{2}(x-x_0)^2 f''(\xi),$$

där ξ ligger mellan x_0 och x_0+x. Eftersom $p(x)$ är en linjärkombination av 1 och x kommer formeln vara exakt även för $p(x)$. Vi får, för några $\xi_1, \xi_2 \in (x_0-h, x_0+h)$,

$$\begin{aligned} \frac{af(x_0-h)+bf(x_0)+cf(x_0+h)}{h} &= \frac{ap(x_0-h)+bp(x_0)+cp(x_0+h)}{h} \\ &\quad + \frac{ah^2 f''(\xi_1)+ch^2 f''(\xi_2)}{2h} \\ &= p'(x_0)+\frac{h}{2}(af''(\xi_1)+cf''(\xi_2)) \\ &= p'(x_0)+O(h), \end{aligned}$$

för fixt a och c. Här utnyttjade vi också att $f''(x)$ är begränsad i en omgivning av x_0. Slutligen har vi $p'(x_0) = f'(x_0)$, vilket ger det sökta sambandet

$$\frac{af(x_0 - h) + bf(x_0) + cf(x_0 + h)}{h} = f'(x_0) + O(h).$$