Lösningar till Tentamen i Beräkningsvetenskap II, 5.0 hp, 2012-03-09

Del A

1. (a) För att anpassa ett polynom som går genom tre punkter behövs ett andragradspolynom. Newtons interpolationsansats ger

$$f(x) = b_1 + b_2(x - x_1) + b_3(x - x_1)(x - x_2),$$

där b_1, b_2 och b_3 entydigt bestäms av mätvärderna. Låt $x_1 = 1.0$, $x_2 = 1.5$, $x_3 = 2$ och $f(x_1) = 0.26$, $f(x_2) = 0.29$, $f(x_3) = 0.31$ från den givna tabellen i uppgiften. Använd sedan ansatsen $f(x_1)$ för att räkna ut värdet på b_1

$$0.26 = f(x_1) = b_1 \Rightarrow b_1 = 0.26,$$

$f(x_2)$ för att räkna ut värdet på b_2

$$\begin{aligned} 0.29 = f(x_2) &= b_1 + b_2(x_2 - x_1) = 0.26 + b_2(1.5 - 1.0) \\ \Rightarrow b_2 &= \frac{0.29 - 0.26}{1.5 - 1.0} = 0.06, \end{aligned}$$

och $f(x_3)$ för att räkna ut värdet på b_3

$$\begin{aligned} 0.31 = f(x_2) &= b_1 + b_2(x_3 - x_1) + b_3(x_3 - x_1)(x_3 - x_2) \\ &= 0.26 + 0.06(2.0 - 1.0) + b_3(2.0 - 1.0)(2.0 - 1.5 \\ \Rightarrow b_3 &= \frac{0.31 - 0.26 - 0.06(2.0 - 1.0)}{(2.0 - 1.0)(2.0 - 1.5)} = -0.02. \end{aligned}$$

Man kan också formulera problemet genom att ställa upp interpolationsvillkoren som ett ekvationssystem med undertriangulär koeffiecientmatris,

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & (x_2 - x_1) & 0 \\ 1 & (x_3 - x_1) & (x_3 - x_2)(x_3 - x_1) \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \end{pmatrix},$$

som med insatta värden blir:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0.5 & 0 \\ 1 & 1.0 & 0.5 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 0.26 \\ 0.29 \\ 0.31 \end{pmatrix}.$$

Detta system kan lösas genom framåtsubstitution och vi får samma lösning som med det tidigare redovisade angreppssättet. Newtons interpolationspolynom blir:

$$f(x) = 0.26 + 0.06\,(x - 1.0) - 0.02\,(x - 1.0)\,(x - 1.5)\,.$$

(b) För att interpolera med ett polynom genom 30 punkter behövs ett polynom av grad 29. Detta riskerar att ge upphov till det så kallade Runges fenomen, alltså att polynomet kommer att ge en bra approximation i mitten av intervallet men oscillera kraftigt närmare ränderna.

2. (a) Tillämpa Euler framåt på testekvationen, $y'(t) = \lambda y(t)$, ger

$$y_{i+1} = y_i + h\lambda y_i \Rightarrow y_{i+1} = y_i\,(1 + h\lambda)\,.$$

Av detta följer att

$$y_i = y_0 \left(1 + h\lambda\right)^i$$

För stabilitet krävs alltså att $\left(1 + h\lambda\right)^i \to 0$ då $h \to 0$. Stabilitetsvillkoret blir därför

$$|1 + h\lambda| < 1.$$

(b) Styva problem kännetecknas av att när explicita metoder tillämpas på sådana problem, så krävs det mycket liten steglängd för att metoden ska vara stabil. Mycket liten steglängd medför att det blir mycket lång exekveringstid. Eftersom Euler framåt är explicit, så är det alltså olämpligt att använda den metoden om problemet är styvt.

3. (a) Funktionsfilen för högerledet:

```
function u = f(t,y)
  u = cos(y+t);
```

Huvudfilen där ODE-problemet löses genom anrop av `ode45`:

```
y0 = 1;  % Begynnelsevillkor
t_start = 0; % Startid
t_stop = 10; % Sluttid
Tspan = [t_start t_stop];

[t y] = ode45(@f, Tspan, y0);
```

(b) Det sökta ordet är “adaptivt”.

4. (a) För en Monte Carlo - metod gäller att $fel \approx C\frac{1}{\sqrt{N}}$. Därav följer att $fel_{10000} \approx \frac{C}{\sqrt{10000}}$ och $fel_{20000} \approx \frac{C}{\sqrt{20000}}$. Vi får $\frac{fel_{10000}}{fel_{20000}} \approx \frac{\frac{1}{\sqrt{10000}}}{\frac{1}{\sqrt{20000}}} = \sqrt{2}$.

   (b) Stokastiska

Del B

5. Vi måste omformulera den matematiska modellen till den form som förutsätts i de numeriska ODE-lösare som har ingått i kursen. Till att börja med skriver vi om pendelns ekvation som ett första ordningens system av ODE. För detta inför vi en ny variabel:

$$v(t) = \theta'(t). \tag{1}$$

Insättning i pendelns ekvation ger:

$$v'(t) = -2\frac{r'(t)}{r(t)}v(t) - \frac{g}{r(t)}\sin\left(\theta(t)\right). \tag{2}$$

Ekvationerna (1) och (2) utgör ett första ordningens system av ODE, men systemet är inte på den form som de numeriska ODE-lösarna förutsätter. Dels har

vi nu två ekvationer men tre obekanta, dels finns det en derivata i högerledet i (2). Vi kan hantera båda dessa komplikationer genom att kombinera (1) och (2) med vinschningsstrategin:

$$\begin{aligned} \theta'(t) &= v(t) \\ v'(t) &= -2\frac{f(t,\theta(t),v(t))}{r(t)}v(t) - \frac{g}{r(t)}\sin(\theta(t)) \\ r'(t) &= f(t,\theta(t),v(t)) \end{aligned}$$

Ekvationssystemet är nu på den form som förutsätts i de numeriska ODE-lösare som har ingått i kursen. Det syns tydligare om vi skriver systemet på vektorform:

$$y'(t) = F(t, y(t)), \tag{3}$$

där

$$y(t) = \begin{pmatrix} \theta(t) \\ v(t) \\ r(t) \end{pmatrix}$$

och $F(t, y(t))$ är den vektor som utgörs av högerleden i ekvationssystemet ovan. För simuleringen antar vi att det finns begynnelsedata $\theta(0)$, $v(0)$ och $r(0)$.

För val av metod finns två förutsättningar angivna i uppgiften. Den ena är att noggrannhetsordningen måste vara större än 1. Den andra förutsättningen är att modellen inte är styv. Bland de metoder som har ingått i kursen uppfyller alla utom de två eu-

lermetoderna förutsättningen om noggrannhetsordning större än 1. Förutsättningen att modellen inte är styv innebär att vi kan välja en explicit metod utan att det blir nödvändigt att av stabilitetsskäl ta extremt små tidssteg. Tidsstegets storlek kommer i stället att avgöras av noggrannhetskravet, oavsett om vi väljer en explicit eller en implicit metod. Om vi jämför Heuns metod och Trapetsmetoden som båda har noggrannhetsordning 2, så bör dessa då behöva ungefär samma storlek på steglängden, det vill säga ungefär lika många beräkningspunkter. Därmed är Heuns metod att föredra, eftersom den är explicit och därför kräver mindre beräkningsarbete per punkt än Trapetsmetoden som är implicit. Slutligen är klassiska Runge-Kuttas metod explicit och har noggrannhetsordning 4. Den ger något mera beräkningsarbete per punkt än Heuns metod, men detta uppvägs med råge av att det går uppnå den önskade noggrannheten med betydligt färre steg när noggrannhetsordningen är större.

Vid bedömningen av studenternas lösningar godkände vi både användning av Heuns metod och klassiska Runge-Kuttas metod samt lösningar där Matlabs kommando `ode45` användes, under förutsättning att studenten gav rimliga argument för sitt val. Det krävdes att argumentationen tog upp båda de förutsättningar som nämndes i uppgiften. Däremot krävdes beträf-

fande styvhetsaspekten inte så utförlig argumentation som ovan, utan även betydligt mera kortfattade påståenden godkändes, exempelvis ”det går bra med en explicit metod eftersom problemet inte är styvt”.

För godkänd lösning krävdes att studenten visade formler eller matlabkod för den valda metoden, specifikt för systemet (3). Enklast är att visa detta för systemet uttryckt på vektorform. Observera att det inte går att tillämpa de nämnda metoderna på varje ekvation för sig, eftersom ekvationen då inte är på den form som förutsätts i metoden. Den numeriska metoden (eller matlabkommandot `ode45`) måste alltså appliceras på systemet som helhet för att det ska fungera.

6. Vi beskriver först en algoritmidé, argumenterar sedan för varför den är lämplig och visar slutligen de formler som behövs för att utveckla algoritmen mera detaljerat. Algoritmidén är:

   - Finn heltalet i så att
     $\texttt{xcoord(i)} \leq x \leq \texttt{xcoord(i+1)}$.
   - Bestäm interpolationspolynomet $p_s(x)$ genom de två punkterna
     $(\texttt{xcoord(i)}, \texttt{s(i)})$, $(\texttt{xcoord(i + 1)}, \texttt{s(i + 1)})$.
   - Bestäm interpolationspolynomet $p_n(x)$ genom de två punkterna

$(\texttt{xcoord(i)}, \texttt{n(i)})$, $(\texttt{xcoord(i + 1)}, \texttt{n(i + 1)})$.

- Om $p_s(x) \leq y \leq p_n(x)$ så returneras "sann" annars returneras "falsk".

Vi argumenterar nu för att angreppssättet är lämpligt. För att avgöra om (x, y) ligger inom sjöns område behöver vi på ett eller annat sätt få en uppfattning om var sjöns strandkant finns mellan de givna punkterna. Kurvanpassning är det angreppssätt i kursen som passar för detta. Eftersom vi har två punktmängder, för södra respektive norra stranden, så kan vi tillämpa kurvanpassning på varje punktmängd för sig. Vi har att välja mellan de alternativ för kurvanpassning, som har ingått i kursen. De innebär alla att vi anpassar ett polynom till data. De övergripande angreppssätten är minstakvadratanpassning och interpolation. Eftersom de givna punkterna utgör vår markering av strandkonturen vill vi att kurvan ska gå genom de givna punkterna. Alltså är interpolation det lämpliga i detta fall (med minstakvadratanpassning får vi ett polynom som inte går igenom de givna punkterna). Interpolation genom samtliga de givna punkterna riskerar att ge upphov till Runges fenomen (kraftiga oscillationer) och därför är det lämpligare med någon form av styckvis interpolation. I det här fallet har vi inte behov av en visuellt tilltalande kurva, så det är olämpligt att lägga ned det extra

beräkningsarbete som kubiska splines skulle innebära. Rent allmänt kan man förmoda att strandkonturen är oregelbunden och inte ser ut som ett polynom oavsett gradtal. Därför är det lämpligast att här använda sig av *styckvis lineär interpolation*, som är det minst beräkningskrävande alternativet när det gäller styckvis interpolation. Algoritmidén ovan bygger på detta angreppssätt.

Nu återstår att härleda de formler för $p_s(x)$ och $p_n(x)$ som angreppssättet med styckvis linär interpolation ger upphov till. Vi börjar med $p_s(x)$. Newtons interpolationsansats blir i detta fall:

$$p_s(x) = c_0 + c_1 \left(x - \texttt{xcoord(i)}\right).$$

Interpolationsvillkoren är:

$$\begin{aligned} p_s(\texttt{xcoord(i)}) &= \texttt{s(i)} \\ p_s(\texttt{xcoord(i+1)}) &= \texttt{s(i+1)} \end{aligned}$$

Om vi sätter in ansatsen i interpolationsvillkoren och löser det resulterande ekvationssystemet, så finner vi att:

$$p_s(x) = \texttt{s(i)} + \frac{\texttt{s(i+1)} - \texttt{s(i)}}{\texttt{xcoord(i+1)} - \texttt{xcoord(i)}} \left(x - \texttt{xcoord(i)}\right).$$

Med analogt tillvägagångssätt får vi att:

$$p_n(x) = \texttt{n(i)} + \frac{\texttt{n(i+1)} - \texttt{n(i)}}{\texttt{xcoord(i+1)} - \texttt{xcoord(i)}} \left(x - \texttt{xcoord(i)}\right).$$

Den detaljerade algoritmen blir därmed, efter insättning av dessa formler i algoritmskissen ovan:

- Finn heltalet i så att
  $\texttt{xcoord(i)} \leq x \leq \texttt{xcoord(i+1)}$.
- Om
  $\texttt{s(i)} + \frac{\texttt{s(i+1)} - \texttt{s(i)}}{\texttt{xcoord(i+1)} - \texttt{xcoord(i)}} (x - \texttt{xcoord(i)}) \leq y \leq \texttt{n(i)} + \frac{\texttt{n(i+1)} - \texttt{n(i)}}{\texttt{xcoord(i+1)} - \texttt{xcoord(i)}} (x - \texttt{xcoord(i)})$
  så returneras ”sann” annars returneras ”falsk”.

För betyg 5 godkändes lösningar som innehöll rätt algoritmidé och där det fanns väl genomförd argumentation för att styckvis interpolation var det lämpligaste angreppssättet. Detta innebär att vi även godkände lösningar där kubiska splines rekommenderades, under förutsättning att rimliga argument för detta framfördes. Vidare godkändes lösningar där algoritmidén fanns väl utvecklad, även om de detaljerade formlerna för p_s och p_n inte härleddes. Lösningar som innehöll rätt algoritmidé, men där minstakvadratanpassning rekommenderades med rimlig argumentation, godkändes för betyg 4. En mycket vanlig lösning var att anta att x ingick i `xcoord` eller var nära något värde i `xcoord` och därför kunde ersättas med motsvarande `xcoord`-värde. Detta godkändes inte, eftersom de alternativ som bygger på kurvanpassning är lämpligare inte minst därför att de fungerar utan

att vi behöver förutsätta att de givna punkterna ligger mycket tätt.