Problemlösning 2

Anastasia Kruchinina

Uppsala Universitet

Februari 2016

Miniprojekt 1

Miniprojekt 2

Miniprojekt 1 - kommentarer

ode45 - explicit enstegsmetod (RK(4,5)), med adaptivt steglängdsval
ode15s - implicit flerstegsmetod (Numerical differentiation formulas), med adaptivt steglängdsval

`http://se.mathworks.com/help/simulink/ug/types-of-solvers.html`

Monte-Carlo methoder

Två viktiga teorem:

1. Law of large numbers - *konsistens*
   Om $n \to \infty$ medelvärdet konvergerar till det sanna förväntade värdet
2. Central limit theorem - *felet*
   Beräkningsfelet med denna metod i snitt avtar som $n^{-\frac{1}{2}}$

Monte-Carlo methoder

Exempel: beräkna flerdimensionella integraler, beräkna arean av områden i planet
`http://www.math.chalmers.se/Math/Grundutb/GU/MAN030/V07-2/mcarlo.pdf`

Varför Monte Carlo?

Felet:

	Trapez. rule	Simpson's rule	MC
1D	n^{-2}	n^{-4}	$n^{-\frac{1}{2}}$
2D	n^{-1}	n^{-2}	$n^{-\frac{1}{2}}$
...	...	...	...
kD	$n^{-\frac{2}{k}}$	$n^{-\frac{4}{k}}$	$n^{-\frac{1}{2}}$

Om felet i 1D är n^{-a}, sedan felet i kD är ofta $n^{-\frac{a}{k}}$.

Pseudoslumptal

Monte Carlo beräkningar använder pseudoslumptal, som genereras med hjälp av deterministiska algoritmer.

Generatorerna initialiseras med hjälp av ett startvärde (seed), som sätter det initiala tillståndet av generatorn.

I Matlab kolla `help rng`. Det ger dig möjlighet att upprepa experimentet med samma slumptal.
Från `help rng`:

```
% Example 1: Retrieve and Restore Generator Settings
s = rng           % get the current generator settings
x = rand(1,5)     % RAND generates some values
rng(s)            % restore the generator settings
y = rand(1,5)     % generate the same values as x
```

Miniprojekt 2

Mål: upptäcka slumpmässiga karaktären av reaktioner

Jämför deterministisk modell (ODE) som vi lösas med ode15s och ode45 med stokastisk modell (markovprocess).

Gillespie algorithm (Stochastic simulation algorithm)

Vår deterministiska modell är ett system av ODE, varje ekvation beskriver ett antal kemiska reaktioner. Tillståndet hos ett system definieras av *koncentrationen av molekyler*. Systemet av ODE beskriver många reaktioner händer samtidigt.

Problemen kommer när antalet molekyler är liten och reaktioner kan hända vid olika tidpunkter och i olika ordning.

Gillespie algoritm (Stochastic simulation algorithm)

Vi har N objekter (t.ex. N typer av molekyler). Tillståndet hos ett system definieras av $x(t) = [x_1(t), \ldots, x_N(t)]$ (var $x_1(t)$ är antalet molekyler av typ 1)
Vi har M reaktioner: r_j, $j = 1, \ldots, M$

Reaktion ändrar tillståndet-
kolla **stoichiometry matrix** (beror inte på det aktuella tillståndet)
Varje ekvation händer med någon sannolikhet -
kolla **propensity function** (beror på det aktuella tillståndet)

Med hjälp av slumptal och propensity funktion vi väljer nästa reaktion och tid.

Använd stoichiometry matrix att uppdatera tillståndet och hitta $x(t + 1)$.

Gillespie algoritm (Stochastic simulation algorithm)

Pseudokod för algoritmen:

```
Initial state x0
while( t < Tf )
  get the time until the next reaction
  get next reaction
  update the state
  update time
end
```

Gillespie algoritm (Stochastic simulation algorithm)

Propensity funktion $\omega_{r_j}(x(t))$ är *liknande* sannolikheten (du kan säga degree of expectation) att reaktion r_j händer i tidsintervalet $(t, t + dt]$ för givet tillstånd $x(t)$ vid tidpunkten t.

Vi definerar:

$$a_0(x(t)) = \sum_{j=1}^{M} \omega_{r_j}(x(t))$$

Gillespie algoritm (Stochastic simulation algorithm)

Teoretisk motivering ges av Gillespie.

Y är en slumpvariabel som ger en nästa reaktion.

Täthetsfunktion $P(Y = r_j|X = x(t))$ av Y för givet tillstånd $x(t)$ vid tidpunkten t är sannolikheten att nästa reaktion kommer att hända i tidsintervallet $(t, t + dt]$, och det ska bli reaktion r_j:

$$P(Y = r_j|X = x(t)) = \omega_{r_j}(x(t))/a_0(x(t))$$

Gillespie algoritm (Stochastic simulation algorithm)

Då den kumulativa fördelningsfunktionen är

$$\begin{aligned} F(r_j, x) &= P(Y \leq r_j | X = x(t)) \\ &= \sum_{i=1}^{j} P(Y = r_i | X = x(t)) = \sum_{i=1}^{j} \omega_{r_i}(x(t))/a_0(x(t)) \\ &= \frac{1}{a_0(x(t))} \left(\sum_{i=1}^{j} \omega_{r_i}(x(t)) \right) \end{aligned}$$

Notera:

$$\sum_{i=1}^{M} P(Y = r_i | X = x(t)) = 1$$

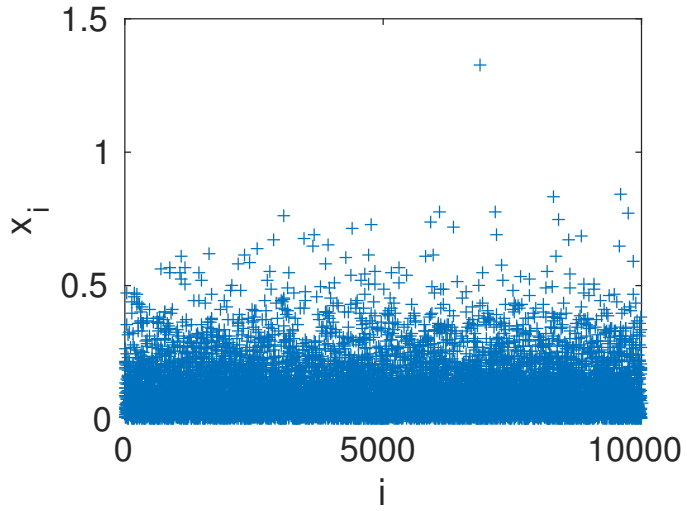
Steg: hitta tid till nästa reaktion

τ är en exponentiellt fördelad slumpvariabel med medelvärdet $\frac{1}{a_0(x(t))}$, var $a_0(x(t))$ som vi definierat tidigare.

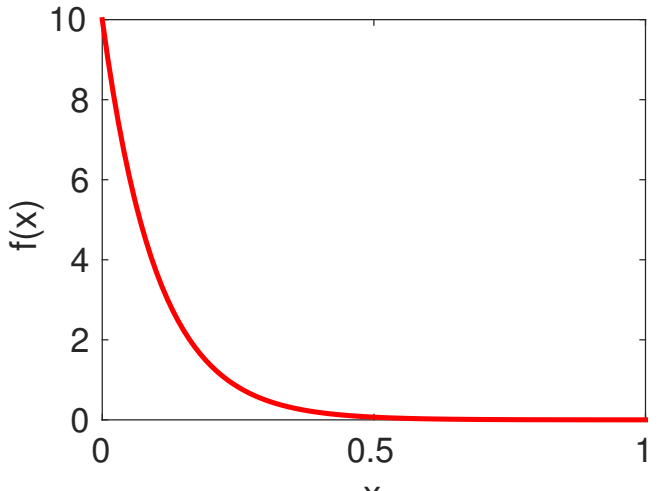
Använd inverse transform sampling algorithm! - Workout 3

Exponentialfördelning - exampel

Exponentialfördelat tal



Fördelningsfunktionen ($x \in [0, +\infty]$)



Steg: hitta vilken reaktion r som ska hända

Nästa reaktion är en diskret slumpvariabel Y. Dess fördelning beror på tillståndet x. Vi kan inte skriva invers av den kumulativa fördelningsfunktionen $F(r_j, x) = P(Y \leq r_j)$.

Hitta r_j som $F(r_{j-1}, x) < u \leq F(r_j, x)$, var u är likformig fördelad slumptal i $[0, 1]$.
Till exempel: vi hittar minimala värde j som $F(r_{j-1}, x) < u$.

```
help cumsum
```

```
help find
```

Kolla: s. 386 (section 11.5)
`http://physics.clarku.edu/courses/125/gtcdraft/chap11.pdf`

Simulera en genuttryck

Reaktions nätverk:

- transkription: $0 \xrightarrow{kR}$ mRNA
- translation : mRNA $\xrightarrow{kP*mRNA}$ mRNA + protein
- mRNA nedbrytning: mRNA $\xrightarrow{gR*mRNA} 0$
- protein nedbrytning: protein $\xrightarrow{gP*protein} 0$

Kolla mappen *stiff_problem1* med implementationen av problemet i Matlab.

Simulera en genuttryck

Law of large numbers - kolla mappen *problem_genuttryck_stochastic_mean* med implementationen av problemet i Matlab.

Rapport

- beskriva hur man får nästa reaktion och tid
- jämföra med figurer från miniprojekt 1, skriv vilken metod som används för varje figur
- diskutera vilken method (stochastik or deterministik) är bättre för vår problem. Kom ihåg att vi löser inte systemet av ODE!
- du måste spara inte resultat för varje tidssteg
- vi arbetar med diskreta värden, inte koncentrationer!

Lycka till!