Uppsala universitet
Institutionen för informationsteknologi
Beräkningsvetenskap

Lösningar till tentamen i *Numeriska metoder och simulering* 5.0 hp, 2018-10-26

Uppgifter som testar måluppfyllelse för betyg 3

1. (a) Skriv om differentialekvationen på den form som `ode45` förutsätter: $y'(t) = 3y(t)\,(1 - y(t))$. Skriv sedan en matlabfunktion som beräknar ODE-problemets högerled:

```
function yprim = myODE( t, y)
   yprim = 3*y*(1-y);
end
```

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

```
[t, y] = ode45(@myODE, [0 10], 0.1);
plot(t,y)
```

   (b) Exempel på ett program enligt uppgiften:

```
N = input('Antal punkter?  ');
resultat = zeros(1,N);
for i = 1:N
      resultat(i) = g(rand);
end
I = mean(resultat)
```

   Ett kortare program som löser samma uppgift:

```
N = input('Antal punkter?  ');
I = mean(g(rand(1,N)))
```

2. (a) Deterministisk metod

   (b) Normalisering

(c) Styv ODE

(d) Implicit metod

3. (a) Skriv om differentialekvationen på den form som förutsätts i Heuns metod:

$$y'(t) = 3y(t)\,(1 - y(t))$$

Heuns metod uppställd för denna ekvation blir:

$$\begin{aligned} K_1 &= 3y_k\,(1 - y_k) \\ K_2 &= 3(y_k + h * K_1)\,(1 - (y_k + h * K_1)) \\ y_{k+1} &= y_k + \frac{h}{2}\,(K_1 + K_2) \end{aligned}$$

(b) *Inverse Transform Sampling* innebär i det kontinuerliga fallet att x ska väljas som det värde som uppfyller $F(x) = u$. I vårt fall ska vi alltså lösa ekvationen:

$$\frac{x - a}{b - a} = u$$

och man får då formeln $x = a + u(b - a)$.

4. (a) Lokala trunkeringsfelet τ för implicita Eulers metod är:

$$\tau = y(t_{k+1}) - y(t_k) - hf(t_{k+1}, y(t_{k+1}))$$

På grund av differentialekvationen kan vi ersätta $f(t_{k+1}, y(t_{k+1}))$ med $y'(t_{k+1})$ vilket ger:

$$\tau = y(t_{k+1}) - y(t_k) - hy'(t_{k+1})$$

Analysera lokala trunkeringsfelet genom taylorutveckling kring lämplig punkt. Här väljer vi att taylorutveckla kring $t = t_k$. Notera att $t_{k+1} = t_k + h$.

$$\tau = y(t_{k+1}) - y(t_k) - hy'(t_{k+1}) = y(t_k) + hy'(t_k) + \frac{h^2}{2}y''(t_k) + \mathrm{O}(h^3)$$

$$-y(t_k) - h\left(y'(t_k) + hy''(t_k) + \mathrm{O}(h^2)\right) =$$

$$\left(\frac{h^2}{2} - h^2\right)y''(t_k) + \mathrm{O}(h^3) = \mathrm{O}(h^2)$$

Slutsatsen är att lokala trunkeringsfelet är $\mathrm{O}(h^2)$. Därmed är noggrannhetsordningen $2 - 1 = 1$.

(b) Det dominerande arbetet i Monte Carlo-metoden i detta fall är att för var och en av punkterna beräkna värdet på integranden (den funktion som ska integreras). Det innebär att exekveringstiden kommer att bli proportionell mot antalet punkter. Fördubbling av antalet punkter från 2000 till 4000 medför därför att exekveringstiden ungefär fördubblas.

5. (a) Om man tänker på de modeller som vi har använt i kursen, så är exekveringstiden för att simulera kemiska reaktioner kortare om man använder en deterministisk ODE-modell än om man gör stokastisk simulering. Om det slumpmässiga inslaget är försumbart är därför en deterministisk modell för de kemiska reaktionerna lämpligast. Under de förutsättningar som anges i uppgiften kan man anta att det vid varje tidpunkt sker många reaktioner mellan de olika kemikalierna (och inte enstaka reaktioner vid slumpmässiga tidpunkter). Därför kommer slumpmässigheten att vara försumbar i det scenario som uppgiften avser. Slutsatsen blir att det är lämpligast med en deterministisk modell i detta fall.

   (b) Båda metoderna är explicita och därför lämpliga i detta fall med ett problem som inte är styvt. Eftersom problemet inte är styvt kan vi anta att stabilitetsvillkoret inte kommer att vara begränsande, utan det kommer att vara toleransen som begränsar valet av steglängd. Heuns metod har noggrannhetsordning 2 och klassiska Runge-Kuttas metod har noggrannhetsordning 4. Det innebär att klassiska Runge-Kuttas metod kan uppfylla toleransen med betydligt större steglängd (och därmed färre steg) än Heuns metod. Detta uppväger mer än väl att klassiska Runge-Kuttas metod behöver utföra mera arbete per beräkningspunkt. Slutsatsen blir att klassiska Runge-Kuttas metod är lämpligare än Heuns metod i detta fall.

Uppgift som testar måluppfyllelse för betyg 4

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

$$\begin{pmatrix} y_1'(t) \\ y_2'(t) \end{pmatrix} = \begin{pmatrix} y_2(t) \\ \mu\left(1 - y_1(t)^2\right) y_2(t) - y_1(t) \end{pmatrix}$$

Vi ska nu implementera ett program i Matlab för att beräkna numerisk lösning till detta system av ekvationer. Eftersom problemet är styvt skulle en explicit metod av stabilitetsskäl behöva ta mycket korta steg. Det är då lämpligare med en implicit metod än en explicit metod. Eftersom `ode15s` bygger på en implicit metod är den ett lämpligt val här. Nedanstående är exempel på ett program som använder `ode15s` för att lösa Van der Pols ekvation.

Högerledet i ODE-systemet beskrivs som en funktion:

```
function yprim = VanDerPol( t, y )
    global mu
    yprim = [y(2);
             mu*(1-y(1)^2)*y(2) - y(1)];
end
```

Följande huvudprogram utför simuleringen:

```
% Läs in värde på parametern mu
global mu
mu = input('Värde på mu?  ');

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

% Läs in sluttiden T
T = input('Sluttid T? ');

% Genomför simuleringen
[t, y] = ode15s(@VanDerPol, [0 T], y0);

% Plotta x som funktion av tiden
plot(t,y(:,1));
xlabel('x');
ylabel('t');
```

Uppgift som testar måluppfyllelse för betyg 5

7. Effektivitetsvärdena E_i kan som framgår av uppgiften vara tal som är större än 1. För att översätta effektivitetsvärdena till sannolikhetsvärden gör vi följande omskalning:

$$P_i = E_i / \sum_{i=1}^{n} E_i,$$

där n är antalet bokare. Värdena P_i kommer att ligga mellan 0 och 1. Vidare gäller att $\sum_{i=1}^{n} P_i = 1$. Därför kan P_i uppfattas som ett sannolikhetsvärde.

För att slumpa fram numret på nästa bokare enligt sannolikheterna $P_i, i = 1, \ldots n$, använder vi algoritmen *Inverse Transform Sampling* i dess diskreta version. Det innebär att vi först genrerar ett slumptal u ur den likformiga fördelningen på intervallet (0, 1). Därefter väljer vi som vårt slumptal ur den önskade fördelningen: det minsta tal i för vilket $F(i) \geq u$, där F är fördelningsfunktionen för den sannolikhetsfördelning som ges av $P_i, i = 1, \ldots n$. I vårt Matlab-program kan F representeras av vektorn $[P_1, P_1 + P_2, \ldots, \sum_{i=1}^{n} P_i]$, som kan beräknas med kommandot `cumsum(P)`.

Med ovanstående som motivering kan en Matlab-funktion för stokastisk simulering baserad på modellen i uppgiften exempelvis formuleras som följer:

```
function antal = taxibokning(m, n, E)

antal = zeros(1,n);


% Beräkna sannolikheterna och fördelningsfunktionen
P = E/sum(E);
F = cumsum(P);

for k = 1:m
      u = rand;
      i = find(F >= u,1);
      antal(i) = antal(i) + 1;
end
```