Multi-level methods for discrete state systems

Christian A. Yates

Centre for Mathematical Biology
Department of Mathematical Sciences
University of Bath

June 10, 2015

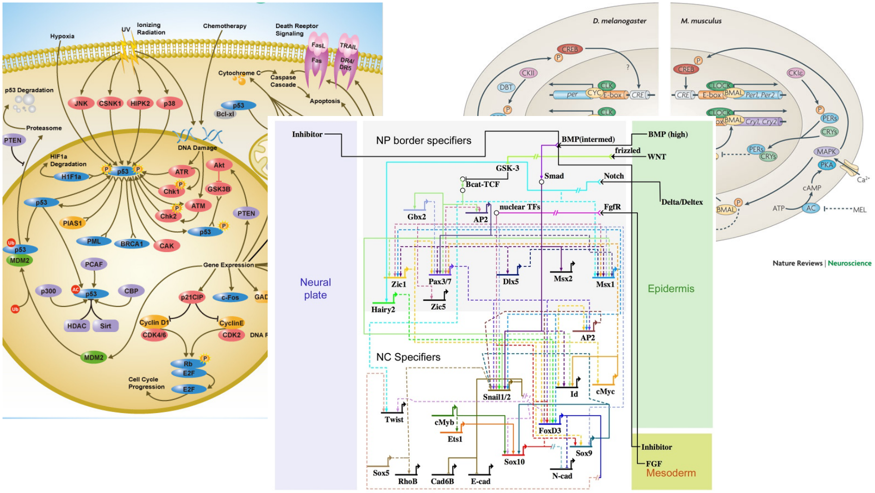
What is Multi-level?

- A computationally efficient method for producing a point estimator of a variable of interest in a stochastic system.
- Combines estimators of the differing accuracy in a ‘telescoping sum’.
- First calculate an estimator using an approximate stochastic method, which is efficient, but has significant bias.
- Then reduce this bias successively on subsequent ‘levels’ by combining estimators from approximate stochastic simulation algorithms with increasing accuracy in an efficient manner.
- Optionally correct with an exact stochastic method to completely remove the bias.

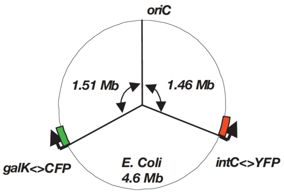
Outline

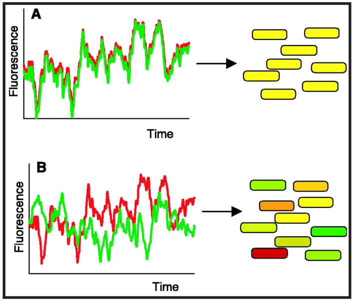
- Biological motivation.
- Introduction to modelling of stochastic biochemical systems.
- Exact simulation methods.
- Approximate simulation methods.
- Introduction to fixed time step multi-level.
- Adaptive multi-level.

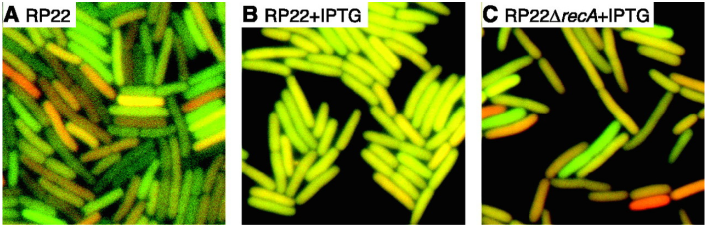
Gene regulatory networks



The role of noise







Stochastic gene expression in a single cell. M.B. Elowitz, and A.J. Levine, and E.D. Siggia, and P.S. Swain, Science (2002).

Modelling biochemical systems

Model using a discrete framework where we have:

- N species, $S_1,\dots,S_N$, and M interactions, $R_1,\dots,R_M$ e.g.
$$S_1 + S_2 \to S_3,$$
- *state vector:*
$$\mathbf{X}(t) := [X_1(t),\dots,X_N(t)]^T, \tag{1}$$
- *stoichiometric or state-change vector:*
$$\boldsymbol{\nu}_j := [\nu_{1j},\dots,\nu_{Nj}]^T, \tag{2}$$
where ν_{ij} is the change in the copy number of S_i caused by reaction R_j taking place.
- *propensity function:*
$$a_j(\mathbf{X}(t)), \tag{3}$$
probability that reaction, j, occurs in the infinitesimally small time interval $[t, t+\mathrm{d}t)$ is $a_j(t)\mathrm{d}t$.

Modelling biochemical systems

The "chemical master equation" or "Kolmogorov's forward equation"

$$\frac{\mathrm{d}\mathbb{P}(\mathbf{x}, t \mid \mathbf{x_0}, t_0)}{\mathrm{d}t} = \sum_{j=1}^{M} \Big\{ \mathbb{P}(\mathbf{x} - \boldsymbol{\nu}_\mathbf{j}, t \mid \mathbf{x_0}, t_0) \cdot a_j(\mathbf{x} - \boldsymbol{\nu}_\mathbf{j}) - \mathbb{P}(\mathbf{x}, t \mid \mathbf{x_0}, t_0) \cdot a_j(\mathbf{x}) \Big\}. \qquad (4)$$

Bad news

No matter how we write the model mathematically, unless the system is *very* simple[1], we can't solve analytically.

The large (possibly infinite) state space of most models makes numerical approximation of $\mathbb{P}(\mathbf{x}, t \mid \mathbf{x_0}, t_0)$ a solution infeasible.

[1]By simple, I mean only zeroth or first order reactions only, and this makes for systems that are pretty boring and whose mean behaviour can be described by the corresponding phenomenological ODE model. Even then the system of probability master equations could be high or infinite dimensional.

Good News

There is an equivalent model to the Chemical master equation:

Pathwise representation: the random-time-change representation

$$\mathbf{X}(t) = \mathbf{X}(0) + \sum_{j=1}^{M} Y_j \left(\int_0^t a_j(\mathbf{X}(s)) \mathrm{d}s \right) \cdot \boldsymbol{\nu}_j, \tag{5}$$

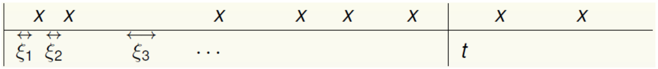
where the Y_j are unit rate Poisson processes.

The Poisson process

A Poisson process, $Y(\cdot)$, is a model for a series of random observations occurring in time.

(a) Let ξ_i be i.i.d. exponential random variables with parameter one.

(b) Now, put points down on a line with spacing equal to the ξ_i.



Let $Y(t)$ denote the number of points hit by time t.

In the figure above, $Y(t) = 6$.

Intuition:

The unit rate Poisson process is simply the number of points hit when we run along the time frame at rate one.

The Poisson process

- Let Y be a unit rate Possion process.
- Define $Y_a(t) \triangleq Y(at)$.

Then Y_a is a Poisson process with parameter a.

Intuition:

The Poisson process with rate a is simply the number of points (of the unit rate Possion process) hit when we run along the time frame at **rate** a.

We have "changed time" to convert a unit rate Poisson process to one which has rate a.
There is no reason a needs to be constant in time, in which case

$$Y_a(t) = Y\left(\int_0^t a(s)\mathrm{d}s\right) \tag{6}$$

is an inhomogeneous Poisson process.

Numerical simulation of paths

There are a number of numerical methods that produce statistically exact sample paths:

1. Gillespie's algorithm (the direct method).
2. The first reaction method.
3. The next reaction method.
4. The optimized direct method.
5. The sorting direct method.
6. The logarithmic direct method.
7. The recycling direct method.

The (exact) Gillespie algorithm

1. Initialize the time $t = t_0$ and the species numbers $\mathbf{X}(t_0) = \mathbf{x}_0$.
2. Evaluate the propensity functions, $a_j(\mathbf{x})$, associated with the reaction channels R_j ($j = 1, \dots, M$) and their sum $a_0(\mathbf{x}) = \sum_{j=1}^{M} a_j(\mathbf{x})$.
3. Generate two random numbers $rand_1$ and $rand_2$ uniformly distributed in $[0, 1]$.
4. Use $rand_1$ to generate a time increment, τ, from the exponentially distributed random variable with mean $1/a_0(\mathbf{x})$. i.e.
$$\tau = \frac{1}{a_0} \ln\left(\frac{1}{rand_1}\right). \tag{7}$$
5. Use $rand_2$ to generate a reaction index j with probability $a_j(\mathbf{x})/a_0(\mathbf{x})$, in proportion with its propensity function. i.e. find j such that
$$\sum_{j'=1}^{j-1} a_{j'}(\mathbf{x}) < a_0(\mathbf{x}) \cdot rand_2 < \sum_{j'=1}^{j} a_{j'}(\mathbf{x}). \tag{8}$$
6. Update the state vector, $\mathbf{x} = \mathbf{x} + \boldsymbol{\nu}_j$ and the time, $t = t + \tau$.
7. If $t < T$, the desired stopping time, then go to step (2). Otherwise stop.

Bad news

Since the Chemical Master equation is often intractable, the estimation of system statistics from collections of many system sample paths is the main means by which to investigate model behaviour.

BUT

For each step of the simulation methods one must find:

1. the amount of time that passes until the next reaction takes place:

$$\tau = \frac{1}{a_0} \ln \left(\frac{1}{rand_1} \right). \tag{7}$$

2. which reaction takes place at that time.

If $a_0 = \sum_{j=1}^{M} a_j(\mathbf{x}) \gg 1$, then $\langle \tau \rangle = \frac{1}{a_0} \ll 1$.

If $M \gg 1$ then searching to find which reaction occurs can be slow.

Time to produce a single path over an interval $[0, T]$ can, therefore, be prohibitive.

A simple example

Consider a model of gene transcription and translation[2]:

$$
\begin{aligned}
R_1: &\quad G \xrightarrow{25} G + M, && (9)\\
R_2: &\quad M \xrightarrow{1000} M + P, && (10)\\
R_3: &\quad P + P \xrightarrow{0.001} D, && (11)\\
R_4: &\quad M \xrightarrow{0.1} \emptyset, && (12)\\
R_5: &\quad P \xrightarrow{1} \emptyset. && (13)
\end{aligned}
$$

Write the numbers of mRNA, protein and dimer molecules at time t, respectively, as

$$\mathbf{X}(t) = (X_1(t), X_2(t), X_3(t))^T.$$

with

$$\mathbf{X}(0) = (0, 0, 0)^T.$$

[2] Multi-level Monte Carlo for continuous time Markov chains, with applications in biochemical kinetics. D. Anderson and D. Higham, SIAM Multiscale Model. Simul. (2012).

A simple example

$$\boldsymbol{\nu} = \left[\begin{array}{ccccc} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & -2 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \end{array} \right]. \tag{14}$$

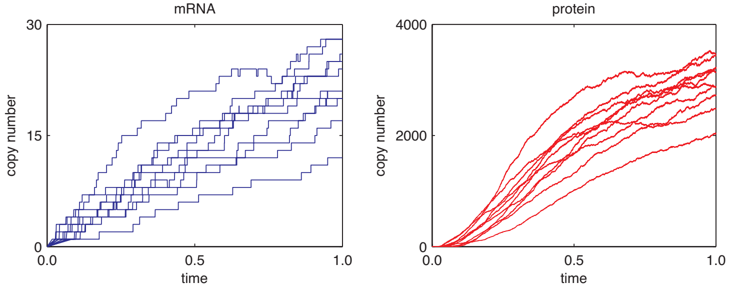
Aim: to estimate the numbers of dimers at time $t = 1$, $X_3(1)$.

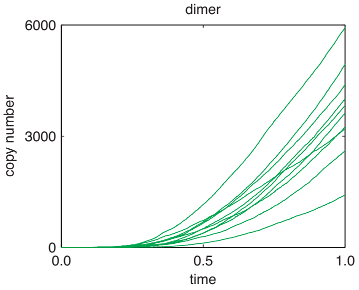
If we use the Gillespie algorithm to generate n sample paths then we estimate:

$$\mathbb{E}[X_i(T)] \approx \frac{1}{n} \sum_{r=1}^{n} X_i^{(r)}(T), \tag{15}$$

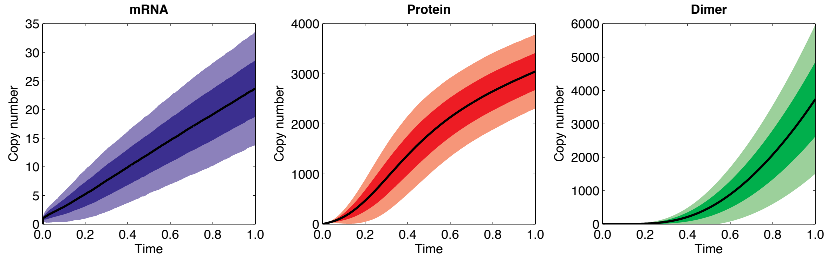
where $X_i^{(r)}(t)$ represents the copy number of species i at time t in path r.

The (exact) Gillespie algorithm





The (exact) Gillespie algorithm



To compute $\mathbb{E}[X_3(1)]$ to within a single dimer with 95% confidence requires the generation of approximately 4,800,000 sample paths[3].

Using an optimized Gillespie algorithm, this calculation took approximately six hours (21,472 seconds) when run on our AMD desktop computer.

Good News

Faster approximate methods exist: τ-leaping.

[3]For later comparison: $\mathbb{E}[X_3(1)] = 3714.0 \pm 0.99$.

The (approximate) τ-leaping algorithm

Recall the time inhomogeneous Poisson process model:

$$\mathbf{X}(t) = \mathbf{X}(0) + \sum_{j=1}^{M} Y_j \left(\int_0^t a_j(\mathbf{X}(s))\mathrm{d}s \right) \cdot \boldsymbol{\nu}_j.$$

The τ-leaping algorithm approximates the integral in steps of (fixed) length, τ.

Let $N_\tau = t/\tau$ to give

$$\mathbf{X}(t) = \mathbf{X}(0) + \sum_{j=1}^{M} \sum_{k=1}^{N_\tau} Y_j \left(\tau a_j(\mathbf{X}(\{(k-1)\tau\}^-)) \right) \cdot \boldsymbol{\nu}_j. \qquad (16)$$

All reaction rates are assumed to remain constant over each time step. [Essentially an Euler approximation of the system dynamics.]

The (approximate) τ-leaping algorithm

1. Initialize the time $t = t_0$, the species numbers $\mathbf{X}(t_0) = \mathbf{x}_0$ and choose the value of τ.
2. Evaluate the propensity functions, $a_j(\mathbf{X}(t))$.
3. For $j = 1, \ldots, M$, generate k_j as a sample of the Poisson random variable with mean $a_j(\mathbf{X}(t))\tau$. k_j represents the number of times reaction R_j fires in $[t, t + \tau)$.
4. Update the state vector according to the number of each reactions that have fired: $\mathbf{X(t + \boldsymbol{\tau})} = \mathbf{X(t)} + \sum_{j=1}^{M} k_j \boldsymbol{\nu}_j$ and the time, $t = t + \tau$.
5. If $t < T$, the desired stopping time, then go to step (2). Otherwise stop.

The (approximate) τ-leaping algorithm

Using the τ-leaping algorithm, we have[4]:

$\boldsymbol{\tau}$	$\mathbb{E}[X_3]$	**Paths**	**Time**
$\mathbf{3^{-2}}$	3186.67 ± 1.03	3.71×10^6	175.7s
$\mathbf{3^{-3}}$	3537.98 ± 1.01	4.37×10^6	585.5s
$\mathbf{3^{-4}}$	3656.02 ± 1.05	4.24×10^6	1,435.6s
$\mathbf{3^{-5}}$	3694.02 ± 1.10	4.56×10^6	3,663.3s
$\mathbf{3^{-6}}$	3707.58 ± 1.02	4.56×10^6	6,851.3s

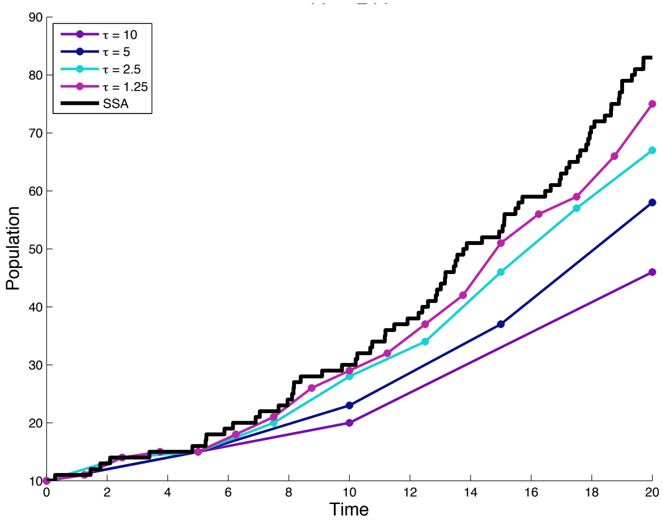
Bad News

Estimators appear to be fast and have a large bias or have a smaller bias, but be considerably slower.

[4]Recall: $\mathbb{E}[X_3(1)] = 3714.0 \pm 0.99$ was the estimate using the exact Gillespie algorithm using 4.8 million paths, and it took 21,472 seconds to compute.

The (approximate) τ-leaping algorithm

$$A \xrightarrow{1/10} 2A.$$



Summary of approaches

Bad News

- Exact stochastic simulation algorithms are computationally expensive (prohibitively so if one wants to do *e.g.* sensitivity analysis or consider complex networks).
- Approximate stochastic simulation algorithms can generate sample paths more quickly, but the bias is often *very* significant.
- If one takes τ small enough to get an accurate estimator then the computational cost is similar to that of an exact algorithm.

Good News

Enter the multi-level Monte Carlo method (which combines all the pros and removes many of the cons...)

Multi-level Monte Carlo

Generate an initial guess for the estimator.

- Use an approximate stochastic simulation algorithm for the initial guess.
- Quick to calculate (so calculate lots of paths) but inaccurate due to significant bias.

Gradually make the initial guess more accurate by adding correction terms.

- Correction terms decrease the bias by combining estimators from approximate stochastic simulations algorithms of increasing accuracy, until a desired level of accuracy is reached.

Suppose we wish to estimate the expected value of $X_i(T)$, the population of the i-th species at time T...

Multi-level Monte Carlo – level 0

Use a τ-leaping SSA with a large value of τ (τ_0) to generate a large number (n_0) of sample paths of the system.

$$Q_0 := \mathbb{E}\left[Z_{\tau_0}\right] \approx \frac{1}{n_0}\sum_{r=1}^{n_0} Z_{\tau_0}^{(r)}(T). \tag{17}$$

- $Z_{\tau}^{(r)}(t)$ – copy number of species i at time t in path r generated using the tau-leaping method with time step τ.
- n_ℓ – number of paths generated on level ℓ.

KEY POINT: estimate is calculated cheaply, but it has considerable bias.

Multi-level Monte Carlo – level 1

GOAL: introduce a correction term that reduces this bias.

$$Q_1 := \mathbb{E}\left[Z_{\tau_1} - Z_{\tau_0}\right] \approx \frac{1}{n_1} \sum_{r=1}^{n_1} \left[Z_{\tau_1}^{(r)}(T) - Z_{\tau_0}^{(r)}(T)\right]. \qquad (18)$$

Two sets of n_1 sample paths:

- one set comes from the τ-leaping SSA with $\tau = \tau_0$;
- the other set also comes from τ-leaping SSA, but with $\tau = \tau_1 < \tau_0$.

Add the correction term to the estimator calculated on level 0:

$$Q_0 + Q_1 = \mathbb{E}\left[Z_{\tau_0}\right] + \mathbb{E}\left[Z_{\tau_1} - Z_{\tau_0}\right] = \mathbb{E}\left[Z_{\tau_1}\right]. \qquad (19)$$

KEY POINT: the sum of the two estimators has a bias equivalent to that of the tau-leaping method with $\tau = \tau_1$.

Multi-level Monte Carlo – Level 2

Repeat to get a second correction term:

$$Q_2 := \mathbb{E}\left[Z_{\tau_2} - Z_{\tau_1}\right] \approx \frac{1}{n_2}\sum_{r=1}^{n_2}\left[Z_{\tau_2}^{(r)}(T) - Z_{\tau_1}^{(r)}(T)\right]. \qquad (20)$$

Two sets of n_2 sample paths:

- one set has $\tau = \tau_1$;
- the second has $\tau = \tau_2 < \tau_1$.

The estimator is now

$$Q_0 + Q_1 + Q_2 = \mathbb{E}\left[Z_{\tau_2}\right]. \qquad (21)$$

KEY POINT: the sum of the two estimators has a bias equivalent to that of the tau-leaping method with $\tau = \tau_2$.

Multi-level Monte Carlo

Repeat this lots of times[5]

$$\vdots$$

to get a telescoping sum of the form

$$\mathbb{E}\left[Z_{\tau_L}\right] = \mathbb{E}\left[Z_{\tau_0}\right] + \sum_{\ell=1}^{L} \mathbb{E}\left[Z_{\tau_\ell} - Z_{\tau_{\ell-1}}\right] = \sum_{\ell=0}^{L} Q_\ell. \qquad (22)$$

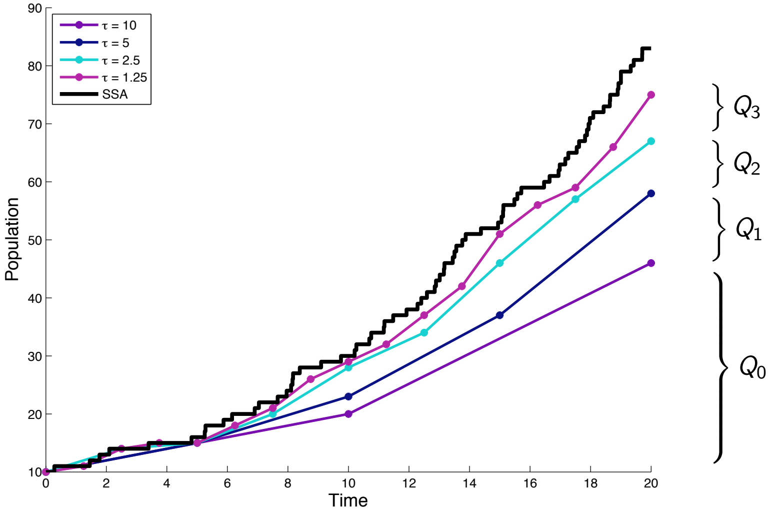
KEY POINT: with the addition of each subsequent level the bias of the estimator is reduced further, until a desired level of accuracy[6] is reached.

[5] The number of levels is TBC.

[6] More about this on the next slide.

Multi-level Monte Carlo

$$A \xrightarrow{1/10} 2A.$$



Multi-level Monte Carlo – accuracy

At the end of it all we have a biased estimator:

$$\mathbb{Q}_b := \sum_{\ell=0}^{L} Q_\ell = \mathbb{E}\left[Z_{\tau_0}\right] + \sum_{\ell=1}^{L} \mathbb{E}\left[Z_{\tau_\ell} - Z_{\tau_{\ell-1}}\right]. \tag{23}$$

Errors

$$\mathrm{MSE}(\mathbb{Q}_b) = \mathbb{E}[(\mathbb{Q}_b - \mu)^2] = \mathrm{var}(\mathbb{Q}_b) + (\mathbb{E}[\mathbb{Q}_b - \mu])^2. \tag{24}$$

Statistical error: $\mathrm{var}(\mathbb{Q}_b)$

- Controlled by bounding the associated estimator variance, $\mathbb{V}_b < \varepsilon^2$, and effectively requires generating "enough" sample paths on each level.

Bias: $\mathbb{E}[\mathbb{Q}_b - \mu]$

- Controlled by taking "sufficiently many" correction levels.

Multi-level Monte Carlo – unbiased estimator

Optionally we can remove the bias using a final level:

$$Q_{L+1}^{*} = \mathbb{E}\left[X_i - Z_{\tau_L}\right] \approx \frac{1}{n_{L+1}} \sum_{r=1}^{n_{L+1}} \left[X_i^{(r)}(T) - Z_{\tau_L}^{(r)}(T)\right]. \qquad (25)$$

Two sets of n_{L+1} sample paths:

- one set generated using τ-leaping with $\tau = \tau_L$;
- and the other set using an exact SSA.

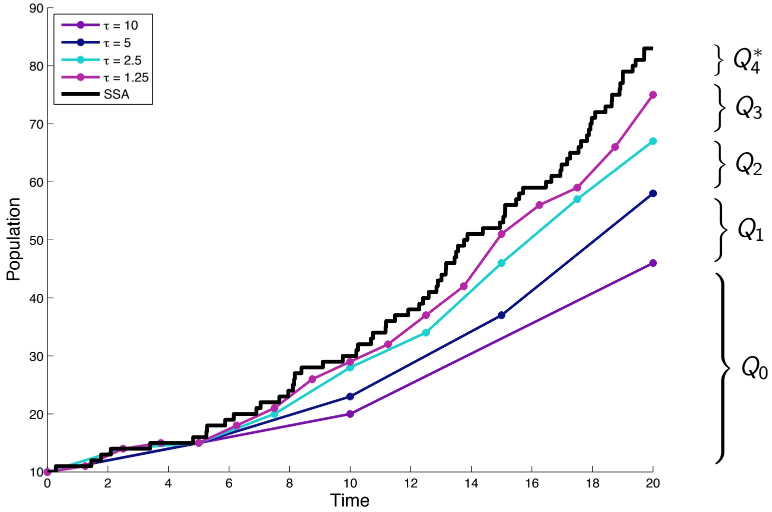
The estimator is now

$$\mathbb{E}\left[X_i\right] = \mathbb{E}\left[Z_{\tau_0}\right] + \sum_{\ell=1}^{L} \mathbb{E}\left[Z_{\tau_\ell} - Z_{\tau_{\ell-1}}\right] + \mathbb{E}\left[X_i - Z_{\tau_L}\right] = \sum_{\ell=0}^{L} Q_\ell + Q_{L+1}^{*}. \qquad (26)$$

KEY POINT: The total time taken to generate the sets of sample paths for the base level, Q_0, and each of the correction terms, Q_ℓ for $\ell = 1, \ldots, L$, and Q_{L+1}^{*}, can be less than that taken to estimate $\mathbb{E}[X_i(T)]$ using an exact SSA.

Multi-level Monte Carlo

$$A \xrightarrow{1/10} 2A.$$



In order to use multi-level...

... a number of decisions have to be made.
We must consider:

1. the choice of time steps $\tau_0, \tau_1, \ldots, \tau_L$ both within and between levels;
2. the values the target variance, $\widehat{V_\ell}$, should take on each level, ℓ. This ensures statistical accuracy, and also affects the simulation time;
3. the choices of simulation techniques for the base level (0), the correcting levels, $1, \ldots, L$, and (if desired) the final level $L+1$.

1. Time step choice

We let $K \in \{2, 3, \dots\}$ be a scaling factor and take $\tau_\ell = \tau_{(\ell-1)}/K$ so that

$$\begin{aligned}
Q_0 &\equiv \mathbb{E}[Z_{\tau_0}], \\
Q_1 &\equiv \mathbb{E}[Z_{\tau_0/K} - Z_{\tau_0}], \\
Q_2 &\equiv \mathbb{E}[Z_{\tau_0/K^2} - Z_{\tau_0/K}], \\
&\vdots \\
Q_L &\equiv \mathbb{E}[Z_{\tau_0/K^L} - Z_{\tau_0/K^{L-1}}].
\end{aligned}$$

This means that the intervals are nested, with the same scaling factor between each, and it renders the algorithm more simple to understand and implement.

2. How to choose n_ℓ and $\widehat{V_\ell}$

- c_ℓ time taken to generate n_ℓ sample paths on level ℓ.
- $c_\ell \propto n_\ell$ and $V_\ell \propto 1/n_\ell$ so

$$c_\ell = \frac{k_\ell}{V_\ell}. \tag{27}$$

- We want to minimize the total expected computational time:

$$\sum_{\ell=0}^{L} \widehat{c_\ell} = \sum_{\ell=0}^{L} \frac{k_\ell}{\widehat{V_\ell}}, \tag{28}$$

  subject to the constraint $\sum_{\ell=0}^{L} \widehat{V_\ell} < \varepsilon^2$.
- Estimate k_ℓ by an small initial number of simulations and solve the Lagrange multiplier problem for V_ℓ and hence n_ℓ.

3. Efficient simulation

To get our estimator, we have a lot of sample paths to generate, with different values of τ:

$$\mathbb{Q}_b := \sum_{\ell=0}^{L} Q_\ell = \mathbb{E}\left[Z_{\tau_0}\right] + \sum_{\ell=1}^{L} \mathbb{E}\left[Z_{\tau_\ell} - Z_{\tau_{\ell-1}}\right],$$

with

$$Q_\ell := \mathbb{E}\left[Z_{\tau_\ell} - Z_{\tau_{\ell-1}}\right] \approx \frac{1}{n_\ell} \sum_{r=1}^{n_\ell} \left[Z_{\tau_\ell}^{(r)}(T) - Z_{\tau_{\ell-1}}^{(r)}(T)\right]. \qquad (29)$$

QUESTION: how can this be made more efficient than just using paths with $\tau = \tau_L$?

KEY IDEA: Generate sample paths to estimate Q_ℓ with low sample variance.

3. Efficient simulation – variance reduction

It is always the case that

$$Q_\ell \equiv \mathbb{E}\left[Z_\ell - Z_{\ell-1}\right] = \mathbb{E}\left[Z_\ell\right] - \mathbb{E}\left[Z_{\ell-1}\right]. \tag{30}$$

So, we *could* generate $\widehat{Q}_\ell$ by first sampling paths for

$$\mathbb{E}\left[Z_\ell\right] \approx \frac{1}{n_\ell}\sum_{r=1}^{n_\ell} Z_{\tau_\ell}^{(r)}, \tag{31}$$

and then sampling paths for

$$\mathbb{E}\left[Z_{\ell-1}\right] \approx \frac{1}{n_\ell}\sum_{r=1}^{n_\ell} Z_{\tau_{\ell-1}}^{(r)} \tag{32}$$

and taking the difference.

3. Efficient simulation – variance reduction

BUT, recall that the accuracy of

$$\widehat{Q}_\ell = \frac{1}{n_\ell}\sum_{r=1}^{n_\ell}\left[Z_{\tau_\ell}^{(r)} - Z_{\tau_{\ell-1}}^{(r)}\right] \tag{33}$$

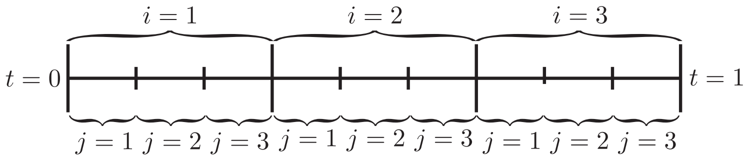
is quantified by the estimator variance:

$$\widehat{V}_\ell = \frac{1}{n_\ell}\text{Var}\left[Z_\ell - Z_{\ell-1}\right] = \frac{1}{n_\ell}\left\{\text{Var}[Z_\ell] + \text{Var}[Z_{\ell-1}] - 2\cdot\text{Cov}[Z_\ell, Z_{\ell-1}]\right\}. \tag{34}$$

THE SMART IDEA: Force a *positive correlation* between $Z_{\tau_\ell}^{(r)}$ and $Z_{\tau_{\ell-1}}^{(r)}$ by keeping the r-th sample paths as similar to each other as possible.

Multi-level Monte Carlo

e.g. for $K = 3$, $\tau_0 = 1$ and $\ell = 2$:


$i = 1$
$i = 2$
$i = 3$
$t = 0$
$t = 1$
$j = 1$ $j = 2$ $j = 3$ $j = 1$ $j = 2$ $j = 3$ $j = 1$ $j = 2$ $j = 3$

Suppose we want to simulate a single pair of sample paths on level ℓ. Path with $\tau_\ell = \tau_0/K^\ell$ is the **fine** path. Path with $\tau_{\ell-1} = \tau_0/K^{\ell-1}$ is the **coarse** path.

REMINDER: Each of the τ-leaping paths (with time-steps τ_ℓ and $\tau_{\ell-1}$) are constructed by generating Poisson random variables (with different rates) in order to simulate Poisson processes (with different rates).
QUESTION: How do we keep the paths together?

Coupling paths to keep them close

Suppose we want to generate random variates from two Poisson processes:

- one with rate 13.0;
- the other with rate 13.1.

We could use two independent, unit rate Poisson processes and set:

- $Z_{13.0}(t) = Y_1(13.0t)$;
- $Z_{13.1}(t) = Y_2(13.1t)$.

This means the processes are independent so that

$$\mathrm{Var}(Z_{13.1}(t) - Z_{13.0}(t)) = \mathrm{Var}(Z_{13.1}(t)) + \mathrm{Var}(Z_{13.0}(t)) = 26.1t. \quad (35)$$

Coupling paths to keep them close

Or, use the Poisson thickening property:

$$\mathcal{P}_1(a+b) \sim \mathcal{P}_2(a) + \mathcal{P}_3(b). \tag{36}$$

So we could set:

- $Z_{13.0}(t) = Y_1(13.0t)$;
- $Z_{13.1}(t) = Y_1(13.0t) + Y_2(0.1t)$.

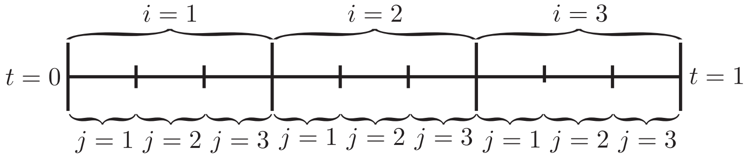
This means the processes are dependent, and

$$\mathrm{Var}(Z_{13.1}(t) - Z_{13.0}(t)) = 0.1t. \tag{37}$$

KEY POINT: Multi-level Monte Carlo applies this logic to every reaction at every time step to keep paths close together.

Multi-level Monte Carlo

e.g. for $K = 3$, $\tau_0 = 1$ and $\ell = 2$:

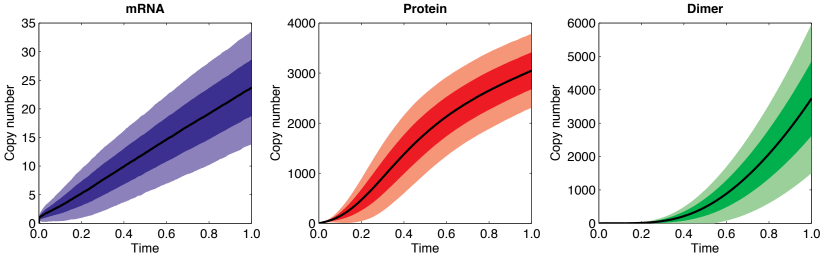

$i = 1$
$i = 2$
$i = 3$
$t = 0$
$t = 1$
$j = 1$ $j = 2$ $j = 3$ $j = 1$ $j = 2$ $j = 3$ $j = 1$ $j = 2$ $j = 3$

- Update the values of the state variables of each path at intervals τ_0 / K^ℓ.
- Update the propensities of the **fine** path (with $\tau = \tau_0 / K^\ell$) each time-step.
- Update the propensities of the **coarse** path (with $\tau = \tau_0 / K^{\ell - 1}$) every K steps.

Multi-level Monte Carlo – Example revisited

$$R_1 : G \xrightarrow{25} M; \quad R_2 : M \xrightarrow{1000} M + P; \quad R_3 : P + P \xrightarrow{0.001} D;$$

$$R_4 : M \xrightarrow{0.1} \emptyset; \quad R_5 : P \xrightarrow{1} \emptyset.$$



Estimating the dimer population at $T = 1$

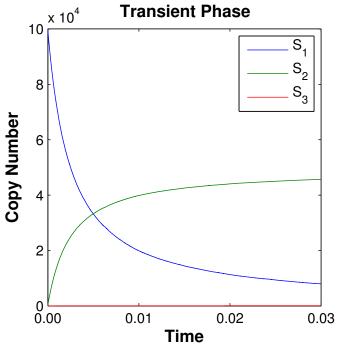
- Gillespie Algorithm – $3,714.0 \pm 1.0$ – 6 hours (21,472 seconds).
- Multi-level Monte Carlo – $3,714.6 \pm 1.0$ – 10 minutes (578 seconds).

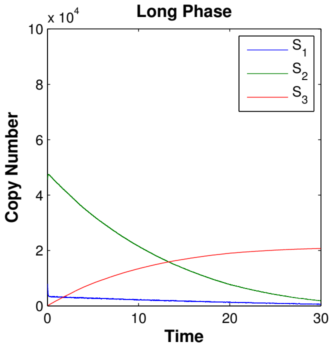
This is a **factor 37** speed up.

A troublesome example: dimerization

$$\begin{aligned} R_1 &: S_1 \xrightarrow{1} \emptyset; & R_2 &: S_2 \xrightarrow{1/25} S_3; \\ R_3 &: S_1 + S_1 \xrightarrow{1/500} S_2; & R_4 &: S_2 \xrightarrow{1/2} S_1 + S_1. \end{aligned} \tag{38}$$

We simulate until $T = 30$ with initial conditions $\mathbf{X}(0) = [10^5, 0, 0]^T$.





An time-adaptive algorithm

Multi-level Monte Carlo vs Gillespie

- Exact simulation method – $20,591.6 \pm 1.0$ – 2,089 seconds.
- Fixed time-step multi-level – $20,591.6 \pm 1.0$ – 632 seconds.

This time we only see a **factor three** speed up.

Problems with a fixed time step algorithm:

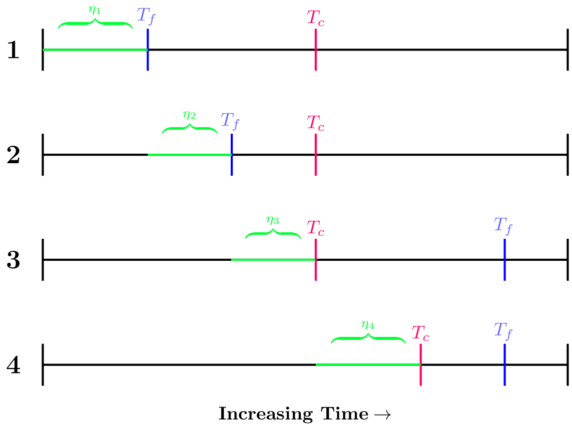
- has to satisfy demands of stiffest timescale;
- currently available algorithms require time-steps between levels to be nested.

Adaptive time-step would ideally:

- adapt to stochastic behaviour of individual sample paths;
- not require time-steps between levels to be nested.

An time-adaptive algorithm[7]: the idea

- Use an adaptive τ-leaping algorithm to generate paths on each level.
- Let $\tau_\ell = \tau_\ell(\epsilon_\ell, \mathbf{X}(t))$ where ϵ_ℓ is a control parameter for level ℓ.



- Adaptive multi-level – $20,591.1 \pm 1.0$ – 70 seconds (**factor 30**).

[7] An adaptive multi-level simulation algorithm for stochastic biological systems. C. Lester, C.A. Yates, M.B. Giles, R.E. Baker (2015) *J. Chem Phys.* 142(2) 024113 or http://arxiv.org/abs/1409.1838

Summary

The multi-level method can be efficient in generating system statistics.

- We have constructed a modified direct method approach that is easier to implement and provides computational savings.
- It can be parallelised (for example, with GPUs) which can further reduce simulation time.

To use the multi-level method, a number of decisions have to be made:

- the choice of levels in the algorithm. This affects both the simulation time and the bias of $\mathbb{Q}_b$, and is determined by both L and the values of $\tau_0, \tau_1, \ldots, \tau_L$;
- the values the target variance, $\widehat{V}_\ell$, should take on each level, ℓ. This ensures statistical accuracy, and also affects the simulation time;
- the choices of simulation techniques for the base level (0), the correcting levels, $1, \ldots, L$, and (if desired) the final level, $L + 1$.

Initially levels are nested geometrically:

$$\tau_\ell = \tau_{(\ell-1)}/K. \tag{39}$$

Summary

BUT, the multi-level method does not perform well for systems with multiple time scales.

- τ_0, the coarsest resolution, must be sufficiently small to ensure numerical stability and (for efficient simulation reasons) avoid negative populations occurring ‘too often’.
- τ_L, the finest resolution, should be sufficiently large so that the multi-level method is justified. Otherwise the unbiased and simple Gillespie SSA will win.
- As trajectories traverse different regimes, an adaptive approach seems to be the safest way to handle the choosing of τ.
- This method makes no assumptions as to the nature of the system, or indeed the estimator.
- How best to choose the ϵ_ℓ and $\tau_\ell = \tau_\ell(\epsilon_\ell, \mathbf{X}(t))$ is still up for discussion.

Acknowledgements

Thanks to:

- My excellent D.Phil. student, Chris Lester.
- Dr Ruth Baker, co-supervisor of Chris.
- Helpful suggestions from Professor Mike Giles.

References

A guide to efficient discrete-state multi-level simulation of stochastic biological systems (2015) http://arxiv.org/pdf/1412.4069v1.pdf. C. Lester, R. E. Baker, M. B. Giles and C. A. Yates.

An adaptive multi-level simulation algorithm for stochastic biological systems (2015) *J. Chem Phys.* 142(2) 024113 or http://arxiv.org/abs/1409.1838. C. Lester, C. A. Yates, M. B. Giles and R. E. Baker.

Multi-level Monte Carlo for continuous time Markov chains, with applications in biochemical kinetics (2012) *SIAM Multiscale Model. Simul.* 10(1) 146-179. D. Anderson and D. Higham.