Norges teknisk–
naturvitenskapelige universitet
Institutt for matematiske fag

Faglig kontakt under eksamen:
Anne Kværnø, tlf. 93542

EKSAMEN I FAG SIF5040 NUMERISKE METODER
Mandag 8. mai 2000
Tid: 09:00–14:00

Hjelpemidler: C1 – Alle trykte og håndskrevne hjelpemidler tillatt.
Alle kalkulatortyper tillatt.

Sensuren faller i uke 22.

Besvarelsen skal inneholde så mange mellomregninger at det tydelig går fram hvilke metoder og mellomresultater som er anvendt.

Bakerst i oppgavesettet finnes to vedlegg, som det kan være en idé å se over først.

Oppgave 1 Finn interpolasjonspolynomet $p(x)$ av lavest mulig grad som interpolerer funksjonen

$$f(x) = \frac{1}{(x-0.3)^2 + 0.1}, \qquad 0 \le x \le 0.4$$

i x-verdiene 0.0, 0.2 og 0.4.

Finn en øvre grense for feilen $|f(x) - p(x)|$.

Fasit: Tabellen over dividerte differenser blir

x_i	$f[x_i]$	$f[,]$	$f[,,]$
0	5.2632		
		19.1388	
0.2	9.090		−47.8469
		0.0	
0.4	9.090		

Newtons interpolasjonspolynom blir da

$$p(x) = 5.2632 + 19.1388x - 47.8469x(x-0.2) = \underline{-47.8469x^2 + 28.7081x + 5.2632}$$

Samme svaret kan oppnås f.eks. ved bruk av Laplaces interpolasjonsformel.
Siden nodene er likt fordelt, vil

$$|p(x) - f(x)| \leq \frac{1}{4(n+1)} \max_{a \leq x \leq b} |f'''(x)| h^{n+1} = \frac{1}{12} \cdot 1500 \cdot 0.2^3 = \underline{1.0}$$

Oppgave 2 Gitt integralet

$$I = \int_0^{0.4} \frac{1}{(x-0.3)^2 + 0.1} dx. \tag{1}$$

a) Bruk trapesformelen, med uniform skrittlengde $h = 0.1$ for å finne en tilnærmelse T til integralet.

Finn en øvre grense for feilen $|I - T|$.

Fasit: Trapesformelen gir, med $f(x) = 1/((x-0.3)^2 + 0.1)$

$$T = h\,(0.5\,f(0) + f(0.1) + f(0.2) + f(0.3) + 0.5\,f(0.4)) = \underline{3.3411}$$

Feilen er

$$|T - I| \leq \frac{1}{12}(b-a)h^2 \max_{a \leq x \leq b} |f''(x)| = \frac{1}{12} \cdot 0.4 \cdot 0.1^2 \cdot 200 = \underline{6.7 \cdot 10^{-2}}$$

b) Hvor liten må skrittlengden h være for at vi er garantert en feil på mindre enn $2 \cdot 10^{-2}$. Bruker vi trapesformelen med denne skrittlengden, blir den relle feilen $|I - T| \approx 7 \cdot 10^{-3}$, altså langt mindre enn hva som ble forlangt. Hva skyldes det?

Fasit: Vi ønsker at

$$\begin{aligned}
|I - T| &= \frac{1}{12} \cdot 0.4 \cdot h^2 \cdot 200 \leq 2 \cdot 10^{-2} \\
\Rightarrow h^2 &\leq \frac{2 \cdot 10^{-2} \cdot 12}{0.4 \cdot 200} = 3 \cdot 10^{-3} \\
\Rightarrow h &\leq 5.48 \cdot 10^{-3}
\end{aligned}$$

Men, siden vi skal ha lik skrittlengde, må nødvendigvis $n = 0.4/h$ bli et heltall, så vi får

$$\underline{h = 0.05}$$

På hvert delintervall får vi en feil $-h^3 f''(\xi_i)/12$, hvor $x_i < \xi_i < x_{i+1}$, og total feil er summen av disse feilene. Og siden $|f''(\xi_i)|$ for det meste er langt mindre enn 200, så vil også total feil være mindre enn det noe pessimistiske anslaget som er gjort.

c) Denne oppgaven går ut på å lage en adaptiv trapes-formel for integralet

$$I = \int_a^b f(x) dx,$$

dvs. at vi ønsker å kunne variere skrittlengden h over intervallet $[a, b]$, og samtidig sikre at den totale feilen er mindre eller omtrent lik en gitt feiltoleranse.

i). Anta at f'' varierer lite over intervallet $[a,b]$. La

$$T_1 = \frac{b-a}{2}(f(a)+f(b)), \qquad T_2 = \frac{b-a}{4}(f(a)+2f(\frac{a+b}{2})+f(b)).$$

Vis at

$$|I - T_2| \approx \frac{1}{3}|T_2 - T_1|$$

ii). Forklar hvordan dette kan brukes til å lage en adaptiv algoritme, dvs. en algoritme der total feil automatisk blir nogenlunde likt fordelt over delintervallene (tilsvarende adaptiv Simpsons metode, men nå basert på trapesformelen).

iii). Anvend algoritmen på integralet (1), med feiltoleransen $2 \cdot 10^{-2}$.

Fasit: i). Anta ay $f'' \approx C$. Da har vi:

$$I - T_1 \approx -\frac{1}{12}(b-a)^3 C$$
$$I - T_2 \approx -\frac{1}{12}\frac{(b-a)^3}{4} C$$

slik at

$$I - T_2 \approx \frac{1}{3}(T_2 - T_1)$$

Det er klart at vi da kan bruke $I \approx T_2 + \frac{1}{3}(T_2 + T_1)$ som en bedre appoksimasjon til I enn T_2.

ii). Vi får en rekursiv algoritme, der vi regner ut T_1, T_2 og feilestimatet. Hvis feilestimatet er mindre enn toleransen ε aksepteres T_2, eventuelt $T_2 + (T_2 - T_1)/3$ som en god nok tilnærmelse, hvis ikke deles intervallet i to like deler, og prosedyren gjentas, men nå med halve toleransen på hvert delområde. En enkel pseudo-kode, hvor ingen feilskjekking er tatt med, vil bli

function I_{app} = trapes(f, a, b, ε)
$c = (b+a)/2$
$T_1 = (b-a)/2 \cdot (f(a)+f(b))$
$T_2 = (b-a)/4 \cdot (f(a)+2f(c)+f(b))$
$feil = (T_2 - T_1)/3$
if $|feil| \leq \varepsilon$
  $I_{app} = T_2 + feil$
else
  I_1 = trapes(f, a, c, $\varepsilon/2$)
  I_2 = trapes(f, c, b, $\varepsilon/2$)
  $I_{app} = I_1 + I_2$
end

iii). Brukt på integralet i oppgaven, ender vi opp med følgende:

<table>
<tr><td colspan="3">$a = 0.0,\ b = 0.4,\ \varepsilon = 2 \cdot 10^{-2}$
$T_1 = 2.8708,\ T_2 = 3.2536,\ feil = 0.13$</td></tr>
<tr><td rowspan="2">$a = 0.0,\ b = 0.2,\ \varepsilon = 1 \cdot 10^{-2}$
$T_1 = 1.4354,\ T_2 = 1.4320$
$feil = 1.1 \cdot 10^{-3}$
$I_{app}(0.0, 0.2) = 1.4308$</td><td colspan="2">$a = 0.2,\ b = 0.4,\ \varepsilon = 1 \cdot 10^{-2}$
$T_1 = 1.8182,\ T_2 = 1.9091$
$feil = 3.0 \cdot 10^{-2}$</td></tr>
<tr><td>$a = 0.2,\ b = 0.3$
$\varepsilon = 5 \cdot 10^{-3}$
$T_1 = 0.9545,\ T_2 = 0.9651$
$feil = 3.5 \cdot 10^{-3}$
$I_{app}(0.2, 0.3) = 0.9686$</td><td>$a = 0.3,\ b = 0.4$
$\varepsilon = 5 \cdot 10^{-3}$
$T_1 = 0.9545,\ T_2 = 0.9651$
$feil = 3.5 \cdot 10^{-3}$
$I_{app}(0.3, 0.4) = 0.9686$</td></tr>
</table>

Siden $f(x)$ er symmetrisk rundt $x = 0.3$, vil resultatene for de to intervallene $[0.2, 0.3]$ og $[0.3, 0.4]$ bli like.
Vi får at

$$I \approx I_{app}(0.0, 0.2) + I_{app}(0.2, 0.3) + I_{app}(0.3, 0.4) = \underline{3.3680}$$

Oppgave 3 Gitt følgende differensialligning

$$y'' = y^3 - y \cdot y', \qquad 0 \le x \le 1 \qquad y(0) = \frac{1}{2}$$

Skriv om dette som et system av første ordens differensialligninger.

Systemet skal løses med en 2.ordens Runge-Kutta metode med skrittlengde $h = 0.2$. Tabellen under gir resultatene av dette for to ulike valg av $y'(0)$, nemlig 0 og $-1/12$:

i	0	1	2	3	4	5
x_i	0.0	0.2	0.4	0.6	0.8	1.0
y_i	0.5000	0.5025	0.5095	$--$	0.5365	0.5563
y'_i	0.0000	0.0238	0.0459	$--$	0.0884	0.1101
y_i	0.5000	0.4867	$--$	0.4760	0.4774	0.4828
y'_i	-0.0833	-0.0530	$--$	-0.0039	0.0170	0.0364

Her er $y_i \approx y(x_i)$.
Fyll ut de åpne plassene i tabellen.

Egentlig ønsker vi å løse et to punkts randverdiproblem, der $y(1) = 1/3$. Basert på resultatene i tabellen, hvordan vil du velge $y'(0)$ for å få en bedre tilnærmelse til løsningen i endepunktet?

Fasit: La $v_1(x) = y(x)$, $v_2(x) = y'(x)$, og systemet blir

$$\begin{aligned} v_1' &= v_2, & \qquad v_1(0) &= 1/2 \\ v_2' &= v_1^3 - v_2 v_2 & \qquad v_2(0) &= -- \end{aligned}$$

I det etterfølgende lar vi $v_{1,i} = y_i \approx y(x_i) = v_1(x_i)$ og $v_{2,i} = y'_i \approx y'(x_i) = v_2(x_i)$, og $\mathbf{v}_i = (v_{1,i}, v_{2,i})^T$. Starter vi med øverste rad, og bruker forbedret Euler (lign. (10), s. 386 i Cheney &

Kincaid), får vi:

$$\begin{aligned}
\mathbf{v}_2 &= \begin{pmatrix} 0.5095 \\ 0.0459 \end{pmatrix}, \quad \mathbf{K}_1 &&= h \cdot \mathbf{f}(x_2, \mathbf{v}_2) = \begin{pmatrix} 9.1800 \cdot 10^{-3} \\ 2.1775 \cdot 10^{-2} \end{pmatrix} \\
\mathbf{v}_2 + \mathbf{K}_1 &= \begin{pmatrix} 0.5187 \\ 0.0677 \end{pmatrix}, \quad \mathbf{K}_2 &&= h \cdot \mathbf{f}(x_2 + h, \mathbf{v}_2 + K1) = \begin{pmatrix} 1.3535 \cdot 10^{-2} \\ 2.0888 \cdot 10^{-2} \end{pmatrix} \\
\mathbf{v}_3 &= \mathbf{v}_2 + 0.5 \cdot h \cdot (\mathbf{K}_1 + \mathbf{K}_2) = \begin{pmatrix} 0.5209 \\ 0.0672 \end{pmatrix}
\end{aligned}$$

slik at altså

$$\underline{y_3 = 0.5209, \quad y_3' = 0.0672}$$

Tilsvarende, for det nedre valget av startverdier, får vi

$$\underline{y_2 = 0.4789, \quad y_2' = -0.0269}$$

Vi har allerede gjort alle forberedelsene for å løse et to punkts randverdi problem med skytemetoden, i og med at ligningen er løst med to forskjellige valg av y_0'. Bruker vi formel (2), s. 505 i Cheney & Kincaid, med

$$z_1 = 0, \quad \phi(z_1) = 0.5563, \quad z_2 = -1/12, \quad \phi(z_2) = 0.4828 \quad \text{og} \quad \beta = 1/3$$

får vi

$$\underline{y_0' = z_3 = -0.2528}$$

Oppgave 4 Gitt datasettet

x_i	0	1	2	3	4
y_i	1.5	2.5	3.5	5.0	7.5

og funksjonen

$$y(x) = C \cdot \mathrm{e}^{Ax} \tag{2}$$

Oppgaven går ut på å beregne konstantene C og A slik at funksjonen best mulig tilpasses dataene.

a) Vis at (2) kan skrives som

$$\ln y = A \ln x + \ln C$$

og bruk lineær minste kvadraters metode for å beregne konstantene A og C.

Fasit: Her er det en feil i oppgaveteksten, noe som ble oppgitt på eksamen. Taes den naturlige logaritmen på begge sider av ligningen, fåes

$$\ln y = Ax + \ln C$$

og følgende tabell

x_i	0	1	2	3	4
$\ln y_i$	.4055	0.9163	1.2528	1.6094	2.0149

Normalligningene er gitt ved (med $B = \ln C$)

$$\begin{aligned} \left(\sum_{i=0}^{4} x_i^2\right) A + \left(\sum_{i=0}^{4} x_i\right) B &= \sum_{i=0}^{4} x_i \ln y_i \\ \left(\sum_{i=0}^{4} x_i\right) A + 5B &= \sum_{i=0}^{4} \ln y_i \end{aligned}$$

eller

$$\begin{aligned} 30A + 10B &= 16.3097 \\ 10A + 5B &= 6.1989 \end{aligned}$$

med løsning

$$\underline{A = 0.3912}, \quad B = 0.4574, \quad \underline{C = e^B = 1.5799}$$

b) Ved å bruke lineær minste kvadraters metode i **a)** har vi minimalisert

$$\sum_{i=0}^{4} (\ln y_i - A \ln x_i - \ln C)^2$$

Anta at vi istedet ønsker å minimalisere feilen

$$E(A, C) = \sum_{i=0}^{4} (y_i - C \cdot \mathrm{e}^{Ax_i})^2$$

direkte.
Skriv et MATLAB-program som beregner A og C ved bruk av funksjonen `fminsearch`, se vedlegg 2 for en beskrivelse.

Fasit: En mulighet er følgende MATLAB-funksjon

```
function f = f4(par)
A = par(1);
C = par(2);
x = [0,1,2,3,4];
y = [1.5,2.5,3.5,5.0,7.5];
f = 0;
for i=1:5
    f = f+(y(i)-C*exp(A*x(i)))^2;
end
```

som blir kalt ved

```
>> [res,fval] = fminsearch('f4',[0.3912,1.5799])
```

Her har vi brukt resultatene fra punkt **a)** som startverdi.

Oppgave 5 Gitt Helmholtz' ligning

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + a \cdot u = 0, \qquad 0 \le x, y \le 1, \qquad a \text{ er en konstant} \tag{3}$$

med randbetingelser

$$
\begin{aligned}
u(x,0) &= \sin(\pi x), && 0 \le x \le 1 \\
u(0,y) &= \sin(\pi y), && 0 \le y \le 1 \\
\frac{\partial u}{\partial y}(x,1) &= 0, && 0 \le x \le 1 \\
u(1,y) &= 0, && 0 \le y \le 1
\end{aligned}
$$

Ligningen skal løses numerisk med fem-punkts formelen, og vi bruker skrittlengde h i begge retninger. I det følgende er $x_i = ih$, $y_j = jh$ og $u_{ij} \approx u(x_i, y_j)$.

a) Skriv opp ligningene som må løses for å finne u_{ij} i alle gitterpunktene der løsningen er ukjent.

Dette systemet skal løses ved hjelp av SOR. Sett $h = 1/3$, $a = -1$ og utfør en SOR-iterasjon med relaksasjonsparameter $\omega = 1.2$. Bruk $u_{ij}^{(0)} = 0.5$ som startverdi.

Fasit: I det etterfølgende bruker vi $n = 1/h$. Ukjente punkter er alle *indre* gitterpunkter, samt gitterpunktene på randa $y = 1$.

For de indre punktene bruker vi fempunktsformelen direkte, slik at

$$
(4 - h^2 a)u_{ij} - (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) = 0, \qquad i,j = 1,2,\cdots,n-1
$$

på randa $y = 1$ bruker vi trikset med ***falsk rand***. Vi bruker $\partial u/\partial y(x_i, 1) \approx (u_{i,n+1} - u_{i,n-1})/(2h)$, der strengt tatt løsningen $u_{i,n+1}$ ligger utenfor området. Men brukes dette sammen med fempunktsformelen, får vi:

$$
(4 - h^2 a)u_{in} - (u_{i+1,n} + u_{i-1,n} + 2u_{i,n-1}) = 0, \qquad i = 1,2,\cdots,n-1
$$

Fra de øvrige randbetingelsene har vi

$$
u_{i,0} = \sin(\pi x_i), \quad u_{0,j} = \sin(\pi y_j), \quad u_{n,j} = 0, \qquad i,j = 1,2,\cdots,n
$$

Tilsammen utgjør disse ligningssystemet som må løses for å finne løsningen i gitterpunktene.

SOR-iterasjoner utført på dette systemet gir:

$$
\begin{aligned}
u_{i,j}^{(k+1)} &= \frac{\omega}{A}(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j-1}^{(k+1)} + u_{i,j+1}^{(k)}) + (1-\omega)u_{i,j}^{(k+1)}, \quad i = 1,2,\cdots,n, \qquad j = 1,2,\cdots,n-1 \\
u_{i,n}^{(k+1)} &= \frac{\omega}{A}(u_{i+1,n}^{(k)} + u_{i-1,n}^{(k+1)} + 2u_{i,n-1}^{(k+1)}) + (1-\omega)u_{i,n}^{(k+1)}, \qquad i = 1,2,\cdots,n
\end{aligned}
$$

der $A = 4 - ah^2$.

For $h = 1/3$, $a = -1$ og $\omega = 1.2$, slik at $n = 3$ og $A = 37/9$, $\omega/A = 0.2919$ og $1 - \omega = -0.2$ får vi:

$$
\begin{aligned}
u_{11}^{(1)} &= 0.2919(u_{21}^{(0)} + \sin(\pi/3) + u_{12}^{(0)} + \sin(\pi/3)) - 0.2u_{11}^{(0)} && = 0.6975 \\
u_{21}^{(1)} &= 0.2919(0 + u_{11}^{(1)} + u_{22} + \sin(2\pi/3)) - 0.2u_{21}^{(0)} && = 0.5023 \\
u_{12}^{(1)} &= 0.2919(u_{22}^{(0)} + \sin(2\pi/3) + u_{13}^{(0)} + u_{11}^{(1)}) - 0.2u_{12}^{(0)} && = 0.6483 \\
u_{22}^{(1)} &= 0.2919(0 + u_{21}^{(1)} + u_{23} + u_{21}^{(1)}) - 0.2u_{22}^{(0)} && = 0.3818 \\
u_{13}^{(1)} &= 0.2919(u_{23}^{(0)} + 0 + 2u_{12}^{(1)}) - 0.2u_{13}^{(0)} && = 0.4244 \\
u_{23}^{(1)} &= 0.2919(0 + u_{13}^{(1)} + 2u_{22}^{(1)}) - 0.2u_{13}^{(0)} && = 0.2464
\end{aligned}
$$

b) For enkelthets skyld vil vi i resten av oppgaven bruke en Gauss-Seidel type iterasjon, modifisert slik at løsningene på samme rad oppdateres samtidig. Iterasjonskjemaet for de indre punktene blir

$$u_{ij}^{(k+1)} = \frac{1}{A}(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)})$$

hvor uttrykket for A burdte være kjent fra punkt **a)**.

Si litt om hvilke fordeler og ulemper et slikt iterasjonsskjema har, sammenlignet med vanlig Gauss-Seidel.

Skriv et fullstendig MATLAB-program for løsning av (3) for en valgt verdi av a. Antall gitterpunkter inngår som en parameter som skal kunne velges, men bruk samme antall gitterpunkter i begge retninger. Bruk iterasjonskjemaet slik det er beskrevet ovenfor, men husk å ta hensyn til at ligningene blir litt anderledes på randa $y = 1$. Iterasjonene skal avsluttes når $|u_{ij}^{(k+1)} - u_{ij}^{(k)}| < 10^{-4}$ for alle i og j. Programmet skal også lage et plott av løsningen.

Fasit: Ulempe: Iterasjonsskjemaet bruker ikke alle de sist oppdaterte verdiene, vi må derfor gå ut i fra at det konvergerer saktere. Fordel: Man har mulighet for å utføre iterasjonene som vektor-operasjoner, man kan gjøre iterasjonen for en hel rad av gangen. I MATLAB er dette raskere enn å gjøre samme operasjonen i en indre løkke (over i).

Her er et forslag til MATLAB program:

```
% Setter noen konstanter
a = -1
n = 20
h = 1/n;
A = 4-a*h^2;

% Allokerer plass, leser inn startverdier
x = [0:h:1]; y = x';
u = 0.5*ones(n+1,n+1);

% Setter randverdier
u(:,1) = sin(pi*y);
u(1,:) = sin(pi*x);
u(:,n+1) = 0;
% Starter iterasjonene
feil = 1
while feil>1.e-4

  % Indre punkter
  for j=2:n
    feil = 0;
    uny=(u(j,3:n+1)+u(j,1:n-1)+u(j+1,2:n)+u(j-1,2:n))/A;
    nyfeil = max(abs(uny-u(j,2:n)));
    feil = max(feil,nyfeil);
    u(j,2:n)=uny;
  end
  % Randa y=1
  uny=(u(n+1,3:n+1)+u(n+1,1:n-1)+2*u(n,2:n))/A;
  nyfeil = max(abs(uny-u(n+1,2:n)))
```

```
  feil = max(feil,nyfeil)
  u(n+1,2:n)=uny;
end

% Plotter loesningen
surf(x,y,u);
```