Continuous time Markov chain models used in biology: models and simulation

David F. Anderson*

*anderson@math.wisc.edu

Department of Mathematics
University of Wisconsin - Madison

The Mathematical Biosciences Institute
Summer Graduate Workshop 2012
Broad Outline

1. Brief comments on assumed knowledge.

2. (Morning 1) Develop mathematical models of interest
   – continuous time Markov chains of chemical (population) processes.
   2.1 These are discrete time Markov chains with exponential holding times.
   2.2 Will start with a brief introduction to DTMCs.

3. (Morning 1) Discuss different, equivalent mathematical representations for the models.

4. (Morning 1) Discuss different simulation methods for the generation of exact sample paths.

5. (Morning 2) Discuss Monte Carlo methods.

6. (Morning 2) Discuss gradient estimation and variance reduction techniques.

7. Maybe more?
Knowledge I will assume

First, some facts that I will assume you know.

I will build essentially everything from two random variables:
1. uniform(0,1) and
2. exponential with rate $\lambda > 0$.

- A uniform(0,1) random variable has density

$$f(x) = \begin{cases} 1 & x \in [0, 1] \\ 0 & \text{else} \end{cases}.$$ 

Hence, if $U \sim \text{uniform}(0, 1)$, then for $0 \leq a \leq b \leq 1$,

$$P \{U \in [a, b]\} = \int_a^b 1 \, dx = b - a.$$ 

Can simulate any discrete random variable with the generation of a uniform (picture).
Simulating a discrete random variable

Theorem
Let $U$ be uniformly distributed on the interval $(0, 1)$. Suppose that $p_k \geq 0$ for each $k \in \{0, 1, \ldots, \}$, and that $\sum_k p_k = 1$. Suppose $X$ is a random variable with probability mass function \{p_k\}. Define

$$q_k = P\{X \leq k\} = \sum_{i=0}^{k} p_i.$$ 

Let

$$\hat{X} = \min\{k \mid q_k \geq U\}.$$ 

Then,

$$P\{\hat{X} = k\} = p_k.$$ 

Thus, $\hat{X}$ has same distribution as $X$: that is, prob. mass function \{p_k\}. 
A random variable $X$ has an exponential distribution with parameter $\lambda > 0$ if it has a probability density function

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & , \quad x \geq 0 \\ \ 0 & , \quad \text{else} \end{cases}.$$  

For an exponential random variable with a parameter of $\lambda > 0$,

$$\mathbb{E}[X] = \frac{1}{\lambda} \quad \text{and} \quad \text{Var}(X) = \frac{1}{\lambda^2}.$$
The exponential random variable

**Lemma**
If for $i = 1, \ldots, n$, the random variables $X_i \sim \text{Exp}(\lambda_i)$ are independent, then

$$X_0 \equiv \min_i \{X_i\} \sim \text{Exp}(\lambda_0), \quad \text{where} \quad \lambda_0 = \sum_{i=1}^{n} \lambda_i.$$  

**Lemma**
For $i = 1, \ldots, n$, let the random variables $X_i \sim \text{Exp}(\lambda_i)$ be independent. Let $j$ be the index of the minimum of $\{X_i\}$. Then $j$ is a discrete random variable with probability mass function

$$P\{j = i\} = \frac{\lambda_i}{\lambda_0}, \quad \text{where} \quad \lambda_0 = \sum_{i=1}^{n} \lambda_i.$$  

First analysis exercise (when we need a break): let’s prove these facts.
How to generate an exponential

Suppose we want an $\exp(\lambda)$ random variable from a uniform.

**Lemma**

If $U$ is uniform $(0, 1)$ and $\lambda > 0$, then

$$X = \ln(1/U)/\lambda,$$

is exponentially distributed with a parameter of $\lambda$.

**Proof.**

For $t \geq 0$,

\[
P\{X \leq t\} = P\{\ln(1/U)/\lambda \leq t\}
= P\{1/U \leq e^{\lambda t}\}
= P\{e^{-\lambda t} \leq U\}
= 1 - e^{-\lambda t}.
\]
Loss of memory property

Lemma
For an exponential random variable, $X$, and any $t, s > 0$,

$$P\{X > t + s \mid X > s\} = P\{X > t\},$$

and exponential random variables are the only random variable with this property.

More background material later on Poisson processes.

Let’s start building models.
What type of models: biochemical reactions

Substrate-enzyme kinetics:

\[ S + E \leftrightarrow SE \rightarrow P + E. \]

Gene transcription & translation:

\[ G \rightarrow G + M \quad \text{transcription} \]
\[ M \rightarrow M + P \quad \text{translation} \]
\[ M \rightarrow \emptyset \quad \text{degradation} \]
\[ P \rightarrow \emptyset \quad \text{degradation} \]
\[ G + P \leftrightarrow B \quad \text{Binding/unbinding of Gene} \]

Cartoon representation:

Modeling choices:

1. **Discrete** vs. continuous state space.
2. Deterministic vs **stochastic** dynamics.

---

\(^{1}\) J. Paulsson, Physics of Life Reviews, 2, 2005 157 – 175.
Modeling. First things first: discrete time Markov chains

- We will first consider a process, $X_n$, in which time, $n$, must satisfy

  $$n \in \{0, 1, 2 \ldots \}.$$

- Thus, we can talk about $X_0$, $X_1$, $X_{17}$, etc.

- We call the values that $X_n$ can take the state space and denote it by $S$.

- $S$ is said to be discrete if it is either finite or countably infinite.

- We will always assume that $S$ is discrete:
  - will typically be modeling molecular, or some other population, counts.
  - hence $S = \mathbb{Z}^d_{\geq 0}$ (for appropriate $d$)
Example

► Suppose a frog can jump between three lily pads, labeled 1, 2, and 3.

► We suppose that if the frog is on lily pad number 1, it will next jump to lily pad number 2 with a probability of one.

► Similarly, if the frog is on lily pad number 3, it will next jump to lily pad number 2.

► However, when the frog is on lily pad number 2, it will next jump to lily pad 1 with probability $1/4$, and to lily pad three with probability $3/4$.

► We can depict this process graphically via

\[
\begin{align*}
1 & \xleftrightarrow{1/4} 2 \\
& \uparrow 1 \\
& \downarrow 3/4 \\
2 & \xleftrightarrow{1} 3
\end{align*}
\]
Discrete time Markov chains

Example continued:

\[ \begin{array}{ccc}
1 & \overset{1/4}{\leftrightarrow} & 2 \\
1 & \downarrow & 3/4 \\
2 & \overset{1}{\leftrightarrow} & 3
\end{array} \]

1. We let \( X_n \) denote the position of the frog after the \( n \)th jump, and assume that \( X_0 = 1 \).

2. We then intuitively have

\[
P\{X_0 = 1, X_1 = 2, X_2 = 3\} = 1 \times 3/4 = 3/4,
\]

whereas

\[
P\{X_0 = 1, X_1 = 3\} = 0.
\]
Discrete time Markov chains

- Computing probabilities of the form

\[ P\{X_0 = i_0, X_1 = i_1, \ldots, X_n = i_n\}, \]

can be difficult for a general stochastic process.

- It is useful to assume the process has some added structure.

- A common choice for such structure is the assumption that the processes satisfies the *Markov property*:

\[ P\{X_n = i_n \mid X_0 = i_0, \ldots, X_{n-1} = i_{n-1}\} = P\{X_n = i_n \mid X_{n-1} = i_{n-1}\}, \quad (1) \]

which says that the distribution of future states only depends upon the current state, and not on the full history of the process.

- Any process \(X_n, n \geq 0\), satisfying the Markov property (1) is called a *discrete time Markov chain*. Note that the processes described in the previous Examples are both discrete time Markov chains.
Discrete time Markov chains

Definition
The one-step transition probability of a Markov chain from state $i$ to state $j$, denoted by $p_{ij}$, is

$$p_{ij} \overset{\text{def}}{=} P\{X_{n+1} = j \mid X_n = i\}.$$ 

Note that the values are time independent (time homogeneity).

The one-step transition probabilities are most conveniently expressed in matrix form.

Definition
The transition matrix $P$ for a Markov chain with state space $S = \{1, 2, \ldots, N\}$ and one-step transition probabilities $p_{ij}$ is the $N \times N$ matrix

$$P \overset{\text{def}}{=} \begin{pmatrix}
p_{11} & p_{12} & \cdots & p_{1N} \\
p_{21} & p_{22} & \cdots & p_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
p_{N1} & p_{N2} & \cdots & p_{NN}
\end{pmatrix}.$$ 

If the state space $S$ is infinite, then $P$ is formally defined to be the infinite matrix with $i, j$th component $p_{ij}$. 
Discrete time Markov chains

Example
Consider a gene that can be repressed by a protein.

▶ By $X_n = 0$, we mean the gene is free at time $n$, and
▶ by $X_n = 1$ we mean that the gene is repressed.

We make the following assumptions:

1. If the gene is free at time $n$, there will be a probability of $p \geq 0$ that it is repressed at time $n + 1$.
2. If the gene is represses at time $n$, there will be a probability of $q \geq 0$ that it is free at time $n + 1$.

In this setting $X_n$ can be modeled as a discrete time Markov chain with finite state space $S = \{0, 1\}$. The transition matrix is

$$P = \begin{bmatrix} 1 - p & p \\ q & 1 - q \end{bmatrix},$$

(2)

where the first row/column is associated with state 0. Note that any two state discrete time Markov chain has a transition matrix of the form (2). □.
Discrete time Markov chains

Examples are endless: picture on board of general states.
Simulating discrete time Markov chains

Consider a DTMC with

1. Initial distribution $\alpha$:

   $$P\{X_0 = j\} = \alpha_j, \quad j \in S,$$

2. transition matrix $P$:

   $$P_{ij} = p_{ij}.$$

The problem is to now simulate a DTMC for a given choice of $\alpha$ and $\{p_{ij}\}$ using more elementary building blocks: uniform random variables.

NOTE: throughout, I take the perspective that the uniform random variables constructed using software (like Matlab) are actually uniformly distributed on $[0, 1]$. 
Simulating a discrete time Markov chain

- We let \( \{ U_0, U_1, \ldots \} \) be independent uniform\((0,1)\) random variables.
- We generate \( X_0 \) from \( U_0 \) using the initial distribution \( \alpha \).
- Next, we note that,
  \[
P\{ X_1 = j \mid X_0 = i \} = p_{ij}.
  \]

Therefore, conditioned upon \( X_0 = i \),

the random variable \( X_1 \) is a discrete random variable with probability mass function determined by the \( i \)th row of the transition matrix \( P \).

- We may then generate \( X_1 \) using \( U_1 \) (conditioned on knowing \( X_0 \)).
- Continuing in this manner constructs the Markov chain \( X_n \).
Simulating a discrete time Markov chain

Some points:

1. each choice of sequence of uniform random variables \( \{U_0, U_1, \ldots \} \) will correspond with a path of the process \( X_n, n \geq 0 \) since:

\[
X_n = f(X_{n-1}, U_n).
\]

2. This algorithm is already one half of the well known “Gillespie Algorithm” used in the generation of sample paths in the continuous time Markov chain setting that will be studied next.
Now we want to add a new wrinkle to the previous model: holding times.

Thus,

1. We still have a discrete state space $S$, which the process is hopping around on, but,

2. now the process will stay, or “hold”, in state $j$ for a random amount of time that depends upon $j \in S$.

- We still want the process to satisfy the Markov property: the distribution of the future is only dependent upon the present.

- We will also assume time homogeneity (nothing depends upon $t$).

- Does this tell us what RV to use for holding time?
Holdings times

- Suppose \( X(0) = x \) and
- Let \( T_x \) denote the time we transition away from state \( x \).
- To find the distribution of \( T_x \), we let \( s, t \geq 0 \) and consider
  \[
P\{T_x > s + t \mid T_x > s\} = P\{X(r) = x \text{ for } r \in [0, s + t] \mid X(r) = x \text{ for } r \in [0, s]\}
  = P\{X(r) = x \text{ for } r \in [s, s + t] \mid X(r) = x \text{ for } r \in [0, s]\}
  = P\{X(r) = x \text{ for } r \in [s, s + t] \mid X(s) = x\} \quad \text{(Markov property)}
  = P\{X(r) = x \text{ for } r \in [0, t] \mid X(0) = x\} \quad \text{(time homogeneity)}
  = P\{T_x > t\}.
\]
- Therefore, \( T_x \) satisfies the loss of memory property, and is therefore exponentially distributed.
Continuous time Markov chains

So, we have a workable model: a continuous time Markov chain is

1. An embedded discrete time Markov chain with

2. Exponential holding times.

Further, we know how to simulate both of these objects using only uniform random variables!
Continuous time Markov chains

We find the following algorithm. Let

1. the transition matrix for the embedded DTMC be $P$.
2. the holding time in state $j \in S$ be exponential with parameter $\lambda(j)$.

Algorithm for the simulation of CTMCs: Let

- $X(0)$ be determined by initial distribution (need a uniform RV)
- $t = 0$

For $j = 0, 1, 2, \ldots$

- (Simulate DTMC) Use an independent uniform to generate $X(j + 1)$ using $X(j)$th row of $P$.

- (Simulate holding time)
  - Let $T_{X(j)}$ be exponential RV with parameter $\lambda(X(j))$.
  - Set $t \leftarrow t + T_{X(j)}$.

Note: need two uniform RVs per step.
Other ways to understand CTMCs

I’ve described CTMC as

\[ \text{CTMC} = \text{DTMC} + \text{exponential holding time}. \]

1. Often, \( \lambda(x) \), the total “rate out” of \( x \), is not what is given to us.

2. Instead, for each state \( x \), we often know the “rate” from \( x \to y \), for each \( y \in S \).
   2.1 Imagine we are at state \( x \) and want to simulate forward in time.

   2.2 Suppose there are three states, \( y_1, y_2, y_3 \) and the rates to them are
      \[ \lambda(x, y_1), \quad \lambda(x, y_2), \quad \text{and} \quad \lambda(x, y_3). \]

   2.3 Then, set exponential for each and see which wins. This is the exponential
      alarm clock interpretation.

   2.4 Note that equivalent to previous alg. by noting min is exponential with rate
      \[ \lambda_0(x) = \lambda(x, y_1) + \lambda(x, y_2) + \lambda(x, y_3). \]
      and each of \( y_1, y_2, y_3 \) wins with probability
      \[ \frac{\lambda(x, y_1)}{\lambda_0(x)}, \quad \frac{\lambda(x, y_2)}{\lambda_0(x)}, \quad \text{and} \quad \frac{\lambda(x, y_3)}{\lambda_0(x)}. \]
Other ways to understand CTMCs

Example (Birth and death process)
Consider a model of a population which can only move in one of two directions: up or down by one.

We are often only given the birth and death rates for the CTMC:

\[ i \rightarrow i + 1 \text{ at a rate of } \lambda_b(i) \]
\[ i \rightarrow i - 1 \text{ at a rate of } \lambda_d(i). \]

The total rate out of \( i \) is then

\[ \lambda_0(i) = \lambda_b(i) + \lambda_d(i). \]

1. Must simulate using \( \lambda_0(i) \) to get holding times.
2. For the transition probabilities (DTMC)

\[ P\{i \rightarrow i + 1\} = \frac{\lambda_b(i)}{\lambda_b(i) + \lambda_d(i)}, \quad P\{i \rightarrow i - 1\} = \frac{\lambda_d(i)}{\lambda_b(i) + \lambda_d(i)}. \]
Recap

We have necessary understanding of

1. DTMC and simulation thereof.

2. General continuous time Markov chains with two simulation strategies:
   2.1 DTMC + exponential holding times.
   2.2 Exponential alarm clocks and waiting for winner.

- Second algorithm looks quite inefficient: need lots of uniforms/exponentials per step.

- However, could be made efficient if we could somehow reuse the exponential random variables. We’ll see how to do this later in the setting we care about.

We can finally get to what we want: biochemical/population models:

We’ll soon make special use of the structure of these models to give a new representation, and simulation strategy!
Our models of interest: Biochemical models

Biochemical models involve (all will be defined precisely later):

1. A set of “reactions” that can take place. Examples:

   \[ 2H + O \rightarrow H_2O, \quad \text{and/or} \quad \text{Gene} \rightarrow \text{Gene} + \text{mRNA}. \]

2. A time-dependent vector \( X(t) \in \mathbb{Z}_d^{\geq 0} \) giving \# of molecules at time \( t \).

3. “Rates” (intensities/propensities) at which the reactions are occurring:

   \[ \lambda_k(X(t)). \]

4. A way to tell us the total change in the state of the system due to reaction \( k \) occurring: \( \zeta_k \in \mathbb{Z}_d^{\geq 0} \).

An ODE (and integral equation) for this model would look like

\[
\dot{x}(t) = \sum_k \zeta_k \lambda_k(x(t)) \implies x(t) = x(0) + \sum_k \zeta_k \int_0^t \lambda_k(x(s))ds.
\]
Our models of interest: Biochemical models

Standard notation for biochemical reactions:

- \( S_1 + S_2 \rightarrow S_3 \)
  
is interpreted as “a molecule of \( S_1 \) combines with a molecule of \( S_2 \) to give a molecule of \( S_3 \).”

- \( \emptyset \rightarrow S_1 \)
  
is interpreted as “a molecule of \( S_1 \) arrived.” (Or maybe born.)

- \( S_2 \rightarrow \emptyset \)
  
is interpreted as “a molecule of \( S_2 \) left / decayed / died.”

- \( A \rightarrow B \)
  
is interpreted as “\( A \) was converted to \( B \).”
Biochemical models

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

  $\sum_{i=1}^{d} \nu_{ik} S_i \rightarrow \sum_{i=1}^{d} \nu'_{ik} S_i$

- $\nu_k \in \mathbb{Z}_{\geq 0}^d$: number of molecules of each chemical species **consumed** in the $k$th reaction.

- $\nu'_k \in \mathbb{Z}_{\geq 0}^d$: number of molecules of each chemical species **created** in the $k$th reaction.

Denote reaction vector as

$$\zeta_k = \nu'_k - \nu_k,$$
• Example: each instance of the reaction $S_1 + S_2 \rightarrow S_3$ changes the state of the system by the reaction vector:

$$\zeta_k = \nu'_k - \nu_k = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}.$$

• The state of the system $X(t) \in \mathbb{Z}_{\geq 0}^d$ gives the number of molecules of each species in the system at time $t$.

• So if $k$th reaction happens at time $s \geq 0$, then

$$X(s) = X(s-) + \zeta_k.$$

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

NOTE:
- $\lambda_k$ gives transition rate from $x$ to $x + \zeta_k$.
- Total rate out of $x$ is $\lambda_0(x) = \sum_k \lambda_k(x)$. 
The standard intensity function chosen is **mass-action kinetics**:

\[ \lambda_k(x) = \kappa_k \left( \prod_i \nu_{ik}! \right) \binom{x}{\nu_k} = \kappa_k \prod_i \frac{x_i!}{(x_i - \nu_{ik})!}. \]

Example: If \( S_1 \to \) anything, then \( \lambda_k(x) = \kappa_k x_1 \).

Example: If \( S_1 + S_2 \to \) anything, then \( \lambda_k(x) = \kappa_k x_1 x_2 \).

Example: If \( 2S_2 \to \) anything, then \( \lambda_k(x) = \kappa_k x_2 (x_2 - 1) \).
Some examples

Substrate-enzyme kinetics:

\[ S + E \rightleftharpoons SE \rightarrow P + E. \]

Gene transcription & translation:

\[
\begin{align*}
G &\rightarrow G + M \quad \text{transcription} \\
M &\rightarrow M + P \quad \text{translation} \\
M &\rightarrow \emptyset \quad \text{degradation} \\
P &\rightarrow \emptyset \quad \text{degradation} \\
G + P &\rightarrow \text{Bound}
\end{align*}
\]

Cartoon representation:

---

\(^2\)J. Paulsson, Physics of Life Reviews, 2, 2005 157 – 175.
Some examples

E. coli Heat Shock Response Model. 9 species, 18 reactions.

<table>
<thead>
<tr>
<th>Reaction</th>
<th>Intensity</th>
<th>Reaction</th>
<th>Intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset \rightarrow A_8$</td>
<td>$4.00 \times 10^0$</td>
<td>$A_6 + A_8 \rightarrow A_9$</td>
<td>$3.62 \times 10^{-4}X_{A_6}X_{A_8}$</td>
</tr>
<tr>
<td>$A_2 \rightarrow A_3$</td>
<td>$7.00 \times 10^{-1}X_{A_2}$</td>
<td>$A_8 \rightarrow \emptyset$</td>
<td>$9.99 \times 10^{-5}X_{A_8}$</td>
</tr>
<tr>
<td>$A_3 \rightarrow A_2$</td>
<td>$1.30 \times 10^{-1}X_{A_3}$</td>
<td>$A_9 \rightarrow A_6 + A_8$</td>
<td>$4.40 \times 10^{-5}X_{A_9}$</td>
</tr>
<tr>
<td>$\emptyset \xrightarrow{A_1} A_2$</td>
<td>$7.00 \times 10^{-3}X_{A_1}$</td>
<td>$\emptyset \rightarrow A_1$</td>
<td>$1.40 \times 10^{-5}$</td>
</tr>
<tr>
<td>stuff + $A_3 \rightarrow A_5 + A_2$</td>
<td>$6.30 \times 10^{-3}X_{A_3}$</td>
<td>$A_1 \rightarrow \emptyset$</td>
<td>$1.40 \times 10^{-6}X_{A_1}$</td>
</tr>
<tr>
<td>stuff + $A_3 \rightarrow A_4 + A_2$</td>
<td>$4.88 \times 10^{-3}X_{A_3}$</td>
<td>$A_7 \xrightarrow{A_4} A_6$</td>
<td>$1.42 \times 10^{-6}X_{A_4}X_{A_7}$</td>
</tr>
<tr>
<td>stuff + $A_3 \rightarrow A_6 + A_2$</td>
<td>$4.88 \times 10^{-3}X_{A_3}$</td>
<td>$A_5 \rightarrow \emptyset$</td>
<td>$1.80 \times 10^{-8}X_{A_5}$</td>
</tr>
<tr>
<td>$A_7 \rightarrow A_2 + A_6$</td>
<td>$4.40 \times 10^{-4}X_{A_7}$</td>
<td>$A_6 \rightarrow \emptyset$</td>
<td>$6.40 \times 10^{-10}X_{A_6}$</td>
</tr>
<tr>
<td>$A_2 + A_6 \rightarrow A_7$</td>
<td>$3.62 \times 10^{-4}X_{A_2}X_{A_6}$</td>
<td>$A_4 \rightarrow \emptyset$</td>
<td>$7.40 \times 10^{-11}X_{A_4}$</td>
</tr>
</tbody>
</table>

Lotka-Volterra predator-prey model: think of $R$ as a prey and $F$ as a predator.

$$R \xrightarrow{\kappa_1} 2R, \quad R + F \xrightarrow{\kappa_2} 2F, \quad F \xrightarrow{\kappa_3} \emptyset,$$

or

$$R \xrightarrow{\kappa_1} 2R, \quad F \xrightarrow{\kappa_2,R} 2F, \quad R + F \xrightarrow{\kappa_3} F, \quad R \xrightarrow{\kappa_4} \emptyset, \quad F \xrightarrow{\kappa_5} \emptyset.$$

---

3 Hye Won Kang, presentation at SPA in 2007.
Gillespie’s Algorithm: simulate DTMC + exponential holding times

1. Initialize. Set the initial number of molecules of each species and set \( t = 0 \).

2. Calculate the intensity/propensity function, \( \lambda_k \), for each reaction.

3. Set \( \lambda_0 = \sum_{k=1}^{M} \lambda_k \).

4. Generate two independent uniform(0,1) random numbers \( r_1 \) and \( r_2 \).

5. Set

\[
\Delta = \left( \frac{1}{\lambda_0} \right) \ln \left( \frac{1}{r_1} \right)
\]

(equivalent to drawing an exponential random variable with parameter \( \lambda_0 \)).

6. Find \( \mu \in [1, \ldots, M] \) such that

\[
\lambda_0^{-1} \sum_{k=1}^{\mu-1} \lambda_k < r_2 \leq \lambda_0^{-1} \sum_{k=1}^{\mu} \lambda_k,
\]

which is equivalent to choosing from reactions \([1, \ldots, M]\) with the \( k \)th reaction having probability \( \lambda_k / \lambda_0 \).

7. Set \( t = t + \Delta \) and update the number of each molecular species according to reaction \( \mu: X \leftarrow X + \zeta_\mu \).

8. Return to step 2 or quit.
Try coding up the Lotka-Volterra model.

\[
R \xrightarrow{10} 2R, \quad F \xrightarrow{0.01 \cdot R} 2F, \quad R + F \xrightarrow{0.01} F, \quad R \xrightarrow{0.01} \emptyset, \quad F \xrightarrow{10} \emptyset.
\]

with initial condition \([300, 300]\).
Question: is there a mathematical equation that this CTMC satisfies?

Answer: Yes, there are a few. Each needs an understanding of Poisson processes.
The Poisson process is a model for a series of random observations occurring in time.

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

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

\[
\begin{array}{ccccccccc}
& x & x & & x & x & x & x & x \\
\hline
\leftrightarrow & e_1 & \leftrightarrow & e_2 & \leftrightarrow & e_3 & \cdots & & t
\end{array}
\]

- 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.
- Define $Y_\lambda(t) \equiv Y(\lambda t)$,

Then $Y_\lambda$ is a Poisson process with parameter $\lambda$.

Intuition: The Poisson process with rate $\lambda$ is simply the number of points hit (of the unit-rate point process) when we run along the time frame at rate $\lambda$. 

\[
\begin{array}{cccccccc}
X & X & & X & X & X & X & X \\
\leftrightarrow & \leftrightarrow & \leftrightarrow & \cdots & \\
e_1 & e_2 & e_3 & \cdots & t
\end{array}
\]

$\lambda = 3$
The Poisson process

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 nonhomogeneous Poisson process with intensity $\lambda(t)$. Thus

$$P\{Y_\lambda(t + \Delta t) - Y_\lambda(t) > 0 | F_t\} = 1 - \exp\left\{-\int_t^{t+\Delta t} \lambda(s)ds\right\} \approx \lambda(t)\Delta t.$$ 

Points:

1. We have “changed time” to convert a unit-rate Poisson process to one which has rate or intensity or propensity $\lambda(t)$.

2. Will use similar time changes of unit-rate processes to build models of interest.
Consider the simple system

$$A + B \rightarrow C$$

where one molecule each of $A$ and $B$ is being converted to one of $C$.

Intuition for standard stochastic model:

$$P\{\text{reaction occurs in } (t, t + \Delta t)|\mathcal{F}_t\} \approx \kappa X_A(t)X_B(t)\Delta t$$

where

- $\kappa$ is a positive constant, the reaction rate constant.
Models of interest

\[ A + B \rightarrow C \]

Simple book-keeping: if \( X(t) = (X_A(t), X_B(t), X_C(t)) \) 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.

**Goal:** represent \( R(t) \) in terms of Poisson process.
Models of interest

Recall that for $A + B \rightarrow C$ our intuition was

$$P\{\text{reaction occurs in } (t, t + \Delta t) | \mathcal{F}_t\} \approx \kappa X_A(t)X_B(t)\Delta t,$$

and that for an inhomogeneous Poisson process with rate $\lambda(t)$ we have

$$P\{Y_\lambda(t + \Delta t) - Y_\lambda(t) > 0 | \mathcal{F}_t\} = 1 - \exp\left\{- \int_0^{\Delta t} \lambda(t)dt\right\} \approx \lambda(t)\Delta t.$$

This suggests 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 process.

Hence

$$\begin{pmatrix} X_A(t) \\ X_B(t) \\ X_C(t) \end{pmatrix} \equiv X(t) = X(0) + \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} Y \left(\int_0^t \kappa X_A(s)X_B(s)ds\right).$$

This equation uniquely determines $X$ for all $t \geq 0$.

NOTE THAT HOLDING TIMES ARE INDEED EXPONENTIAL!
General stochastic models of biochemical reactions

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

$$
\sum_{i=1}^{d} \nu_{ik} S_i \rightarrow \sum_{i=1}^{d} \nu'_{ik} S_i
$$

- $\nu_k \in \mathbb{Z}_{\geq 0}^d$: number of molecules of each chemical species consumed in the $k$th reaction.

- $\nu'_k \in \mathbb{Z}_{\geq 0}^d$: number of molecules of each chemical species created in the $k$th reaction.

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.
Population Example: Lotka-Volterra predator-prey model

Think of $A$ as a prey and $B$ as a predator.

\[
A \xrightarrow{\kappa_1} 2A, \quad A + B \xrightarrow{\kappa_2} 2B, \quad B \xrightarrow{\kappa_3} \emptyset,
\]

with $X_A(0) = X_B(0) = 1000$ and $\kappa_1 = 2$, $\kappa_2 = .002$, $\kappa_3 = 2$.

Deterministic model. Let $x(t) = [A(t), B(t)]^T$.

\[
x(t) = x(0) + \kappa_1 \int_0^t x_1(s)ds \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \kappa_2 \int_0^t x_1(s)x_2(s)ds \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \kappa_3 \int_0^t x_2(s)ds \begin{bmatrix} 0 \\ -1 \end{bmatrix}
\]

Stochastic model. Let $X(t) = [A(t), B(t)]^T$.

\[
X(t) = X(0) + Y_1 \left( \kappa_1 \int_0^t X_1(s)ds \right) \begin{bmatrix} 1 \\ 0 \end{bmatrix} + Y_2 \left( \kappa_2 \int_0^t X_1(s)X_2(s)ds \right) \begin{bmatrix} -1 \\ 1 \end{bmatrix}
+ Y_3 \left( \kappa_3 \int_0^t X_2(s)ds \right) \begin{bmatrix} 0 \\ -1 \end{bmatrix}
\]
Lotka-Volterra

Think of $A$ as a prey and $B$ as a predator.

\[ A \xrightarrow{\kappa_1} 2A, \quad A + B \xrightarrow{\kappa_2} 2B, \quad B \xrightarrow{\kappa_3} \emptyset, \]

with $A(0) = B(0) = 1000$ and $\kappa_1 = 2$, $\kappa_2 = .002$, $\kappa_3 = 2$. 

![Graphs comparing stochastic model and ODE model]
Wait! I just pulled a fast one on you.

1. I told you how to simulate a CTMC (DTMC + holding times)

2. Then developed a CTMC model for biochemical processes.

3. Is simulation of the model I developed equivalent to the simulation strategy I proposed for CTMCs? (answer: no, but this is more subtle than you might expect)

\[ X(t) = X(0) + \sum_{k} Y_k \left( \int_{0}^{t} \lambda_k(X(s))ds \right) \zeta_k, \]
Statistically exact simulation methods

\[ X(t) = X(0) + \sum_k Y_k \left( \int_0^t \lambda_k(X(s)) \, ds \right) (\nu'_k - \nu_k). \]

Alternative to the Gillespie algorithm: at time \( t \) let

\[ T_k = \int_0^t \lambda_k(X(s)) \, ds \]
\[ P_k = \min \{ s > T_k : Y_k(s) > Y_k(T_k) \}. \]

Assuming no other reactions fire first, amount of time until reaction \( k \) fires given by

\[ \lambda_k \Delta t_k = P_k - T_k \]
\[ \Rightarrow \quad \Delta t_k = \frac{P_k - T_k}{\lambda_k}. \]
The (modified) Next Reaction Method: simulating the random time change representation

1. Initialize system and set $T_k = 0$ for all $k$.

2. For each $k$, set $P_k = \ln(1/r_k)$, where $r_k \sim \text{unif}(0, 1)$.

3. For each $k$ set: $\Delta t_k = (P_k - T_k)/\lambda_k$.

4. $\Delta = \min \{\Delta t_k\}$. Reaction that fires is where min. achieved.

5. Set $P_\mu = P_\mu + \ln(1/r)$, $r \sim \text{unif}(0, 1)$.

6. For each $k$, $T_k = T_k + \lambda_k \Delta$.

7. Update system and propensities.

8. Return to step 3 or quit.

Remarks:

A. Only one random variable needed per iteration after the first: could play important role if use too many RVs.

B. Let’s code it up!
Representation of Gillespie’s algorithm

**Problem:** Can we derive a stochastic representation for Gillespie’s algorithm.

What do I need?
1. **DTMC:** uniform random variables!
   - Let \( \{U_0, U_1, U_2, \ldots \} \) be independent uniform(0,1) RVs.

2. **Exponential holding times with rates** \( \lambda_0(X(t)) \).
   - Let \( Y \) be a unit-rate Poisson process: will run at rate \( \lambda_0(X(t)) \).

Then, let

\[
R_0(t) = Y \left( \int_0^t \lambda_0(X(s)) ds \right)
\]

\[
X(t) = X(0) + \sum_k \zeta_k \int_0^t 1_{(q_{k-1}(X(s-), q_k(X(s-)))} (U_{R_0(s-)} ds) dR_0(s),
\]

where

\[
q_k(x) = \sum_{\ell=1}^k \lambda_\ell(x)
\]
Recap

We now have two representations and, hence, simulation strategies for the generation of sample paths. One is more useful than the other:

1. Random time change representation of Tom Kurtz (looks more like integrated ODE):

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

where

\[ \lambda_k(x) = \sum_{\ell=1}^k \lambda_\ell(x) \]

\[ Y_k \] are independent, unit-rate Poisson processes.

2. Gillespie's algorithm (thinning of a Poisson process):

\[ R_0(t) = Y \left( \int_0^t \lambda_0(X(s)) ds \right) \]

\[ X(t) = X(0) + \sum_k \zeta_k \int_0^t 1_{(q_{k-1}(X(s-)), q_k(X(s-))]} \left( U_{R_0(s-)} \right) dR_0(s), \]

where

Next time: Monte Carlo, gradient estimation, and variance reduction.
Simulation and Monte Carlo

Using whatever algorithm we choose, we can now generate paths of biochemical and/or population systems.

We now consider the following general question:

- given a random variable $X$ with unknown distribution function $F(x)$, how can we estimate $\mu = \mathbb{E}[X]$?

Note that for us (those studying biochemical processes), this covers many topics:

1. Straight expectations and variances:
   \[ \mathbb{E}X_i(t), \quad \mathbb{E}\left[X_i(t)^2\right], \]
   for some $i \in \{1, \ldots, d\}$ and $t \geq 0$.

2. Expected average values:
   \[ \mathbb{E}\left[\frac{1}{t} \int_0^t X_i(s)ds\right], \]

3. Any probability,
   \[ P(X(t) \in A) = \mathbb{E}[1\{X(t) \in A\}]. \]
If we can generate realizations of $X$ via a computer, the simulation approach is to estimate

$$\mu \equiv \mathbb{E}[X]$$

by running $n$ independent and identical experiments, thereby obtaining $n$ i.i.d. random variables $X_1, X_2, \ldots, X_n$, with each having the distribution $F(x)$.

Then, take the estimate as

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} X_i.$$ 

We call $\hat{\mu}_n$ an estimator.
In this case (each $X_i$ having distribution $F$), we have an *unbiased estimator* as

$$E\hat{\mu}_n = \frac{1}{n}E \sum_{i=1}^{n} X_i = \frac{1}{n} \sum_{i=1}^{n} E[X_i] = \mu.$$ 

Further, by the *strong law of large numbers* we know that

$$\hat{\mu}_n \to \mu,$$

as $n \to \infty$, with a probability of one. Thus, it is also *consistent*.

**Obvious point:** knowing that

$$\hat{\mu}_n \to \mu,$$

as $n \to \infty$, does not actually tell us how large of an $n$ we need in practice!

**This brings us to the next logical question:** how good is the estimate for a given, finite $n$. 

Simulation and Monte Carlo

To answer this question, we will apply the central limit theorem.

We know from the central limit theorem that

\[
\frac{X_1 + X_2 + \cdots + X_n - n\mu}{\sigma\sqrt{n}} \overset{D}{\approx} N(0, 1),
\]

or

\[
\frac{\sqrt{n}}{\sigma} (\hat{\mu}_n - \mu) \overset{D}{\approx} N(0, 1).
\]

Specifically, for any \( z \in \mathbb{R} \)

\[
P \left\{ -z \leq N(0, 1) \leq z \right\} \approx P \left\{ -z \leq \frac{\sqrt{n}}{\sigma} (\hat{\mu}_n - \mu) \leq z \right\}
\]

\[
= P \left\{ -\frac{\sigma z}{\sqrt{n}} \leq (\hat{\mu}_n - \mu) \leq \frac{\sigma z}{\sqrt{n}} \right\}
\]

\[
= P \left\{ \hat{\mu}_n - \frac{\sigma z}{\sqrt{n}} \leq \mu \leq \hat{\mu}_n + \frac{\sigma z}{\sqrt{n}} \right\}.
\]
Simulation and Monte Carlo

\[
P \{ -z \leq N(0,1) \leq z \} \approx P \left\{ \hat{\mu}_n - \frac{\sigma z}{\sqrt{n}} \leq \mu \leq \hat{\mu}_n + \frac{\sigma z}{\sqrt{n}} \right\}.
\]

1. The interval
\[
\left( \hat{\mu}_n - \frac{\sigma z}{\sqrt{n}}, \hat{\mu}_n + \frac{\sigma z}{\sqrt{n}} \right)
\]

is called our confidence interval and

2. the probability \( P \{ -z \leq N(0,1) \leq z \} \) is our confidence.

For example,

1. If a 90\% confidence interval is required, then \( z = 1.65 \).

2. If, on the other hand, we want a 95\% confidence interval is desired, then \( z = 1.96 \).
There is a major problem with the preceding arguments:

- If we don’t know $\mu$, we most likely do not know $\sigma$ either.

Therefore, we will also need to estimate it from our independent samples $X_1, X_2, \ldots, X_n$.

**Theorem**

Let $X_1, \ldots, X_n$ be independent and identical samples with mean $\mu$ and variance $\sigma^2$, and let

$$
\hat{\sigma}^2_n = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \hat{\mu}_n)^2,
$$

where $\hat{\mu}_n$ is the sample mean

$$
\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} X_i.
$$

Then,

$$
\mathbb{E} \sigma^2_n = \sigma^2.
$$

Therefore, we can use

$$
\sigma_n = \sqrt{\hat{\sigma}^2_n}
$$

as an estimate of the standard deviation in the confidence interval.
We have the following algorithm for producing a confidence interval for an expectation given a number of realizations.

**Algorithm for producing confidence intervals for a given n.**

1. Select \( n \), the number of experiments to be run, and \( \delta > 0 \).
2. Perform \( n \) independent replications of the experiment, obtaining the observations \( X_1, X_2, \ldots, X_n \) of the random variable \( X \).
3. Compute the sample mean and sample variance
   \[
   \hat{\mu}_n = \frac{1}{n}(X_1 + \cdots + X_n)
   \]
   \[
   \hat{\sigma}^2_n = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \hat{\mu}_n)^2.
   \]
4. Select \( z \) such that \( \Phi(z) = 1 - \delta/2 \). Then an approximate \((1 - \delta)100\%\) confidence interval for \( \mu = \mathbb{E}[X] \) is
   \[
   \left( \hat{\mu}_n - \frac{\hat{\sigma}_n z}{\sqrt{n}}, \hat{\mu}_n + \frac{\hat{\sigma}_n z}{\sqrt{n}} \right).
   \]
So, the confidence interval,

\[
\left( \hat{\mu}_n - \frac{\hat{\sigma}_n z}{\sqrt{n}}, \hat{\mu}_n + \frac{\hat{\sigma}_n z}{\sqrt{n}} \right).
\]

which is what you report, is of size

\[
O \left( \frac{\sigma}{\sqrt{n}} \right).
\]

This is terrible news. To get a decrease of size 10 you must

- Increase \( n \) by a factor of 100. (!)

- Or decrease \( \sigma \).... This is the goal of variance reduction techniques. (more later)
Computer exercise: Lotka-Volterra model.

\[ R \overset{10}{\rightarrow} 2R, \quad F \overset{0.01}{\rightarrow} R \rightarrow 2F, \quad R + F \overset{0.01}{\rightarrow} F, \quad R \overset{0.01}{\rightarrow} \emptyset, \quad F \overset{10}{\rightarrow} \emptyset. \]

with initial condition \([300, 300]\).

Using your codes from yesterday,

- estimate the expectation of the number of rabbits at time \(T = 0.2\).
- Give a 95% confidence interval based upon 1,000 sample paths.

- Try estimating to later times: say time \(T = 1\).
A specific problem of interest: Gradient estimation

We have

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

and we define

\[ J(\theta) = \mathbb{E}f(X^\theta(t)]. \]

We know how to estimate \( J(\theta) \) using Monte Carlo.

However, what if we want

\[ J'(\theta) = \frac{d}{d\theta} \mathbb{E}f(X^\theta(t)). \]

Thus, we want to know how sensitive our statistic is to perturbations in \( \theta \).

Tells us:

1. How best to control a system for desired outcome.
2. Robustness of system to perturbations in parameters.
3. Which parameters we need to estimate well from data, etc.

There are multiple methods. We will consider:

1. Finite differences.
2. Likelihood transform method.
Finite differencing

This method is pretty straightforward and is therefore used most.

Simply note that

$$J'(\theta) = \frac{d}{d\theta} \mathbb{E} f(X^{\theta}(t)) = \frac{\mathbb{E} f(X^{\theta+\epsilon}(t)) - \mathbb{E} f(X^{\theta}(t))}{\epsilon} + O(\epsilon).$$

Centered differencing reduces bias to $O(\epsilon^2)$.

The usual finite difference estimator is

$$D_R(\epsilon) = \frac{1}{R} \sum_{i=1}^{R} \frac{f(X^{\theta+\epsilon}_i(t)) - f(X^{\theta}_i(t))}{\epsilon}$$

If generated independently, then

$$\text{Var}(D_R(\epsilon)) = R^{-1} \epsilon^{-2} \text{Var}(f(X^{\theta+\epsilon}_i(t)) - f(X^{\theta}_i(t)))$$

$$= O(R^{-1} \epsilon^{-2}).$$

Terrible. Worse than expectations. Let's explore behavior on example.

How about common random numbers for variance reduction?
Common random numbers

It’s exactly what it sounds like. Reuse the random numbers used in the generation of

\[ X_{[i]}^{\theta + \epsilon}(t) \quad \text{and} \quad X_{[i]}^\theta(t). \]

Why? Because:

\[
\text{Var}(f(X_{[i]}^{\theta + \epsilon}(t)) - f(X_{[i]}^{\theta + \epsilon}(t))) = \text{Var}(f(X_{[i]}^{\theta + \epsilon}(t))) + \text{Var}(f(X_{[i]}^\theta(t)))
- 2\text{Cov}(f(X_{[i]}^{\theta + \epsilon}(t)), f(X_{[i]}^\theta(t))).
\]

So, if we can “couple” the random variables, we can get a variance reduction! Sometimes substantial.

Example

Let \( U_1 \sim \text{unif}(0, 1) \) and let \( U_2 = 1 - U_1 \). Then,

\[ U_1 + U_2 = 1 \implies \text{Var}(U_1 + U_2) = 0. \]

If generated independently,

\[ \text{Var}(U_1 + U_2) = \text{Var}(U_1) + \text{Var}(U_2) = \frac{1}{6}. \]
Common random numbers

- In the context of Gillespie’s algorithm, we simply reuse all the same random numbers (uniforms).

- This can be achieved simply by setting the “seed” of the random number generator before generating $X^{\theta+\epsilon}$ and $X^\theta$.

Mathematically, this is equivalent to using the same Poisson process and uniforms in:

$$R^{\theta+\epsilon}_0(t) = Y \left( \int_0^t \lambda_0^{\theta+\epsilon}(X^{\theta+\epsilon}(s))ds \right)$$

$$X^{\theta+\epsilon}(t) = X^{\theta+\epsilon}(0) + \sum_k \zeta_k \int_0^t 1_{(q_{k-1}^{\theta+\epsilon}(X^{\theta+\epsilon}(s-)), q_k^{\theta+\epsilon}(X^{\theta+\epsilon}(s-)))}(UR_0(s-))dR^{\theta+\epsilon}_0(s),$$

$$R^\theta_0(t) = Y \left( \int_0^t \lambda_0^{\theta}(X^{\theta}(s))ds \right)$$

$$X^\theta(t) = X^\theta(0) + \sum_k \zeta_k \int_0^t 1_{(q_{k-1}^\theta(X^\theta(s-)), q_k^\theta(X^\theta(s-)))}(UR_0(s-))dR^\theta_0(s),$$
Common random numbers

CRN + Gillespie is good idea.

1. Costs little in terms of implementation.
2. Variance reduction can be huge.

Thus, it is probably the most common method used today.

But:
- Over time, the processes decouple, often completely.

**Computer exercise:** Let’s explore behavior on example.

Can we do better?
Using common random numbers in previous fashion is a way of “coupling” the processes together.

Is there a natural way to couple processes using random time change? Can we couple the Poisson processes?

Answer: yes. Multiple ways. I will show one which works very well.
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

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

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:

$$\text{Var}(Z_{13.1}(t) - Z_{13}(t)) = \text{Var}(Y_2(0.1t)) = 0.1t.$$ 

Using a fact: sum of homogeneous Poisson process is again a Poisson process.
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 $g(t)$.

We can let $Y_1$, $Y_2$, and $Y_3$ be independent, unit-rate Poisson processes and define

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

$$Z_g(t) = Y_1 \left( \int_0^t f(s) \wedge g(s) \, ds \right) + Y_3 \left( \int_0^t g(s) - (f(s) \wedge g(s)) \, ds \right),$$

where we are using that, for example,

$$Y_1 \left( \int_0^t f(s) \wedge g(s) \, ds \right) + Y_2 \left( \int_0^t f(s) - (f(s) \wedge g(s)) \, ds \right) = Y \left( \int_0^t f(s) \, ds \right),$$

where $Y$ is a unit rate Poisson process.
Next problem: parameter sensitivities.

Couple the processes.

\[ X^{\theta+\epsilon}(t) = X^{\theta+\epsilon}(0) + \sum_k Y_{k,1} \left( \int_0^t \lambda_k^{\theta+\epsilon}(X^{\theta+\epsilon}(s)) \wedge \lambda_k^{\theta}(X^{\theta}(s)) ds \right) \zeta_k \]

\[ + \sum_k Y_{k,2} \left( \int_0^t \lambda_k^{\theta+\epsilon}(X^{\theta+\epsilon}(s)) - \lambda_k^{\theta+\epsilon}(X^{\theta+\epsilon}(s)) \wedge \lambda_k^{\theta}(X^{\theta}(s)) ds \right) \zeta_k \]

\[ X^{\theta}(t) = X^{\theta}(0) + \sum_k Y_{k,1} \left( \int_0^t \lambda_k^{\theta+\epsilon}(X^{\theta+\epsilon}(s)) \wedge \lambda_k^{\theta}(X^{\theta}(s)) ds \right) \zeta_k \]

\[ + \sum_k Y_{k,3} \left( \int_0^t \lambda_k^{\theta}(X^{\theta}(s)) - \lambda_k^{\theta+\epsilon}(X^{\theta+\epsilon}(s)) \wedge \lambda_k^{\theta}(X^{\theta}(s)) ds \right) \zeta_k, \]
Next problem: parameter sensitivities.

Theorem
Suppose \((X^{\theta+\epsilon}, X^\theta)\) satisfy coupling. Then, for any \(T > 0\) there is a \(C_{T,f} > 0\) for which
\[
\mathbb{E} \sup_{t \leq T} \left( f(X^{\theta+\epsilon}(t)) - f(X^\theta(t)) \right)^2 \leq C_{T,f} \epsilon.
\]

This lowers variance of estimator from
\[
O(R^{-1} \epsilon^{-2}),
\]
to
\[
O(R^{-1} \epsilon^{-1}).
\]

Lowered by order of magnitude (in \(\epsilon\)).
Parameter Sensitivities

\[ G \xrightarrow{2} G + M, \]
\[ M \xrightarrow{10} M + P, \]
\[ M \xrightarrow{k} \emptyset, \]
\[ P \xrightarrow{1} \emptyset. \]

Want
\[ \frac{\partial}{\partial k} \mathbb{E} \left[ X^k_{\text{protein}}(30) \right], \quad k \approx 1/4. \]

<table>
<thead>
<tr>
<th>Method</th>
<th>( R )</th>
<th>95% CI</th>
<th># updates</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood</td>
<td>689,600</td>
<td>-312.1 ± 6.0</td>
<td>2.9 \times 10^9</td>
<td>3,506.6 S</td>
</tr>
<tr>
<td>CMC</td>
<td>246,000</td>
<td>-319.3 ± 6.0</td>
<td>2.1 \times 10^9</td>
<td>2,364.8 S</td>
</tr>
<tr>
<td>CRP/CRN</td>
<td>25,980</td>
<td>-316.7 ± 6.0</td>
<td>2.2 \times 10^8</td>
<td>270.9 S</td>
</tr>
<tr>
<td>CFD</td>
<td>4,580</td>
<td>-319.9 ± 6.0</td>
<td>2.0 \times 10^7</td>
<td>29.2 S</td>
</tr>
</tbody>
</table>

Table: Each finite difference method used \( \epsilon = 1/40 \). The exact value is \( J(1/4) = -318.073 \).

Computer exercise: Explore the behavior on our example.
Finite differencing conclusions

There are benefits and drawbacks to finite differencing:

Major benefit: very low variance if coupling is performed.

Small drawback: there is a bias of order $O(\epsilon^2)$ since

$$\frac{\mathbb{E}f(X^{\theta+\epsilon/2}(t)) - \mathbb{E}f(X^{\theta-\epsilon/2}(t))}{\epsilon} = O(\epsilon^2).$$

There is another method, likelihood transforms, that gets rid of bias, but at cost of (much) higher variance.
Likelihood transforms

Basic idea: suppose we have a random variable $X$ with density $\rho_X^\theta$, that depends upon $\theta$. Then,

$$\mathbb{E}X^\theta = \int x \rho_X^\theta(x) dx.$$

You would then compute the derivative via

$$\frac{d}{d\theta} \mathbb{E}X^\theta = \lim_{\epsilon \to 0} \frac{\mathbb{E}X^{\theta+\epsilon} - \mathbb{E}X^\theta}{\epsilon} = \int x \lim_{\epsilon \to 0} \frac{\rho_X^{\theta+\epsilon}(x) - \rho_X^\theta(x)}{\epsilon} dx = \int x \left[ \frac{d}{d\theta} \rho_X^\theta(x) \right] dx,$$

so long as the limit can be passed through the integral (dominated convergence theorem).
Likelihood transforms

The same basic idea can work for us. If we think of \( \omega \) as a specific realization of our process:

1. Jump times.
2. Jump positions.

and if we let \( \rho^\theta(\omega) \) be the density of our path, then we can formally write

\[
\mathbb{E}f(X^\theta(T)) = \int f(\omega)\rho^\theta(\omega)d\omega
\]

and try to compute

\[
\frac{d}{d\theta} \mathbb{E}f(X^\theta(T)) = \int f(\omega)\frac{d}{d\theta}\rho^\theta(\omega)d\omega
\]

\[
= \int f(\omega) \left[ \frac{d}{d\theta} \ln(\rho^\theta(\omega)) \right] \rho^\theta(\omega)d\omega
\]

\[
= \mathbb{E}g^\theta(X^\theta(t)),
\]

where

\[
g^\theta(X^\theta(t)) = f(X^\theta(t)) \left[ \frac{d}{d\theta} \ln(\rho^\theta(X^\theta(t))) \right]
\]
Likelihood transforms

Obvious question: what is density of a path $\rho(X(T))$? This has been worked out. Consider process run up to some jump time (usually first jump past terminal time)

$$\rho(X(t)) = \prod_{i \in \text{jumps}} \frac{\lambda_{k_i}(X(t_i-))}{\lambda_0(X(t_i-))} \times \lambda_0(X(t_i-))e^{-\lambda_0(X(t_i-))\Delta_i}$$

$$= \prod_{i \in \text{jumps}} \lambda_{k_i}(X(t_i-))e^{-\lambda_0(X(t_i-))\Delta_i}$$

where

1. $i$ enumerates over the jumps,
2. $k_i$ is reaction channel of $i$th jump.
3. $\Delta_i$ gives how long we held before jumping.

Now we just need to take logs and differentiate assuming $\theta$ is some rate constant.
Likelihood transforms

Assuming mass action kinetics and:

\[ \lambda_1(x) = \theta \Lambda_1(x), \]

we have

\[ \rho^\theta(X^\theta(t)) = \prod_{i \in \text{jumps}} \lambda_{ki}^\theta(X^\theta(t_i-)) e^{-\lambda_0^\theta(X^\theta(t_i-)) \Delta_i} \]

and so

\[ \ln \rho^\theta(X^\theta(t)) = \sum_{i \in \text{jumps}} \left[ \ln(\lambda_{ki}^\theta(X^\theta(t_i-))) - \lambda_0^\theta(X^\theta(t_i-)) \Delta_i \right]. \]

Differentiating is now immediate:

\[
\frac{d}{d\theta} \ln \rho^\theta(X^\theta(t)) = \sum_{i \in \text{jumps}} \left[ (1 \{k_i=1\} \frac{1}{\theta}) - \Lambda_1(X^\theta(t_i-)) \Delta_i \right]
= \sum_{i \in \text{jumps}} \frac{1}{\theta} \left( 1 \{k_i=1\} - \lambda_1(X^\theta(t_i-)) \Delta_i \right)
\]
This all leads to easy algorithm: we want

\[
\frac{d}{d\theta} \mathbb{E} f(X^\theta(T)) = \int f(\omega) \frac{d}{d\theta} \rho^\theta(\omega) d\omega
\]

\[
= \int f(\omega) \left[ \frac{d}{d\theta} \ln(\rho^\theta(\omega)) \right] \rho^\theta(\omega) d\omega,
\]

we have

\[
\frac{d}{d\theta} \ln \rho^\theta(X^\theta(t)) = \sum_{i \in \text{jumps}} \frac{1}{\theta} (1_{\{k_i=1\}} - \lambda_1(X^\theta(t_i^-)) \Delta_i) \equiv M^\theta(t),
\]

So simply perform usual exact simulation and use

\[
\frac{1}{n} \sum_{i=1}^{n} f(X^\theta(t)) M^\theta(t).
\]

This is likelihood ratio method: unbiased, but often huge variance.
Consider the model of gene transcription and translation

\[ G \overset{2}{\rightarrow} G + M, \quad M \overset{10}{\rightarrow} M + P, \quad M \overset{\theta}{\rightarrow} \emptyset, \quad P \overset{1}{\rightarrow} \emptyset, \tag{3} \]

where a single gene is being translated into mRNA, which is then being transcribed into proteins. We suppose \( \theta \approx 1/4 \).

Assuming that there is a single gene copy, the stochastic equation is

\[
X^\theta(t) = X^\theta(0) + Y_1(2t) \begin{pmatrix} 1 \\ 0 \end{pmatrix} + Y_2 \left( \int_0^t 10X^\theta_1(s)ds \right) \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\
+ Y_3 \left( \int_0^t \theta X^\theta_1(s)ds \right) \begin{pmatrix} -1 \\ 0 \end{pmatrix} + Y_4 \left( \int_0^t X^\theta_2(s)ds \right) \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \tag{4} \]

Defining

\[
J(\theta_0, t) \overset{\text{def}}{=} \frac{d}{d\theta} \mathbb{E} \left[ X^\theta_2(t) \right] \bigg|_{\theta=\theta_0},
\]

our goal is to efficiently estimate \( J(1/4, t) \).
Approximations

<table>
<thead>
<tr>
<th>Method</th>
<th>$R$</th>
<th>95% CI</th>
<th># updates</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood</td>
<td>689,600</td>
<td>-312.1 ± 6.0</td>
<td>$2.9 \times 10^9$</td>
<td>3,506.6 S</td>
</tr>
<tr>
<td>CMC</td>
<td>246,000</td>
<td>-319.3 ± 6.0</td>
<td>$2.1 \times 10^9$</td>
<td>2,364.8 S</td>
</tr>
<tr>
<td>CRP</td>
<td>25,980</td>
<td>-316.7 ± 6.0</td>
<td>$2.2 \times 10^8$</td>
<td>270.9 S</td>
</tr>
<tr>
<td>CFD</td>
<td>4,580</td>
<td>-319.9 ± 6.0</td>
<td>$2.0 \times 10^7$</td>
<td>29.2 S</td>
</tr>
</tbody>
</table>

**Table:** Each finite difference method used $\epsilon = 1/40$. The exact value is $J(1/4) = -318.073$. 
Comparison from 5,000 samples each with $\epsilon = 1/40$
Recap

I hope you’ve learned something. In particular:

1. How to simulate CTMCs for biochemical processes:
   - Gillespie’s algorithm.
   - Next reaction method (RTC representation).

   2.1 Different FD methods (CMC, CRN, Coupled Finite Differences).
   2.2 Likelihood ratio transform.
This is just the beginning. You’ve seen that even simply systems (LV) can take a very long time to simulate if you need thousands of paths. What can you do?

1. Use an Euler approximation: tau-leaping.

2. Use LLN to derive ODE model (Kurtz 1972).

3. Use CLT to derive a diffusion approximation (SDE model) to system (Kurtz 1978).
That’s it.

Thanks for listening and thanks to the MBI and organizers for doing all of this.