

TENTAMEN I GRUNDKURS I NUMERISKA METODER - DEL 2

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
```

- [6p]** Beräkna integralen med trapetsregeln med $h = 0.1$, för komplicerade beräkningar kan du ta hjälp av utdata från programmet.
- [6p]** Använd Richardsonextrapolation med $h = 0.1$ och $h = 0.05$ för att bestämma en noggrannare lösning.

Uppgift 2 Vi vill lösa randvärdesproblemet

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

- [8p]** Skriv ett MATLAB-program som löser problemet med matrismetoden (finita differenser) med $N = 10$ intervall.
- [4p]** Utvidga ditt program så att det även beräknar $\int_0^1 y(x) dx$.

Var god vänd!

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.

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

```
[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).