Uppsala universitet
Institutionen för informationsteknologi
Beräkningsvetenskap

Lösningar till tentamen i *Numeriska metoder och simulering* 5.0 hp, 2015-10-21

Uppgifter som testar måluppfyllelse för betyg 3

1. (a) En funktion som beräknar ODE-problemets högerled:

```
function yprim = myODE( t, y)
   yprim = sin(y^2);
end
```

Exempel på ett skript som löser problemet med de indata som angavs i uppgiften:

```
[t, y] = ode45(@myODE, [0 2], 0.5);
plot(t,y)
```

(b) Exempel på ett program enligt uppgiften:

```
N = input('Antal simuleringar av partikelbana?  ');
resultat = zeros(1,N);

for i = 1:N
      resultat(i) = partikel(2);
end

dmean = mean(resultat);
disp(['Genomsnittligt avstånd till utgångspunkten '...
      'efter 2 sekunder:  ', ...
      num2str(dmean), ' längdenheter'])
```

2. (a) Deterministisk modell
   (b) Explicit metod

(c) Kancellation

(d) Noggrannhetsordning

3. (a)

$$y_{i+1} = y_i + h\sin(y_{i+1}^2)$$

(b) Kumulativa sannolikhetsfunktionen, representerad som vektor:

$$F = \begin{pmatrix} 0.4 \\ 0.6 \\ 1.0 \end{pmatrix}$$

Slumptalet r väljs som det minsta heltal i så att $F(i) \geq u$. I vårt fall är $u = 0.53$, så $F(1) < u$, $F(2) > u$. Följaktligen blir resultatet att $r = 2$.

4. (a) Se föreläsningsanteckningar.

(b) Euler bakåt tillämpad på testekvationen ($y\prime(t) = \lambda y(t)$) är: $y_{k+1} = y_k + h\lambda y_{k+1}$. Omformulering ger att $y_{k+1} = y_k/(1 - \lambda h)$. Kom ihåg att i testekvationen står y för störningen. För stabilitet krävs att störningen avtar med tiden, det vill säga att $|y_{k+1}| < |y_k|$. Med insättning av ovanstående uttryck för y_{k+1} blir kravet att $|y_k/(1 - \lambda h)| < |y_k|$. Slutsatsen blir att stabilitetsvillkoret för Euler bakåt är:

$$|1 - \lambda h| > 1$$

5. (a) Monte Carlo-metoder kan vara det enda rimliga alternativet för *numerisk integrering över stort antal dimensioner*. Det beror på att för numerisk integrering över stort antal dimensioner kräver en deterministisk metod så många beräkningspunkter att minnesåtgången kan bli en begänsande faktor. Det behövs åtminstone *några* punkter i varje dimension och med n punkter i varje dimension får man ett "nät" med n^d beräkningspunkter. Antalet punkter växer alltså exponentiellt med antal dimensioner. Därtill kommer att noggrannhetsordningen hos en deterministisk metod för numerisk integrering avtar med växande antal dimensioner. Med en Monte Carlo-metod kan man välja antalet beräkningspunkter friare och noggrannhetsordningen är oberoende av antal dimensioner. Därför kan en Monte Carlo-metod för numerisk integrering fungera när antalet dimensioner blir så stort att en deterministisk metod inte lägre är användbar.

(b) Båda metoderna är explicita men problemet är inte styvt, så vi antar att stabilitetsvillkoret inte kommer att vara begränsande. Vi kan då fokusera på noggrannheten. Metoden med högre noggrannhetsordning ger något mera arbete per beräkningspunkt, men den behöver *väsentligt* färre punkter (än metoden med lägre noggrannhetsordning) för att uppfylla toleransen. Sammantaget ger därför metoden med högre noggrannhetsordning (i detta fall klassiska Runge-Kuttas metod) kortast exekveringstid.

Uppgift som testar måluppfyllelse för betyg 4

6. Skriv om problemet som ett första ordningens system av ODE (eftersom denna form förutsätts i Matlabs ODE-lösare). Inför variablerna $y_1(t) = \theta(t)$ och $y_2(t) = \theta'(t)$. Vår ODE kan då skrivas på formen:

$$\begin{pmatrix} y_1'(t) \\ y_2'(t) \end{pmatrix} = \begin{pmatrix} y_2(t) \\ -Ky_2(t) - \frac{g}{L}y_1(t) \end{pmatrix}$$

Vi ska nu implementera ett program Matlab för att lösa dessa ekvationer. Eftersom problemet inte är styvt kan en explicit metod användas utan att man av stabilitetsskäl behöver ta mycket korta steg. Det är då *lämpligare* med en explicit metod än en implicit metod av samma noggrannhetsordning, eftersom den implicita metoden behöver göra mera arbete per tidssteg och alltså ger längre exekveringstid. Eftersom `ode45` bygger på en explicit metod av hög noggrannhetsordning är den ett lämpligt val här. Nedanstående är exempel på ett program som använder `ode45` för att simulera pendelns rörelse baserat på modellen ovan.

Högerledet i ODE-systemet beskrivs som en funktion:

```
function yprim = pendelODE( t, y )
    global g L K

    yprim = [y(2);
             -K*y(2) - (g/L)*sin(y(1))];
end
```

Följande huvudprogram utför simuleringen:

```
% Ge värden på problemparametrar
global g L K
```

```
g = 9.82;
L = 10;
K = 0.1;

% Läs in begynnelsevärden
theta = input('Begynnelsevärde för theta?  ');
thetaprim = input('Begynnelsevärde för derivatan av theta?  ');
y0 = [theta; thetaprim];

% Genomför simuleringen
[t, y] = ode45(@pendelODE, [0 10], y0);

% Plotta vinkeln theta som funktion av tiden
plot(t,y(:,1));
```

Uppgift som testar måluppfyllelse för betyg 5

7. Skriv om den andra ordningens ODE som ett första ordningens system analogt med hur vi gjorde i föregående uppgift. Lägg sedan till vinschningsstrategin, med beteckningen $y_3(t) = r(t)$, så får vi ett första ordningens system med tre ekvationer och tre obekanta:

$$\begin{pmatrix} y_1'(t) \\ y_2'(t) \\ y_3'(t) \end{pmatrix} = \begin{pmatrix} y_2(t) \\ -K\frac{f(t,y_1(t),y_2(t))}{y_3(t)}y_2(t) - \frac{g}{y_3(t)}y_1(t) \\ f(t,y_1(t),y_2(t)) \end{pmatrix}$$

   Argumentationen för val av ODE-lösare i Matlab blir samma som i föregående uppgift. Nedanstående är exempel på ett program som använder `ode45` för att simulera pendelns rörelse baserat på modellen ovan.

   Högerledet i ODE-systemet beskrivs som en funktion:

```
function yprim = vinschODE(t, y)
    global g

    fty = f(t,y(1),y(2));
    yprim = [y(2);
            -2*(fty/y(3))*y(2) - (g/y(3))*sin(y(1));
            fty];
end
```

Följande huvudprogram utför simuleringen:

```
% Ge värden på problemparametrar
global g
g = 9.82;

% Läs in begynnelsevärden
theta = input('Begynnelsevärde för theta?  ');
thetaprim = input('Begynnelsevärde för derivatan av theta?  ');
r = input('Begynnelsevärde på pendelns längd?  ');
y0 = [theta; thetaprim; r];

% Genomför simuleringen
[t, y] = ode45(@vinschODE,[0 10], y0);

% Plotta vinkeln och längden som funktion av tiden
plot(t, y(:,[1 3]))
title('Vinkeln och längden som funktion av tiden')
legend('theta','r')
```