TENTAMEN I GRUNDKURS I NUMERISKA METODER - DEL 2

Lösningsförslag

Rättas endast om del 1 är godkänd. Betygsgräns (inkl bonuspoäng): 10p D, 20p C, 30p B, 40p A. Maximal poäng 50 + bonuspoäng (max 4p). Miniräknare är ej tillåten på denna tentamen. Svar skall motiveras och uträkningar redovisas. Korrekt svar utan motivering eller med felaktig motivering medför poängavdrag. Då algoritmbeskrivning begärs, avses om inte annat anges beskrivning i Matlab. Eftersom miniräknare ej är tillåten är det tillåtet att lämna enkla beräkningsuttryck oförenklade.

Uppgift 1
Vi ska beräkna integralen

$$I = \int_0^1 \cos(\pi x^2)\, dx$$

och har följande MATLAB-kod till vår hjälp

```
>>  xv1=0:0.1:1;
>>  s1=sum(cos(pi*xv1.^2))
s1 =  3.7397
>>  xv2=0:0.05:1;
>>  s2=sum(cos(pi*xv2.^2))
s2 =  7.4796
```

(a) **[6p]** Beräkna integralen med trapetsregeln med $h = 0.1$, för komplicerade beräkningar kan du ta hjälp av utdata från programmet.

(b) **[6p]** Använd Richardsonextrapolation med $h = 0.1$ och $h = 0.05$ för att bestämma en noggrannare lösning.

Lösning:

(a) Det beräknade värdet `s1` motsvarar

$$\sum_{i=0}^{n} \cos(\pi(hi)^2)$$

där $h = 1/(n+1) = 0.1$. Trapetsregeln kan uttryckas som

$$I = \int_0^1 \cos(\pi x^2)\, dx \approx T(h) = h\sum_{i=0}^{n} \cos(\pi(hi)^2) - \frac{h}{2}\cos(0) - \frac{h}{2}\cos(\pi)$$

Eftersom $\cos(0) = 1$ och $\cos(\pi) = -1$ är trapetsgegeln approximationen $T(0.1) = 0.1 * s1$ dvs 0.37397. Motivering för varför de extra termerna ger noll bidrag i trapetsregeln krävs för full poäng.

(b) Vi använder formeln för Richardsonextrapolation med $n = 2$:

$$R(h) = \frac{4T(h/2) - T(h)}{3}.$$

Värder för $T(h)$ har beräknats i a. Värdet för $T(h/2)$ erhålls på samma sätt $T(0.05) = 0.05 * 7.4796$.

$$R(h) = 0.37399.$$

Uppgift 2 Vi vill lösa randvärdesproblemet

$$\begin{aligned} -y''(x) + y(x) &= e^x \\ y(0) &= 0 \\ y(1) &= 0 \end{aligned}$$

(a) **[8p]** Skriv ett MATLAB-program som löser problemet med matrismetoden (finita differenser) med $N = 10$ intervall.

(b) **[4p]** Utvidga ditt program så att det även beräknar $\int_0^1 y(x)\,dx$.

Lösning: a och b:

```
N=10;
A=zeros(N-1,N-1);
h=1/N;
xv=0:h:1; xv=xv(2:end-1);
for i=1:N-1
   A(i,i)=2/h^2+1;
   if (i>2)
      A(i-1,i)=-1/h^2;
      A(i,i-1)=-1/h^2;
   end
end
b=exp(xv)';
u=A\b;
T=sum(A)*h; approximation av integral. Randpunkter ger noll bidrag pga randvillkor
```

Raden `zeros` kan ersättas med `A=sparse(N-1,N-1);` som gör att matrisen sparas i glest format och blir i detta fall snabbare.

Uppgift 3 Vi vill bestämma en finit differens för att approximera tredjederivatan.

(a) **[7p]** Vi approximerar

$$f^{(3)}(0) \approx \frac{\alpha f(0) + \beta f(h) - \beta f(2h) - \alpha f(3h)}{h^3}$$

Bestäm konstanter α, β så att approximationen har minst noggrannhetsordning 2 för alla funktioner som är fyra gånger differentierbara.

(b) **[6p]** Konstruera en finit differensformel för tredjederivatan som har noggrannhetsordning (minst) 3. Du kan basera din metod på (a). Du behöver dock inte ha löst (a) för att lösa denna deluppgift.

Lösning:

(a) Med hjälp av Taylorutveckling ser vi att

$$\begin{aligned}
\alpha(f(0) - f(3h)) &= -\alpha(3hf'(0) + \frac{(3h)^2}{2!}f''(0) + \frac{(3h)^3}{3!}f'''(0) + \frac{(3h)^4}{4!}f^{(4)}(0) + \cdots \\
\beta(f(h) - f(2h)) &= \beta(-hf'(0) + \frac{1-2^2}{2!}h^2 f''(0) + \frac{1-2^3}{3!}h^2 f'''(0) + \frac{1-2^4}{4!}h^2 f^{(4)}(0) + \cdots
\end{aligned}$$

Genom att summera ovanstående och observera vilka koefficienter som ska sättas till noll får vi:
koefficient framför $f'(0)$ sätts till noll:

$$-3\alpha - \beta = 0$$

koefficient framför $f''(0)$ sätts till noll:

$$-9\alpha - 3\beta = 0$$

koefficient framför $f'''(0)$ sätts till h^3:

$$-\frac{3^3 h^3}{3!}\alpha - \frac{7h^3}{3!}\beta = h^3 \qquad (*)$$

De första två ekvationerna leder till $\beta = -3\alpha$. Insätting av $\beta = -3\alpha$ i ekvation (*) leder till

$$1 = -\frac{3^2}{2}\alpha + \frac{7}{2}\alpha = -\alpha.$$

Med andra ord $\alpha = -1$ och $\beta = 3$.
(Uppgift upplevdes som svår. För full poäng krävs ej att visa noggrannhetsordningen.)

(b) Löses med Richardsonextrapolation. Om vi definierar

$$D(h) = \frac{\alpha f(0) + \beta f(h) - \beta f(2h) - \alpha f(3h)}{h^3}$$

så är Richardsonextrapolation blir

$$R(h) = \frac{4D(h/2) - D(h)}{3} = \frac{3\alpha f(0) + 4\beta f(h/2) - 5\beta f(h) - 4\alpha f(3h/2) - 2\beta f(2h) - \alpha f(3h)}{3h^3}$$

Uppgift 4 Datorteknikern Daniela behöver lösa en olinjär ekvation i n variabler, dvs hon behöver bestämma $x_1,\ldots,x_n$ så att $\mathbf{f}(x_1,\ldots,x_n)=0$. Hon har programmerat ett program som beräknar både $\mathbf{f}(x_1,\ldots,x_n)$ och samtidigt beräknar dess Jacobimatris:

```
[f,J]=danielas_program(x)
%  x är en vektor av längd n med de obekanta variablerna
%  f är en vektor av längd n som innehåller f(x_1,...x_n)
%  J är en n x n matris som innehåller Jacobi-matrisen för f
```

(a) **[7p]** Skriv ett program som anropar `danielas_program` och löser det olinjära ekvationssystemet. Daniela vet att en lösning ligger nära noll-vektorn.

(b) **[6p]** Ett anrop till programmet `danielas_program` tar ungefär αn^2 tidsenheter när n är stort när $\alpha = 5$. Ungefär hur stor blir beräkningskostnaden för att lösa det olinjära ekvationssystemet till noggrannhet 10^{-16} när n är stort? Antag att Newtons metod har kvadratisk konvergens och att startgissningen har fel 0.1. Du kan dessutom anta att Jacobi-matrisen är en full matris och att det linjära ekvationssystemet löses med Gauss-eliminering som har beräkningskostnad βn^3 för stora n (med något okänt β-värde).

Lösning:

(a) Vi använder Newtons metod med nollvektorn som startgissning

```
x=zeros(n,1);
TOL=1e-10;
while (norm(dx)>TOL)
   [f,J]=danielas_program(x);
   dx=J\f;
   x=x+dx;
end
```

(b) Kvadratisk konvergens innebär $e_{k+1} \approx a e_k^2$, och vi uppskattar $a=1$. Med antagandet att Newtons metod har kvadratisk konvergens uppskattas felet som $e_{k+1} \approx e_k^2$.

$$
\begin{aligned}
e_0 &= 0.1 && (1)\\
e_1 &= 0.1^2 = 0.01 = 10^{-2} && (2)\\
e_2 &= 0.01^2 = 10^{-4} && (3)\\
e_3 &= (10^{-4})^2 = 10^{-8} && (4)\\
e_4 &= (10^{-4})^2 = 10^{-16} && (5)
\end{aligned}
$$

Vi behöver alltså cirka 4 iterationer för att felet ska vara storleksordning 10^{-16}. Den totala beräkningskostaden blir alltså:

$$4\beta n^3 + 5\cdot 4n^2 + \gamma n$$

där γn motsvarar beräkningstiden för att summera två vektorer av längd n.