Problem solving 2

Anastasia Kruchinina

Uppsala University

Ferbruary 2015

Miniproject 1

Miniproject 2

Miniproject 1 - comments

ode45 - explicit one-step (RK(4,5)), adaptive
ode15s - implicit multistep (Numerical differentiation formulas), adaptive

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

Monte-Carlo methods

Two important theorems:

1. Law of large numbers - *consistency*
   If we repeatedly sample a stochastic variable, the mean value converges to the true expected value
2. Central limit theorem - *error*
   Error decays as $n^{-\frac{1}{2}}$, where n is a sample size

Monte-Carlo methods

Examples: evaluation of integrals, computing area

Why Monte Carlo?

Errors:

	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	$n^{-\frac{2}{k}}$	$n^{-\frac{4}{k}}$	$n^{-\frac{1}{2}}$

In general if error is n^{-a} in 1D, then in kD error will be $n^{-\frac{a}{k}}$.

Pseudorandom numbers

Monte Carlo calculations use pseudorandom numbers, which are generated using deterministic algorithms.

The generators are initialized using a seed number, which sets the initial state of the generator.

In Matlab you can remember and set the seed with command `rng`: check `help rng`. It gives you the possibility to repeat the experiment with the same random numbers.
From `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
```

Miniproject 2

Goal: discover random nature of reactions

Compare deterministic model (ODE) which we solved using ode15s and ode45 with stochastic model (continuous time Markov chains).

Gillespie algorithm (Stochastic simulation algorithm)

Our deterministic model is a system od ODE, each equation describes a number of chemical reactions. The variables are the concentrations of the molecules and parameters are reaction rates. The system of ODE describe many reactions occuring simultaniously.

The problems comes when the number of molecules is small and reactions can occur at different time and in random order.

Gillespie algorithm (Stochastic simulation algorithm)

We have N objects (ex. N kinds of molecules) and the state vector $x(t) = [x_1(t), \ldots, x_N(t)]$ (where $x_1(t)$ is the number of molecules of the first kind)
We have M reactions: r_j, $j = 1, \ldots, M$

Reaction changes the state -
see **stoichiometry matrix** (does not depends on the current state)
Every reaction happens with some probability -
see **propensity function** (depends on the current state)

Using random numbers and the propensity function we choose reaction and time for it.

Use stoichiometry matrix to update the state and get $x(t + 1)$.

Gillespie algorithm (Stochastic simulation algorithm)

Pseudocode for the algorithm:

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

Gillespie algorithm (Stochastic simulation algorithm)

Propensity function $\omega_{r_j}(x(t))$ is *like* the probability (you can say degree of expectation) that reaction r_j occur in time interval $(t, t + dt]$ given the state $x(t)$ at time t

We define

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

Gillespie algorithm (Stochastic simulation algorithm)

Theoretical justification is given by Gillespie.

Let Y is a random variable giving a next reaction.

Probability distribution of Y given state $x(t)$ is $P(Y = r_j|X = x(t))$ probability that, given the state $x(t)$ at time t, the next reaction will occur in the time interval $(t, t + dt]$, and will be a reaction r_j:

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

Gillespie algorithm (Stochastic simulation algorithm)

Then the cumulative distribution function is

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

Notice:

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

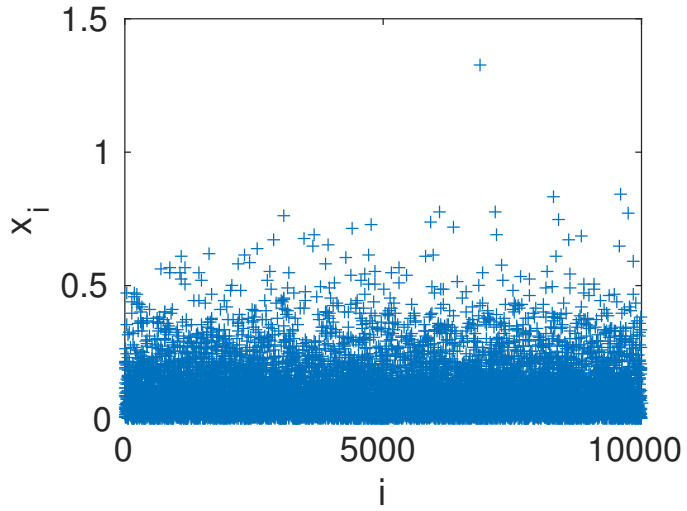
Step: get the time until the next reaction

Let τ is a exponentially distributed random variable with mean $\frac{1}{a_0(x(t))}$, where $a_0(x(t))$ is given before.

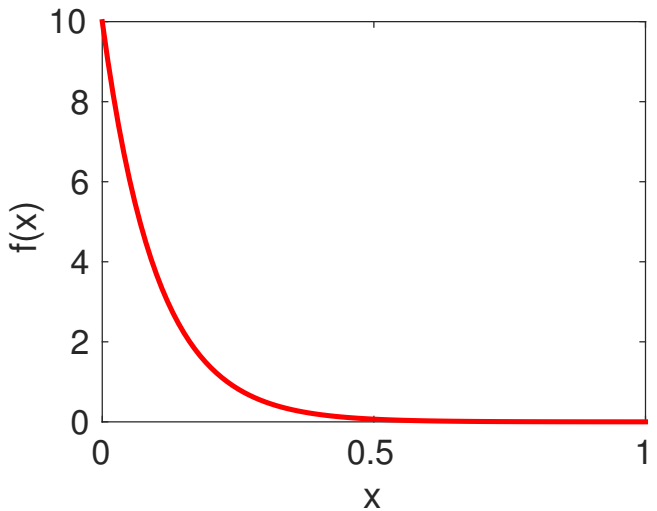
Use inverse transform sampling algorithm! - Workout 3

Exponential distribution - example for some particular a_0

Exponentially distributed numbers



Probability density function for exponential distribution ($x \in [0, +\infty]$)



Step: get next reaction

The next reaction is a discrete random variable Y. Its distribution depends on the state x. You cannot explicitly write the inverse of the cumulative distribution function $F(r_j, x) = P(Y \leq r_j)$.

Find r_j such that $F(r_{j-1}, x) < u \leq F(r_j, x)$, where u is a uniform random number in $[0, 1]$.
Is equivalent for example that we find minimal j such that $F(r_{j-1}, x) < u$.

```
help cumsum
```

```
help find
```

Simulate gene expression

Reaction network:

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

Check the *problem_genuttryck_stochastic* folder with implementation of the problem in Matlab.

Simulate gene expression

law of large numbers - check the *problem_genuttryck_stochastic_mean* folder with implementation of the problem in Matlab.

Reports

- describe how to get next reaction and its time
- comparison of figures with Miniproject 1, be clear which method is used for each figures
- discussion which method (stochastic or deterministic is better for given problem). Remeber that here we are not solving the system of ODEs!
- do not save results for every time step
- we work with discrete values, not concentrations anymore!