Multilevel Monte Carlo for the continuous time Markov chain models arising in biology

David F. Anderson*

*anderson@math.wisc.edu

Department of Mathematics
University of Wisconsin - Madison

Work is joint with Des Higham

MCQMC 2012

February 17th, 2012
Overview

- Most common stochastic models of biochemical reaction systems are continuous time Markov chains.

- Common examples include:
  2. Models of viral infection.
  3. General population models (epidemic, predator-prey, etc.)

- These models are becoming dominant in molecular biology.

Problem: extend multi-level Monte Carlo to this setting.
Why?

1. Why would the be useful?
   - Short answer: the common simulation methods can be very computationally expensive.
   - Many reactions happening over time-frame of interest.

2. Why is this non-trivial?
   - In SDE case, we have a useful representation
     \[
     X(t) = X(0) + \int_0^t b(X(s))ds + \int_0^t \sigma(X(s))dW(s)
     \]
     \[
     Z\ell(t) = Z\ell + \int_0^t b(Z\ell \circ \eta(s))ds + \int_0^t \sigma(Z\ell \circ \eta(s))dW(s).
     \]
   - Two processes are coupled via Brownian motion \(W\).
   - Clear how to simulate two processes with different time steps.
   - We need a pathwise representation to get a good coupling that:
     2.1 Couples closely.
     2.2 Easy to analyze.
The Poisson process

A Poisson process, $Y(\cdot)$:

(a) Let $\{\xi_i\}$ be i.i.d. exponential random variables with parameter one.

(b) Now, put points down on a line with spacing equal to the $\xi_i$:

$$
\begin{array}{cccccc}
\xi_1 & \xi_2 & \xi_3 & \cdots & x \\
\hline
X & X & X & X & X
\end{array}
$$

- Let $Y(t)$ denote the number of points hit by time $t$.
- In the figure above, $Y(t) = 6$. 
The Poisson process

Let $Y$ be a unit rate Poisson process and $\lambda \geq 0$.

Then $Y(\lambda t)$ is a Poisson process with parameter $\lambda$.

There is no reason $\lambda$ needs to be constant in time, in which case

$$Y_\lambda(t) \equiv Y \left( \int_0^t \lambda(s) \, ds \right)$$

is an non-homogeneous Poisson process with intensity $\lambda(t)$. 
Consider the simple system

\[ A + B \rightarrow C \]

where one molecule each of \( A \) and \( B \) is being converted to one of \( C \).

Simple book-keeping: if \( X(t) = (X_A(t), X_B(t), X_C(t))^T \) gives the state at time \( t \),

\[ X(t) = X(0) + R(t) \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}, \]

where

- \( R(t) \) is the \# of times the reaction has occurred by time \( t \), and
- \( X(0) \) is the initial condition.
Assuming intensity or propensity of reaction is
\[ \lambda(X(s)) = \kappa X_A(s) X_B(s), \]
we can model
\[ R(t) = Y \left( \int_0^t \kappa X_A(s) X_B(s) ds \right) \]
where \( Y \) is a unit-rate Poisson point process.

Hence
\[
\begin{pmatrix}
X_A(t) \\
X_B(t) \\
X_C(t)
\end{pmatrix}
\equiv X(t) = X(0) +
\begin{pmatrix}
-1 \\
-1 \\
1
\end{pmatrix}
Y \left( \int_0^t \kappa X_A(s) X_B(s) ds \right).
\]
Build up model: Random time change representation of Tom Kurtz

- Now consider a network of reactions involving $d$ chemical species, $S_1, \ldots, S_d$:

$$
\nu_{k1} S_1 + \nu_{k2} S_2 + \cdots + \nu_{kd} S_d \longrightarrow \nu_{k1}' S_1 + \nu_{k2}' S_2 + \cdots + \nu_{kd}' S_d
$$

Denote reaction vector as

$$
\zeta_k = \nu_k' - \nu_k,
$$

- The intensity (or propensity) of $k$th reaction is $\lambda_k : \mathbb{Z}_{\geq 0}^d \rightarrow \mathbb{R}$.

- By analogy with before

$$
X(t) = X(0) + \sum_k Y_k \left( \int_0^t \lambda_k(X(s)) \, ds \right) \zeta_k,
$$

$Y_k$ are independent, unit-rate Poisson processes.
Example

Consider a model of gene transcription and translation:

\[ G \xrightarrow{25} G + M, \quad \text{(Transcription)} \]

\[ M \xrightarrow{1000} M + P, \quad \text{(Translation)} \]

\[ P + P \xrightarrow{0.001} D, \quad \text{(Dimerization)} \]

\[ M \xrightarrow{0.1} \emptyset, \quad \text{(Degradation of mRNA)} \]

\[ P \xrightarrow{1} \emptyset \quad \text{(Degradation of Protein)} \]

Then, if \( X = [X_M, X_P, X_D]^T \),

\[
X(t) = X(0) + Y_1(25t) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + Y_2 \left( 1000 \int_0^t X_M(s)ds \right) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}
+ Y_3 \left( 0.001 \int_0^t X_P(s)(X_P(s) - 1)ds \right) \begin{bmatrix} 0 \\ -2 \\ 1 \end{bmatrix}
+ Y_4 \left( 0.1 \int_0^t X_M(s)ds \right) \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} + Y_5 \left( 1 \int_0^t X_P(s)ds \right) \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}
\]
Computing Expectations

Problem: Approximate $\mathbb{E}f(X(T))$ to some desired tolerance, $\epsilon > 0$.

Easy!

- Simulate the CTMC exactly,
- generate independent paths, $X_{[i]}(t)$, use the unbiased estimator

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} f(X_{[i]}(t)),$$

- stop when desired confidence interval is $\pm \epsilon$.

Total computational complexity $= (\text{cost per path}) \times (\# \text{ paths})$

$= O(\bar{N} \times \epsilon^{-2})$. 
Tau-leaping: Euler’s method

Recall:

$$X(t) = X(0) + \sum_k Y_k \left( \int_0^t \lambda_k(X(s)) ds \right) \zeta_k.$$ 

**Tau-leaping** is essentially an **Euler approximation** of $$\int_0^t \lambda_k(X(s)) ds$$:

$$Z(h) = Z(0) + \sum_k Y_k \left( \int_0^h \lambda_k(Z(s)) ds \right) \zeta_k$$

$$\approx Z(0) + \sum_k Y_k \left( \lambda_k(Z(0)) h \right) \zeta_k$$

$$\overset{d}{=} Z(0) + \sum_k \text{Poisson} \left( \lambda_k(Z(0)) h \right) \zeta_k.$$
Euler’s method

Path-wise representation for $Z(t)$ generated by Euler’s method is

$$Z(t) = X(0) + \sum_k Y_k \left( \int_0^t \lambda_k(Z \circ \eta(s))ds \right) \zeta_k,$$

where

$$\eta(s) = t_n \text{ if } t_n \leq s < t_{n+1} = t_n + h$$

is a step function giving left endpoints of time discretization.
Return to approximating $\mathbb{E}f(X(T))$

Let

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} f(Z_{L,[i]}(t)).$$

We have

$$\text{MSE}(\hat{\mu}_n) = \text{(Bias)}^2 + \text{(variance)},$$

If have an order one method

\[\text{Total computational complexity} = (\text{cost per path}) \times (\text{# paths}) \]
\[= O(\epsilon^{-1} \times \epsilon^{-2}) \]
\[= O(\epsilon^{-3}).\]
Benefits/drawbacks

Benefits:
1. Can drastically lower the computational complexity of a problem

Drawbacks:
1. Convergence results usually give order of convergence. Can’t give a precise $h_L$.
2. Tau-leaping has problems: what happens if you go negative?
3. Gone away from an unbiased estimator.
Multi-level Monte Carlo

Usual MLMC:

\[ \mathbb{E}f(X(t)) \approx \mathbb{E}[f(Z_L(t)) - f(Z_{L-1}(t))] + \sum_{\ell=\ell_0+1}^{L-1} \mathbb{E}[f(Z_\ell(t)) - f(Z_{\ell-1}(t))] + \mathbb{E}f(Z_{\ell_0}(t)). \]

In our setting:

\[ \mathbb{E}f(X(t)) = \mathbb{E}[f(X(t)) - f(Z_L(t))] + \sum_{\ell=\ell_0+1}^{L} \mathbb{E}[f(Z_\ell(t)) - f(Z_{\ell-1}(t))] + \mathbb{E}f(Z_{\ell_0}(t)). \]

gives an **unbiased** estimator.
Multi-level Monte Carlo: an unbiased estimator

\[ \mathbb{E} f(X(t)) = \mathbb{E} [f(X(t)) - f(Z_L(t))] + \sum_{\ell = \ell_0 + 1}^L \mathbb{E} [f(Z_\ell(t)) - f(Z_{\ell-1}(t))] + \mathbb{E} f(Z_{\ell_0}(t)). \]

Estimators:

\[ \hat{Q}_E \overset{\text{def}}{=} \frac{1}{n_E} \sum_{i=1}^{n_E} (f(X_{[i]}(T)) - f(Z_{L,[i]}(T))), \]

\[ \hat{Q}_\ell \overset{\text{def}}{=} \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} (f(Z_{\ell,[i]}(T)) - f(Z_{\ell-1,[i]}(T))), \quad \text{for } \ell \in \{\ell_0 + 1, \ldots, L\} \]

\[ \hat{Q}_0 \overset{\text{def}}{=} \frac{1}{n_0} \sum_{i=1}^{n_0} f(Z_{\ell_0,[i]}(T)). \]

Question: what is the coupling and the computational cost?
How do we couple Poisson processes

Suppose I want to generate:

▶ A Poisson process with intensity 13.1.
▶ A Poisson process with intensity 13.

We could let \( Y_1 \) and \( Y_2 \) be independent, unit-rate Poisson processes, and set

\[
Z_{13.1}(t) = Y_1(13.1t), \\
Z_{13}(t) = Y_2(13t),
\]

Using this representation, these processes are independent and, hence, not coupled.

The variance of difference is large:

\[
\text{Var}(Z_{13.1}(t) - Z_{13}(t)) = \text{Var}(Y_1(13.1t)) + \text{Var}(Y_2(13t)) = 26.1t.
\]
How do we generate processes simultaneously

Suppose I want to generate:

- A Poisson process with intensity 13.1.
- A Poisson process with intensity 13.

We could let $Y_1$ and $Y_2$ be independent unit-rate Poisson processes, and set

$$Z_{13.1}(t) = Y_1(13t) + Y_2(0.1t)$$
$$Z_{13}(t) = Y_1(13t),$$

The variance of difference is much smaller:

$$\text{Var}(Z_{13.1}(t) - Z_{13}(t)) = \text{Var}(Y_2(0.1t)) = 0.1t.$$
How do we generate processes simultaneously

More generally, suppose we want

1. non-homogeneous Poisson process with intensity \( f(t) \) and
2. non-homogeneous Poisson process with intensity \( g(t) \).

Let \( Y_1, Y_2, \) and \( Y_3 \) be independent, unit-rate Poisson processes and define

\[
Z_f(t) = Y_1 \left( \int_0^t f(s) \wedge g(s) \, ds \right) + Y_2 \left( \int_0^t f(s) - (f(s) \wedge g(s)) \, ds \right),
\]

\[
Z_g(t) = Y_1 \left( \int_0^t f(s) \wedge g(s) \, ds \right) + Y_3 \left( \int_0^t g(s) - (f(s) \wedge g(s)) \, ds \right),
\]
Back to our processes

\[ X(t) = X(0) + \sum_k Y_k \left( \int_0^t \lambda_k(X(s)) ds \right) \zeta_k, \]

\[ Z(t) = X(0) + \sum_k Y_k \left( \int_0^t \lambda_k(Z \circ \eta(s)) ds \right) \zeta_k. \]

Now couple

\[ X(t) = X(0) + \sum_k Y_{k,1} \left( \int_0^t \lambda_k(X(s)) \land \lambda_k(Z_\ell \circ \eta_\ell(s)) ds \right) \zeta_k \]

\[ + \sum_k Y_{k,2} \left( \int_0^t \lambda_k(X(s)) - \lambda_k(X(s)) \land \lambda_k(Z_\ell \circ \eta_\ell(s)) ds \right) \zeta_k \]

\[ Z_\ell(t) = Z_\ell(0) + \sum_k Y_{k,1} \left( \int_0^t \lambda_k(X(s)) \land \lambda_k(Z_\ell \circ \eta_\ell(s)) ds \right) \zeta_k \]

\[ + \sum_k Y_{k,3} \left( \int_0^t \lambda_k(Z_\ell \circ \eta_\ell(s)) - \lambda_k(X(s)) \land \lambda_k(Z_\ell \circ \eta_\ell(s)) ds \right) \zeta_k \]

Algorithm for simulation is equivalent to simulating CTMC.
For approximate processes

\[ Z_\ell(t) = Z_\ell(0) + \sum_k Y_{k,1} \left( \int_0^t \lambda_k(Z_\ell \circ \eta_\ell(s)) \land \lambda_k(Z_{\ell-1} \circ \eta_{\ell-1}(s)) ds \right) \zeta_k \]

\[ + \sum_k Y_{k,2} \left( \int_0^t \lambda_k(Z_\ell \circ \eta_\ell(s)) - \lambda_k(Z_\ell \circ \eta_\ell(s)) \land \lambda_k(Z_{\ell-1} \circ \eta_{\ell-1}(s)) ds \right) \zeta_k \]

\[ Z_{\ell-1}(t) = Z_{\ell-1}(0) + \sum_k Y_{k,1} \left( \int_0^t \lambda_k(Z_{\ell-1} \circ \eta_{\ell-1}(s)) \land \lambda_k(Z_{\ell-1} \circ \eta_{\ell-1}(s)) ds \right) \zeta_k \]

\[ + \sum_k Y_{k,3} \left( \int_0^t \lambda_k(Z_{\ell-1} \circ \eta_{\ell-1}(s)) - \lambda_k(Z_\ell \circ \eta_\ell(s)) \land \lambda_k(Z_{\ell-1} \circ \eta_{\ell-1}(s)) ds \right) \zeta_k, \]

Algorithm for simulation is equivalent to Euler’s method.

What about the variance and, hence, computational cost?
Multi-level Monte Carlo: chemical kinetic setting

Can prove:

**Theorem (Anderson, Higham 2011)**

Suppose $(X, Z_\ell)$ satisfy coupling. Then,

$$
\sup_{t \leq T} \mathbb{E}|X(t) - Z_\ell(t)|^2 \leq C_1(T)N^{-\rho}h_\ell + C_2(T)h_\ell^2.
$$

**Theorem (Anderson, Higham 2011)**

Suppose $(Z_\ell, Z_{\ell-1})$ satisfy coupling. Then,

$$
\sup_{t \leq T} \mathbb{E}|Z_\ell(t) - Z_{\ell-1}(t)|^2 \leq C_1(T)N^{-\rho}h_\ell + C_2(T)h_\ell^2.
$$

---

Multi-level Monte Carlo: an unbiased estimator

For well chosen $n_0$, $n_\ell$, and $n_E$. We have

$$\text{Var}(\hat{Q}) = \text{Var} \left( \hat{Q}_E + \sum_{\ell=\ell_0+1}^L \hat{Q}_\ell + \hat{Q}_0 \right) = O(\epsilon^2),$$

with

$$\text{Comp. cost} = \left[ \epsilon^{-2} (N^{-\rho} h_L + h_L^2) \right] \bar{N} + \epsilon^{-2} \left( h_{\ell_0}^{-1} + \ln(\epsilon)^2 N^{-\rho} + \ln(\epsilon)^{-1} \frac{1}{M-1} h_{\ell_0} \right).$$
Some observations:

1. Weak error plays no role in analysis: free to choose $h_L$.

2. Common problems associated with tau-leaping
   - Negativity of species numbers,
   - does not matter. Just define approximate process in a sensible way.

3. The method is unbiased.
Example

Consider a model of gene transcription and translation:

\[
G \xrightarrow{25} G + M,
\]

\[
M \xrightarrow{1000} M + P,
\]

\[
P + P \xrightarrow{0.001} D,
\]

\[
M \xrightarrow{0.1} \emptyset,
\]

\[
P \xrightarrow{1} \emptyset.
\]

Suppose:

1. initialize with: \( G = 1, M = 0, P = 0, D = 0, \)

2. want to estimate the expected number of dimers at time \( T = 1, \)

3. to an accuracy of \( \pm 1.0 \) with 95% confidence.
Example

Method: Exact algorithm with crude Monte Carlo.

<table>
<thead>
<tr>
<th>Approximation</th>
<th># paths</th>
<th>CPU Time</th>
<th># updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>3,714.2 ± 1.0</td>
<td>4,740,000</td>
<td>149,000 CPU S (41 hours!)</td>
<td>8.27 × 10^{10}</td>
</tr>
</tbody>
</table>

Method: Euler tau-leaping with crude Monte Carlo.

<table>
<thead>
<tr>
<th>Step-size</th>
<th>Approximation</th>
<th># paths</th>
<th>CPU Time</th>
<th># updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>h = 3^{-7}</td>
<td>3,712.3 ± 1.0</td>
<td>4,750,000</td>
<td>13,374.6 S</td>
<td>6.2 × 10^{10}</td>
</tr>
<tr>
<td>h = 3^{-6}</td>
<td>3,707.5 ± 1.0</td>
<td>4,750,000</td>
<td>6,207.9 S</td>
<td>2.1 × 10^{10}</td>
</tr>
<tr>
<td>h = 3^{-5}</td>
<td>3,693.4 ± 1.0</td>
<td>4,700,000</td>
<td>2,803.9 S</td>
<td>6.9 × 10^{9}</td>
</tr>
<tr>
<td>h = 3^{-4}</td>
<td>3,654.6 ± 1.0</td>
<td>4,650,000</td>
<td>1,219.0 S</td>
<td>2.6 × 10^{9}</td>
</tr>
</tbody>
</table>

Method: unbiased MLMC with ℓ_0 = 2, and M and L detailed below.

<table>
<thead>
<tr>
<th>Step-size parameters</th>
<th>Approx.</th>
<th>CPU Time</th>
<th># updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>M = 3, L = 6</td>
<td>3,713.9 ± 1.0</td>
<td>1,063.3 S</td>
<td>1.1 × 10^{9}</td>
</tr>
<tr>
<td>M = 3, L = 5</td>
<td>3,714.7 ± 1.0</td>
<td>1,114.9 S</td>
<td>9.4 × 10^{8}</td>
</tr>
<tr>
<td>M = 3, L = 4</td>
<td>3,714.2 ± 1.0</td>
<td>1,656.6 S</td>
<td>1.0 × 10^{9}</td>
</tr>
<tr>
<td>M = 4, L = 4</td>
<td>3,714.2 ± 1.0</td>
<td>1,334.8 S</td>
<td>1.1 × 10^{9}</td>
</tr>
<tr>
<td>M = 4, L = 5</td>
<td>3,713.8 ± 1.0</td>
<td>1,014.9 S</td>
<td>1.1 × 10^{9}</td>
</tr>
</tbody>
</table>

▶ the exact algorithm with crude Monte Carlo demanded 140 times more CPU time than our unbiased MLMC estimator!
Method: Exact algorithm with crude Monte Carlo.

<table>
<thead>
<tr>
<th>Approximation</th>
<th># paths</th>
<th>CPU Time</th>
<th># updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>3,714.2 ± 1.0</td>
<td>4,740,000</td>
<td>149,000 CPU S (41 hours!)</td>
<td>$8.27 \times 10^{10}$</td>
</tr>
</tbody>
</table>

Unbiased Multi-level Monte Carlo with $M = 3$, $L = 5$, and $\ell_0 = 2$.

<table>
<thead>
<tr>
<th>Level</th>
<th># paths</th>
<th>CPU Time</th>
<th>Var. estimator</th>
<th># updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(X, Z_{3-5})$</td>
<td>3,900</td>
<td>279.6 S</td>
<td>0.0658</td>
<td>$6.8 \times 10^7$</td>
</tr>
<tr>
<td>$(Z_{3-5}, Z_{3-4})$</td>
<td>30,000</td>
<td>49.0 S</td>
<td>0.0217</td>
<td>$8.8 \times 10^7$</td>
</tr>
<tr>
<td>$(Z_{3-4}, Z_{3-3})$</td>
<td>150,000</td>
<td>71.7 S</td>
<td>0.0179</td>
<td>$1.5 \times 10^8$</td>
</tr>
<tr>
<td>$(Z_{3-3}, Z_{3-2})$</td>
<td>510,000</td>
<td>112.3 S</td>
<td>0.0319</td>
<td>$1.7 \times 10^8$</td>
</tr>
<tr>
<td>Tau-leap with $h = 3^{-2}$</td>
<td>8,630,000</td>
<td>518.4 S</td>
<td>0.1192</td>
<td>$4.7 \times 10^8$</td>
</tr>
<tr>
<td>Totals</td>
<td>N.A.</td>
<td>1031.0 S</td>
<td>0.2565</td>
<td>$9.5 \times 10^8$</td>
</tr>
</tbody>
</table>
Some conclusions about this method

1. Gillespie’s algorithm is by far the most common way to compute expectations:
   1.1 Means.
   1.2 Variances.
   1.3 Probabilities.

2. The new method (MLMC) also performs this task with no bias (exact).

3. Will commonly be many orders of magnitude faster than usual methods.

4. Applicable to essentially all continuous time Markov chains:

\[ X(t) = X(0) + \sum_{k} Y_k \left( \int_{0}^{t} \lambda_k(X(s)) ds \right) \zeta_k. \]

5. Makes no use of any specific structure or scaling in the problem.
Another example: Viral infection

Let

1. \( T = \text{viral template} \).
2. \( G = \text{viral genome} \).
3. \( S = \text{viral structure} \).
4. \( V = \text{virus} \).

Reactions:

1. \( \frac{T}{\kappa_1} + \text{stuff} \rightarrow T + G \quad \kappa_1 = 1 \)
2. \( G \rightarrow T \quad \kappa_2 = 0.025 \)
3. \( T + \text{stuff} \rightarrow T + S \quad \kappa_3 = 1000 \)
4. \( T \rightarrow \emptyset \quad \kappa_4 = 0.25 \)
5. \( S \rightarrow \emptyset \quad \kappa_5 = 2 \)
6. \( G + S \rightarrow V \quad \kappa_6 = 7.5 \times 10^{-6} \)

Another example: Viral infection

Stochastic equations for $X = (X_G, X_S, X_T, X_V)$ are

$$X_1(t) = X_1(0) + Y_1 \left( \int_0^t X_3(s)ds \right) - Y_2 \left( 0.025 \int_0^t X_1(s)ds \right) - Y_6 \left( 7.5 \times 10^{-6} \int_0^t X_1(s)X_2(s)ds \right)$$

$$X_2(t) = X_2(0) + Y_3 \left( 1000 \int_0^t X_3(s)ds \right) - Y_5 \left( 2 \int_0^t X_2(s)ds \right) - Y_6 \left( 7.5 \times 10^{-6} \int_0^t X_1(s)X_2(s)ds \right)$$

$$X_3(t) = X_3(0) + Y_2 \left( 0.025 \int_0^t X_1(s)ds \right) - Y_4 \left( 0.25 \int_0^t X_3(s)ds \right)$$

$$X_4(t) = X_4(0) + Y_6 \left( 7.5 \times 10^{-6} \int_0^t X_1(s)X_2(s)ds \right).$$
Another example: Viral infection

Reactions:

R1) $T + \text{stuff} \xrightarrow{\kappa_1} T + G$  \quad \kappa_1 = 1
R2) $G \xrightarrow{\kappa_2} T$  \quad \kappa_2 = 0.025
R3) $T + \text{stuff} \xrightarrow{\kappa_3} T + S$  \quad \kappa_3 = 1000
R4) $T \xrightarrow{\kappa_4} \emptyset$  \quad \kappa_4 = 0.25
R5) $S \xrightarrow{\kappa_5} \emptyset$  \quad \kappa_5 = 2
R6) $G + S \xrightarrow{\kappa_6} V$  \quad \kappa_6 = 7.5 \times 10^{-6}

If $T > 0$,

- reactions 3 and 5 are much faster than others.
- $\text{M/M/\infty queue} \implies S$ is approximately Poisson($500 \times T$).

Can average out to get approximate process $Z(t)$. 

Another example: Viral infection

Approximate process satisfies.

\[ Z_1(t) = X_1(0) + Y_1 \left( \int_0^t Z_3(s) \, ds \right) - Y_2 \left( 0.025 \int_0^t Z_1(s) \, ds \right) \]
\[ - Y_6 \left( 3.75 \times 10^{-3} \int_0^t Z_1(s) Z_3(s) \, ds \right) \]  

\[ Z_3(t) = X_3(0) + Y_2 \left( 0.025 \int_0^t Z_1(s) \, ds \right) - Y_4 \left( 0.25 \int_0^t Z_3(s) \, ds \right) \]  

\[ Z_4(t) = X_4(0) + Y_6 \left( 3.75 \times 10^{-3} \int_0^t Z_1(s) Z_3(s) \, ds \right) . \]  

Now use

\[ \mathbb{E} f(X(t)) = \mathbb{E} [f(X(t)) - f(Z(t))] + \mathbb{E} f(Z(t)). \]
Another example: Viral infection

\[ X(t) = X(0) + Y_{1,1} \left( \int_0^t \min\{X_3(s), Z_3(s)\} \, ds \right) \zeta_1 + Y_{1,2} \left( \int_0^t X_3(s) - \min\{X_3(s), Z_3(s)\} \, ds \right) \zeta_1 \\
+ Y_{2,1} \left( 0.025 \int_0^t \min\{X_1(s), Z_1(s)\} \, ds \right) \zeta_2 + Y_{2,2} \left( 0.025 \int_0^t X_1(s) - \min\{X_1(s), Z_1(s)\} \, ds \right) \zeta_2 \\
+ Y_3 \left( 1000 \int_0^t X_3(s) \, ds \right) \zeta_3 \\
+ Y_{4,1} \left( 0.25 \int_0^t \min\{X_3(s), Z_3(s)\}(s) \, ds \right) \zeta_4 + Y_{4,2} \left( 0.25 \int_0^t X_3(s) - \min\{X_3(s), Z_3(s)\}(s) \, ds \right) \zeta_4 \\
+ Y_5 \left( 2 \int_0^t X_2(s) \, ds \right) \zeta_5 \\
+ Y_{6,1} \left( \int_0^t \min\{\lambda_6(X(s)), \Lambda_6(Z(s))\} \, ds \right) \zeta_6 - Y_{6,2} \left( \int_0^t \lambda_6(X(s)) - \min\{\lambda_6(X(s)), \Lambda_6(Z(s))\} \, ds \right) \zeta_6 \\

\]

\[ Z(t) = Y_{1,1} \left( \int_0^t \min\{X_3(s), Z_3(s)\} \, ds \right) \zeta_1 + Y_{1,3} \left( \int_0^t Z_3(s) - \min\{X_3(s), Z_3(s)\} \, ds \right) \zeta_1 \\
+ Y_{2,1} \left( 0.025 \int_0^t \min\{X_1(s), Z_1(s)\} \, ds \right) \zeta_2 + Y_{2,3} \left( 0.025 \int_0^t Z_1(s) - \min\{X_1(s), Z_1(s)\} \, ds \right) \zeta_2 \\
+ Y_{4,1} \left( 0.25 \int_0^t \min\{X_3(s), Z_3(s)\}(s) \, ds \right) \zeta_4 + Y_{4,3} \left( 0.25 \int_0^t Z_3(s) - \min\{X_3(s), Z_3(s)\}(s) \, ds \right) \zeta_4 \\
+ Y_{6,1} \left( \int_0^t \min\{\lambda_6(X(s)), \Lambda_6(Z(s))\} \, ds \right) \zeta_6 - Y_{6,3} \left( \int_0^t \lambda_6(Z(s)) - \min\{\lambda_6(X(s)), \Lambda_6(Z(s))\} \, ds \right) \zeta_6, \]
Another example: Viral infection

Suppose want $E X_{virus}(20)$

Given $T(0) = 10$, all others zero.

Method: Exact algorithm with crude Monte Carlo.

<table>
<thead>
<tr>
<th>Approximation</th>
<th># paths</th>
<th>CPU Time</th>
<th># updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.85 ± 0.07</td>
<td>75,000</td>
<td>24,800 CPU S</td>
<td>$1.45 \times 10^{10}$</td>
</tr>
</tbody>
</table>

Method: $E f(X(t)) = E[f(X(t)) - f(Z(t))] + E f(Z(t))$.

<table>
<thead>
<tr>
<th>Approximation</th>
<th>CPU Time</th>
<th># updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.91 ± 0.07</td>
<td>1,118.5 CPU S</td>
<td>$2.41 \times 10^{8}$</td>
</tr>
</tbody>
</table>

Exact + crude Monte Carlo used:

1. 60 times more total steps.
2. 22 times more CPU time.
Thanks!

References:


   Available at arXiv.org:1107.2181. Also on my website: www.math.wisc.edu/~anderson.

Funding:

1. NSF Grant DMS-1009275.
2. Thanks to Art Owen.