Illustrating SSA with stochastic single-cell transcription

Michael Anselmi

December 6, 2012
Outline

Introduction to SSA
  Multiple SSAs
  Direct method

StochPy
  Three-stage model of gene expression
  Single-cell transcription
  Model specification
  Run simulations
  Results
Stochastic Simulation Algorithm(s)

There is not just one stochastic simulation algorithm, e.g.:

- Direct method (1977)
- First reaction method (1977)
- Next reaction method (2000)
Stochastic Simulation Algorithm(s)

There is not just one stochastic simulation algorithm, e.g.:

- Direct method (1977)
- First reaction method (1977)
- Next reaction method (2000)

Monte Carlo methods
Exact vs. approximate
Each of these is implemented in StochPy. (coming soon!)
Direct method

Here we use the direct method.

Profile

► Published by Dan Gillespie in 1977. [1]
► Used to simulate (bio)chemical systems of reactions.
► Useful for simulating systems with few particles.
► Too slow for large systems.
Direct method

Algorithm

1. Initialization: Initialize number of particles, reaction rates, RNGs.
Direct method

Algorithm

1. Initialization: Initialize number of particles, reaction rates, RNGs.
2. Monte Carlo step: Generate random numbers to determine (i) next reaction to occur and (ii) time step. The probability of choosing a given reaction is proportional to the number of substrate particles.

Update: Advance time by amount determined in step 2. Update particle count.

4. Iterate: Return to step 2 unless (i) no reactants remain or (ii) simulation time exceeded.
Direct method

Algorithm

1. Initialization: Initialize number of particles, reaction rates, RNGs.

2. Monte Carlo step: Generate random numbers to determine (i) next reaction to occur and (ii) time step. The probability of choosing a given reaction is proportional to the number of substrate particles.

Direct method

Algorithm

1. Initialization: Initialize number of particles, reaction rates, RNGs.
2. Monte Carlo step: Generate random numbers to determine (i) next reaction to occur and (ii) time step. The probability of choosing a given reaction is proportional to the number of substrate particles.
4. Iterate: Return to step 2 unless (i) no reactants remain or (ii) simulation time exceeded.
StochPy is a versatile modelling package for stochastic simulation of molecular control networks inside living cells. Its integration with Python’s scientific libraries and PySCeS makes it an easily extendible and an user-friendly simulator. The high-level statistical and plotting functions of StochPy allow for quick and interactive model interrogation at the command-line. Python’s scripting capabilities allow for more complicated and in-depth analysis of stochastic models.
Three-stage model of gene expression [4]

1. Activation/deactivation
2. Transcription (DNA copied to mRNA)
3. Translation (mRNA to protein)
Three-stage model of gene expression

Let $P_{m,n}^{(0)}$ ($P_{m,n}^{(1)}$) be the probability of having $m$ mRNAs and $n$ proteins when DNA is inactive (active). Then

$$
\frac{\partial P_{n,m}^{(0)}}{\partial \tau} = \kappa_1 P_{m,n}^{(1)} - \kappa_0 P_{m,n}^{(0)} + (n + 1) P_{m,n+1}^{(0)} - nP_{m,n}^{(0)} \\
\quad \quad \quad \quad \quad + \gamma \left[ (m + 1) P_{m+1,n}^{(0)} - m P_{m,n}^{(0)} + b m \left( P_{m,n-1}^{(0)} - P_{m,n}^{(0)} \right) \right]
$$

$$
\frac{\partial P_{n,m}^{(1)}}{\partial \tau} = -\kappa_1 P_{m,n}^{(1)} + \kappa_0 P_{m,n}^{(0)} + (n + 1) P_{m,n+1}^{(1)} - nP_{m,n}^{(1)} \\
\quad \quad \quad \quad \quad + a \left( P_{m-1,n}^{(1)} - P_{m,n}^{(1)} \right) \\
\quad \quad \quad \quad \quad + \gamma \left[ (m + 1) P_{m+1,n}^{(1)} - m P_{m,n}^{(1)} + b m \left( P_{m,n-1}^{(1)} - P_{m,n}^{(1)} \right) \right],
$$

where $a = \nu_0/d_1$, $b = \nu_1/d_0$, $\gamma = d_0/d_1$, $\tau = d_1 t$, $\kappa_0 = k_0/d_1$, and $\kappa_1 = k_1/d_1$. 
Three-stage model of gene expression

Let \( P_{m,n}^{(0)} (P_{m,n}^{(1)}) \) be the probability of having \( m \) mRNAs and \( n \) proteins when DNA is inactive (active). Then

\[
\frac{\partial P_{n,m}^{(0)}}{\partial \tau} = \kappa_1 P_{m,n}^{(1)} - \kappa_0 P_{m,n}^{(0)} + (n + 1) P_{m,n+1}^{(0)} - n P_{m,n}^{(0)} \\
+ \gamma \left[ (m + 1) P_{m+1,n}^{(0)} - m P_{m,n}^{(0)} + b m \left( P_{m,n-1}^{(0)} - P_{m,n}^{(0)} \right) \right]
\]

\[
\frac{\partial P_{n,m}^{(1)}}{\partial \tau} = -\kappa_1 P_{m,n}^{(1)} + \kappa_0 P_{m,n}^{(0)} + (n + 1) P_{m,n+1}^{(1)} - n P_{m,n}^{(1)} \\
+ a \left( P_{m-1,n}^{(1)} - P_{m,n}^{(1)} \right) \\
+ \gamma \left[ (m + 1) P_{m+1,n}^{(1)} - m P_{m,n}^{(1)} + b m \left( P_{m,n-1}^{(1)} - P_{m,n}^{(1)} \right) \right],
\]

where \( a = v_0/d_1, b = v_1/d_0, \gamma = d_0/d_1, \tau = d_1 t, \kappa_0 = k_0/d_1, \) and \( \kappa_1 = k_1/d_1. \) There is a closed-form solution!
Single-cell transcription

We remove translation and work with a transcription-only model.

1. Activation/deactivation
2. Transcription (DNA copied to mRNA)
Model specification [2]

Provide model description in Python Simulator for Cellular Systems (PyS CeS) format.

```
# Burstmodel.psc

# Reactions
R1:
   ONstate > OFFstate
   koff*ONstate

R2:
   OFFstate > ONstate
   kon*OFFstate

R3:
   ONstate > mRNA + ONstate
   ksyn*ONstate

R4:
   mRNA > $ pool
   kdeg*mRNA

# Initialization
kon    = 0.05
koff   = 0.05
kdeg   = 2.5
ksyn   = 80
ONstate = 0
OFFstate = 1    # cell volume size of 1
mRNA   = 0
```
Run simulation until time 100 reached.

```python
import matplotlib.gridspec as gridspec
import os, stochpy

sim_end = 100
mod = stochpy.SSA()
mod.Model(File='Burstmodel.psc', dir=os.getcwd())
mod.ChangeParameter('kon', 0.05)
mod.ChangeParameter('koff', 0.05)
mod.DoStochSim(end=sim_end, mode='time', method='Direct', trajectories=1)
```

Change reaction rates and run simulation again.

```python
mod.ChangeParameter('kon', 5.0)
mod.ChangeParameter('koff', 5.0)
mod.DoStochSim(end=sim_end, mode='time', method='Direct', trajectories=1)
```
mRNA time plot
Summary statistics

Print species means and standard deviations.

mod.ShowMeans()
mod.ShowStandardDeviations()

Species Mean
ONstate 0.363673952002
OFFstate 0.636326047998
mRNA 11.6278151286
Species Standard Deviation
ONstate 0.481056346634
OFFstate 0.481056346634
mRNA 15.2336344651

Species Mean
ONstate 0.924810389099
OFFstate 0.075189610901
mRNA 29.2790840005
Species Standard Deviation
ONstate 0.26369704836
OFFstate 0.26369704836
mRNA 9.89652706128
Now, run both simulations for 1,000,000 time steps each.

```python
tonimesteps = 1000000
mod = stochpy.SSA(File='Burstmodel.psc', dir=os.getcwd())
mod.ChangeParameter('kon', 0.05)
mod.ChangeParameter('koff', 0.05)
mod.DoStochSim(end=ntimesteps, mode='steps')

mod.ChangeParameter('kon', 5.0)
mod.ChangeParameter('koff', 5.0)
mod.DoStochSim(end=ntimesteps, mode='steps')
```
mRNA equilibrium distributions
mRNA waiting time distributions

![Graph showing mRNA waiting time distributions with probability density on the y-axis and time between mRNA synthesis events on the x-axis. The graph includes a bimodal distribution and an analytic solution.](image-url)
References I

