Engineering Genetic Circuits

Chris J. Myers

Lecture 8: SSA Variations

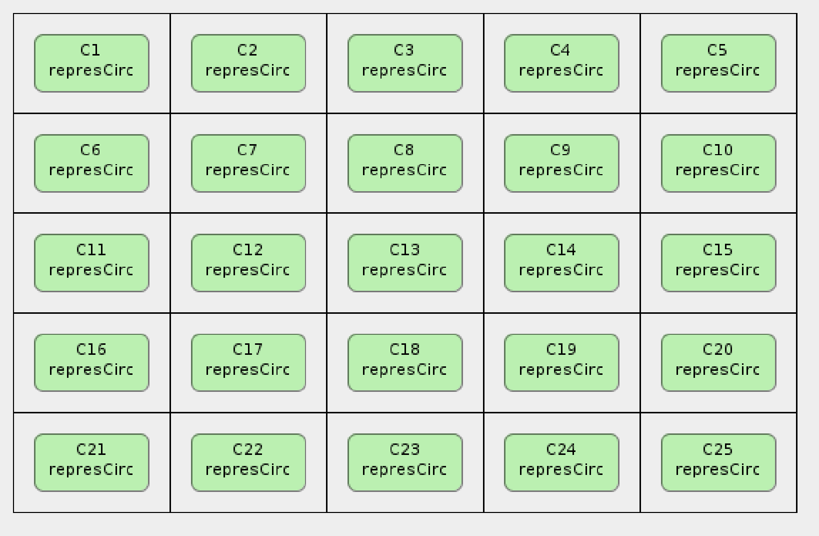
Outline

- *Hierarchical SSA* (hSSA)
- *Weighted SSA* (wSSA)
- *Incremental SSA* (iSSA)

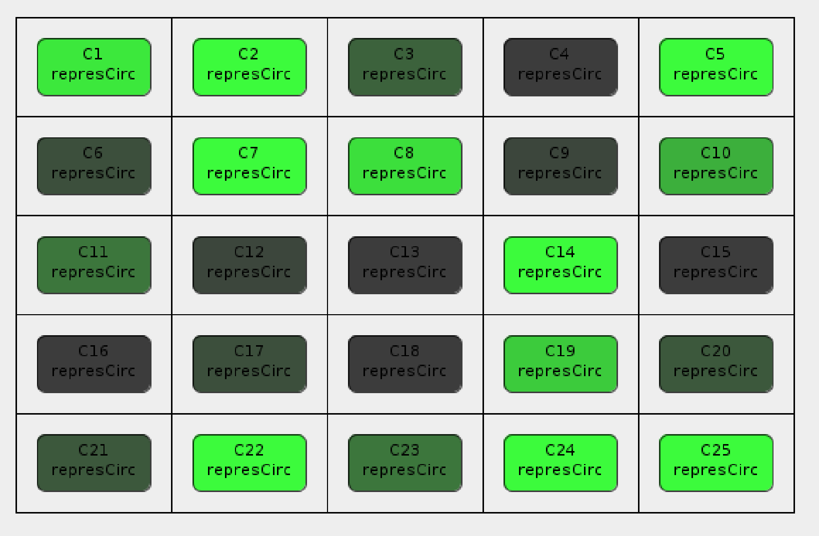
Cellular Population Models

- Genetic circuits have been constructed for many applications, such as genetic timers, oscillators, and logic gates, among others.
- These applications are usually analyzed in a single cell.
- However, there are applications in which population modeling is a necessity, such as biomedical applications.

Population-based Models within iBioSim



Visualization of Population-based Models



Hierarchical Model Composition Package

- The hierarchy in grid models is represented using SBML's *hierarchical model composition package*.
- Allows top-level models to be constructed from a collection of sub-models.
- *Replacements* and *deletions* customizes connection of sub-models.

Problems with Hierarchy

- Dealing with hierarchy can be difficult.
- Many modeling tools flatten (inline) the hierarchy of a model before simulation.
- Flattening causes the size of the model representation to grow quickly.
- The flattening process can be very time consuming.

Comparison of Flattening to Simulation Runtime

Num. Components	Flattening (sec)	Simulation (sec)
4	1.101	0.488
16	9.482	2.458
36	40.065	8.408
64	119.619	24.272
100	285.714	62.709

Hierarchical Stochastic Simulation Algorithm (hSSA)

- These results motivated the development of the hSSA.
- The hierarchical simulator avoids the cost of flattening while preserving identical simulation results.

Algorithm 1: Hierarchical SSA

Input: Hierarchical reaction model, $M = \langle M_0, \ldots, M_p \rangle$;
Output: Time series simulation, α;
$\alpha := \langle\rangle$;
$\langle t, \mathbf{x} \rangle := \mathit{initialize}(M)$;
repeat
    $\alpha := \alpha \cdot \langle t, \mathbf{x} \rangle$;
    $\langle \mathbf{a}, a_0 \rangle := \mathit{computePropensities}(M, \mathbf{x})$;
    $\tau := \mathit{computeNextReactionTime}(a_0)$;
    $\langle \nu, \mu \rangle := \mathit{selectNextReaction}(\mathbf{a}, a_0)$;
    $\langle t, \mathbf{x} \rangle := \mathit{updateState}(M, t, \tau, \mathbf{x}, \nu, \mu)$;
until $t > \mathit{timeLimit}$;

hSSA Algorithm

Algorithm 2: Hierarchical SSA

Input: Hierarchical reaction model, $M = \langle M_0, \ldots, M_p \rangle$;

Output: Time series simulation, α;

$\alpha := \langle \rangle$;

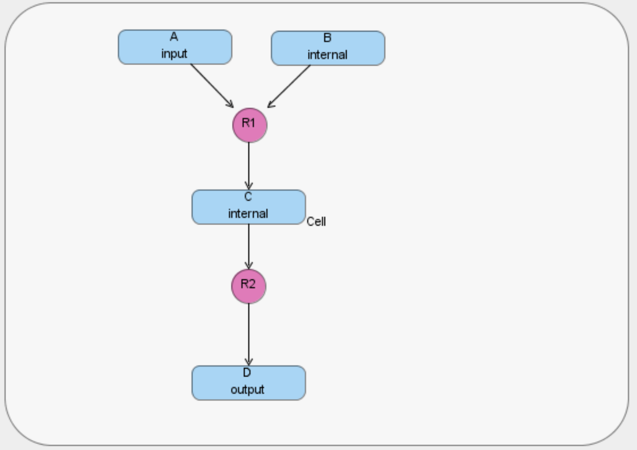
$\langle t, \mathbf{x} \rangle := \mathit{initialize}(M)$;

repeat

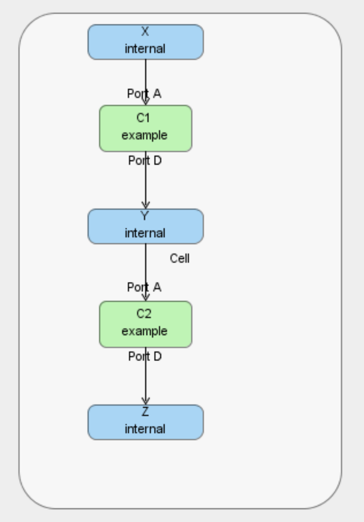
- $\alpha := \alpha \cdot \langle t, \mathbf{x} \rangle$;
- $\langle \mathbf{a}, a_0 \rangle := \mathit{computePropensities}(M, \mathbf{x})$;
- $\tau := \mathit{computeNextReactionTime}(a_0)$;
- $\langle \nu, \mu \rangle := \mathit{selectNextReaction}(\mathbf{a}, a_0)$;
- $\langle t, \mathbf{x} \rangle := \mathit{updateState}(M, t, \tau, \mathbf{x}, \nu, \mu)$;

until $t > \mathit{timeLimit}$;

Example



Example



Reaction R2 is deleted in C2

$\alpha := \langle\rangle$

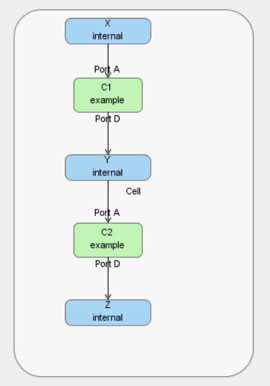
C1				
t	A	B	C	D

Top			
t	X	Y	Z

C2				
t	A	B	C	D

Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0

τ	υ	μ

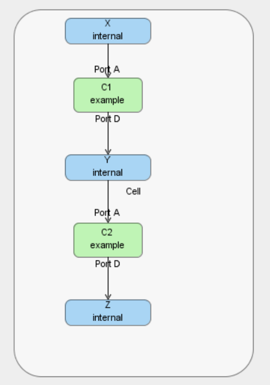


$\langle t, \mathbf{x} \rangle := initialize(M)$

C1				
t	A	B	C	D
0	10	10	0	0

Top			
t	X	Y	Z
0	5	10	10

C2				
t	A	B	C	D
0	10	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0

τ	υ	μ

$\langle t, \mathbf{x} \rangle := initialize(M)$

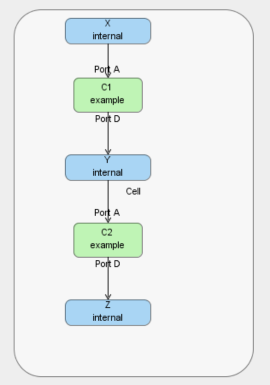
C1				
t	A	B	C	D
0	10	10	0	0

Top			
t	X	Y	Z
0	5	10	10

C2				
t	A	B	C	D
0	10	10	0	0

Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0

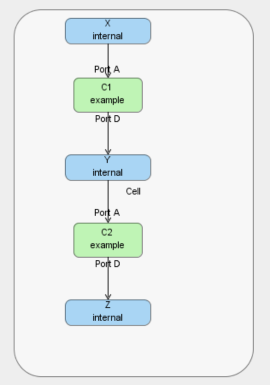
τ	υ	μ



C1				
t	A	B	C	D
0	10	10	0	0

Top			
t	X	Y	Z
0	5	0	10

C2				
t	A	B	C	D
0	10	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0

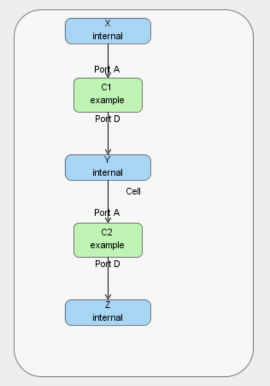
τ	υ	μ

$\langle t, \mathbf{x} \rangle := initialize(M)$

C1				
t	A	B	C	D
0	10	10	0	0

Top			
t	X	Y	Z
0	5	0	10

C2				
t	A	B	C	D
0	10	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0

τ	υ	μ

$\langle t, \mathbf{x} \rangle := initialize(M)$

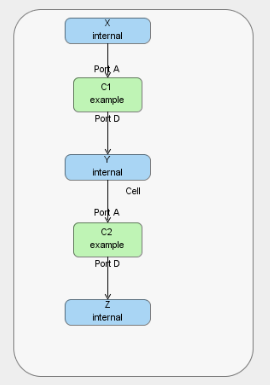
C1				
t	A	B	C	D
0	10	10	0	0

Top			
t	X	Y	Z
0	5	0	0

C2				
t	A	B	C	D
0	10	10	0	0

Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0

τ	υ	μ

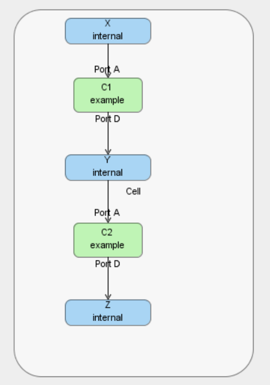


$\langle t, \mathbf{x} \rangle := initialize(M)$

C1				
t	A	B	C	D
0	10	10	0	0

Top			
t	X	Y	Z
0	5	0	0

C2				
t	A	B	C	D
0	10	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0

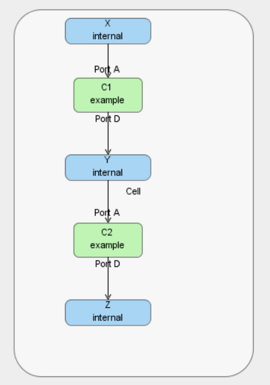
τ	υ	μ

$\langle t, \mathbf{x} \rangle := initialize(M)$

C1				
t	A	B	C	D
0	5	10	0	0

Top			
t	X	Y	Z
0	5	0	0

C2				
t	A	B	C	D
0	10	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0

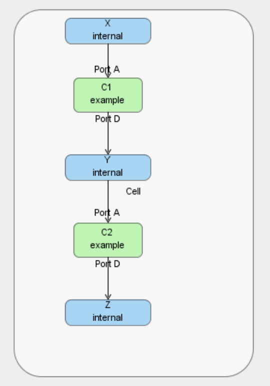
τ	υ	μ

$\langle t, \mathbf{x} \rangle := initialize(M)$

C1				
t	A	B	C	D
0	5	10	0	0

Top			
t	X	Y	Z
0	5	0	0

C2				
t	A	B	C	D
0	10	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0

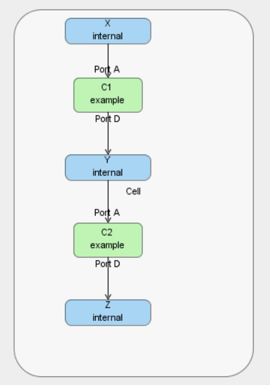
τ	υ	μ

$\langle t, \mathbf{x} \rangle := initialize(M)$

C1				
t	A	B	C	D
0	5	10	0	0

Top			
t	X	Y	Z
0	5	0	0

C2				
t	A	B	C	D
0	0	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0

τ	υ	μ

$\alpha := \alpha \cdot \langle t, \mathbf{x} \rangle$

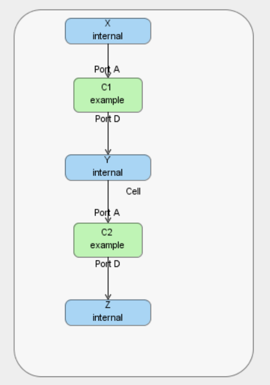
C1				
t	A	B	C	D
0	5	10	0	0

Top			
t	X	Y	Z
0	5	0	0

C2				
t	A	B	C	D
0	0	10	0	0

Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0

τ	υ	μ

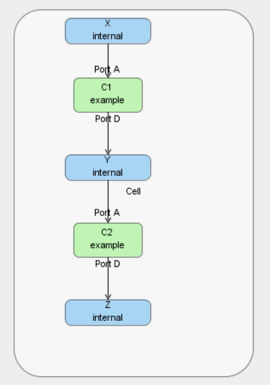


$\langle \mathbf{a}, a_0 \rangle := computePropensities(M, \mathbf{x})$

C1				
t	A	B	C	D
0	5	10	0	0

Top			
t	X	Y	Z
0	5	0	0

C2				
t	A	B	C	D
0	0	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
5	0	0	0	5

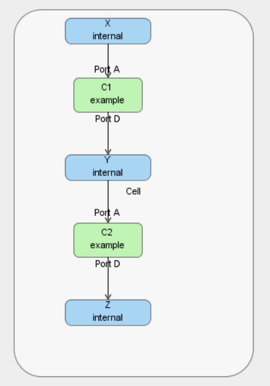
τ	υ	μ

$\tau := computeNextReactionTime(a_0)$

C1				
t	A	B	C	D
0	5	10	0	0

Top			
t	X	Y	Z
0	5	0	0

C2				
t	A	B	C	D
0	0	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
5	0	0	0	5

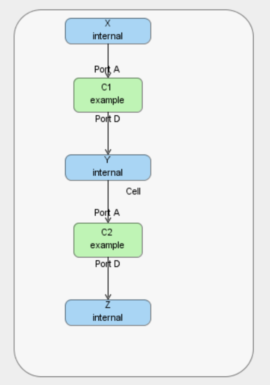
τ	υ	μ
0.1		

$\langle v, \mu \rangle := selectNextReaction(\mathbf{a}, a_0)$

C1				
t	A	B	C	D
0	5	10	0	0

Top			
t	X	Y	Z
0	5	0	0

C2				
t	A	B	C	D
0	0	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
5	0	0	0	5

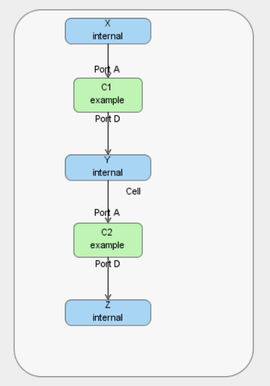
τ	v	μ
0.1	C1	R1

$\langle t, \mathbf{x} \rangle := \mathit{updateState}(M, t, \tau, \mathbf{x}, \nu, \mu)$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0

Top			
t	X	Y	Z
0	5	0	0
0.1	5	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
5	0	0	0	5

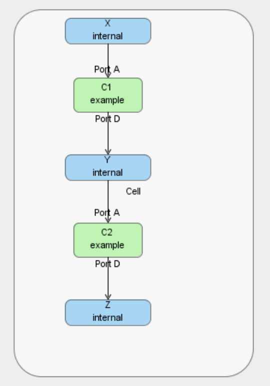
τ	υ	μ
0.1	C1	R1

$\langle t, \mathbf{x} \rangle := updateState(M, t, \tau, \mathbf{x}, \nu, \mu)$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
5	0	0	0	5

τ	υ	μ
0.1	C1	R1

$\alpha := \alpha \cdot \langle t, \mathbf{x} \rangle$

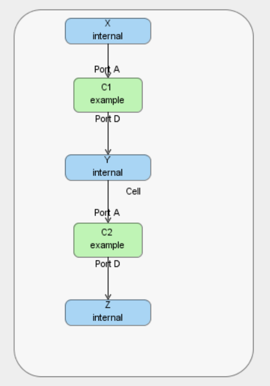
C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0

Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
5	0	0	0	5

τ	υ	μ
0.1	C1	R1

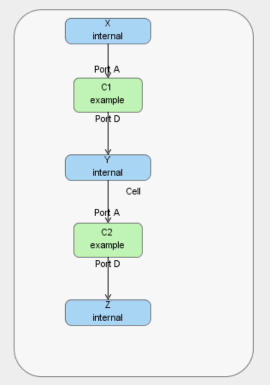


$\langle \mathbf{a}, a_0 \rangle := computePropensities(M, \mathbf{x})$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	1	0	0	4.6

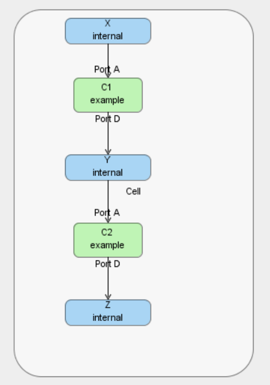
τ	υ	μ
0.1	C1	R1

$\tau := computeNextReactionTime(a_0)$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	1	0	0	4.6

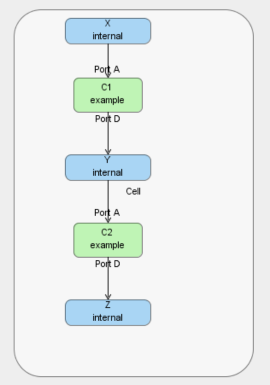
τ	υ	μ
0.2	C1	R1

$\langle \nu, \mu \rangle := selectNextReaction(\mathbf{a}, a_0)$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	1	0	0	4.6

τ	υ	μ
0.2	C1	R2

$\langle t, \mathbf{x} \rangle := updateState(M, t, \tau, \mathbf{x}, \nu, \mu)$

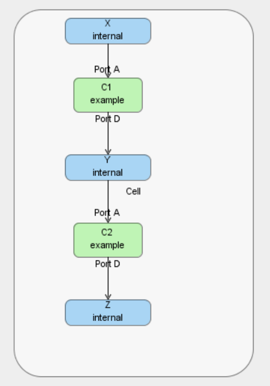
C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	0	10	0	0

Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	1	0	0	4.6

τ	υ	μ
0.2	C1	R2



$\langle t, \mathbf{x} \rangle := \mathit{updateState}(M, t, \tau, \mathbf{x}, \nu, \mu)$

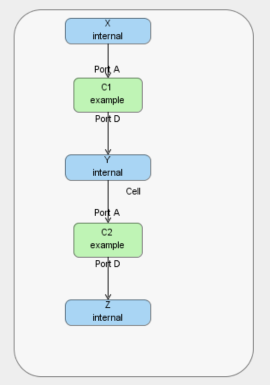
C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	0	10	0	0

Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	1	0	0	4.6

τ	υ	μ
0.2	C1	R2

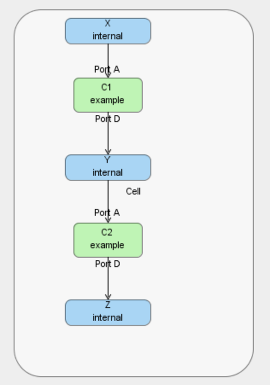


$\langle t, \mathbf{x} \rangle := updateState(M, t, \tau, \mathbf{x}, \nu, \mu)$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	1	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	1	0	0	4.6

τ	υ	μ
0.2	C1	R2

$\alpha := \alpha \cdot \langle t, \mathbf{x} \rangle$

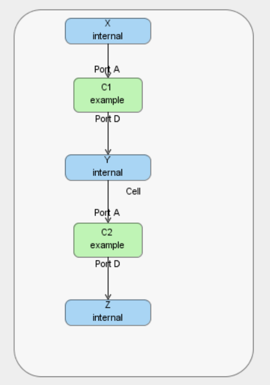
C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	1	10	0	0

Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	1	0	0	4.6

τ	υ	μ
0.2	C1	R2



$\langle \mathbf{a}, a_0 \rangle := computePropensities(M, \mathbf{x})$

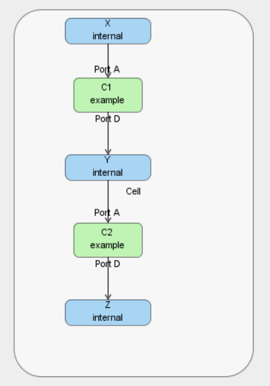
C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	1	10	0	0

Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	0	1	0	4.6

τ	υ	μ
0.2	C1	R2

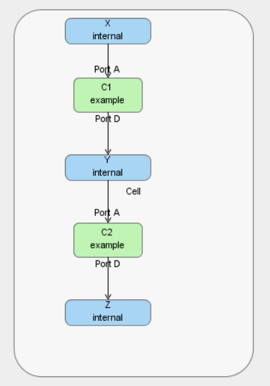


$\tau := computeNextReactionTime(a_0)$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	1	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	0	1	0	4.6

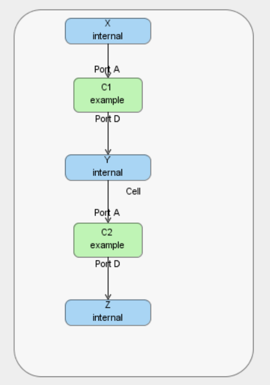
τ	υ	μ
0.2	C1	R2

$\langle v, \mu \rangle := selectNextReaction(\mathbf{a}, a_0)$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	1	10	0	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	0	1	0	4.6

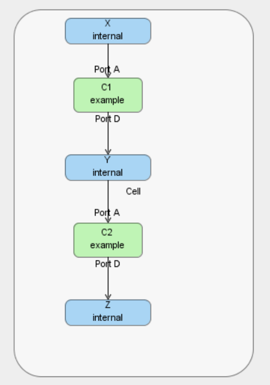
τ	υ	μ
0.2	C2	R1

$\langle t, \mathbf{x} \rangle := \mathit{updateState}(M, t, \tau, \mathbf{x}, \nu, \mu)$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1
0.5	4	9	0	1

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0
0.5	4	1	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	1	10	0	0
0.5	0	9	1	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	0	1	0	4.6

τ	υ	μ
0.2	C2	R1

$\langle t, \mathbf{x} \rangle := \mathit{updateState}(M, t, \tau, \mathbf{x}, \nu, \mu)$

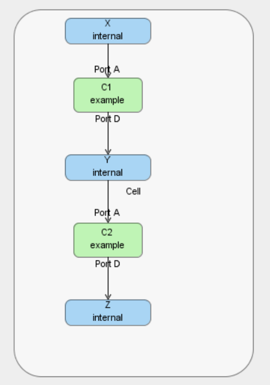
C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1
0.5	4	9	0	1

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0
0.5	4	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	1	10	0	0
0.5	0	9	1	0

Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	0	1	0	4.6

τ	υ	μ
0.2	C2	R1



$\langle t, \mathbf{x} \rangle := updateState(M, t, \tau, \mathbf{x}, \nu, \mu)$

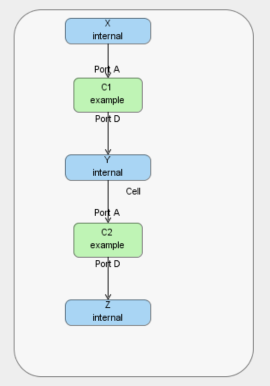
C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1
0.5	4	9	0	0

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0
0.5	4	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	1	10	0	0
0.5	0	9	1	0

Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	0	1	0	4.6

τ	υ	μ
0.2	C2	R1

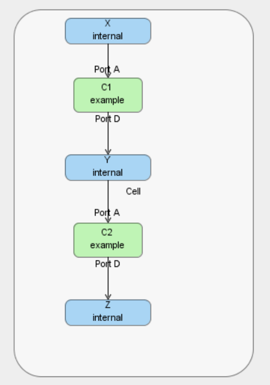


$\alpha := \alpha \cdot \langle t, \mathbf{x} \rangle$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1
0.5	4	9	0	0

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0
0.5	4	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	1	10	0	0
0.5	0	9	1	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	0	1	0	4.6

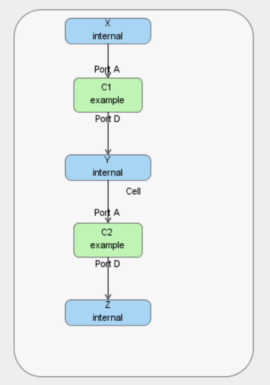
τ	υ	μ
0.2	C2	R1

$\langle \mathbf{a}, a_0 \rangle := computePropensities(M, \mathbf{x})$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1
0.5	4	9	0	0

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0
0.5	4	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	1	10	0	0
0.5	0	9	1	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	0	0	0	3.6

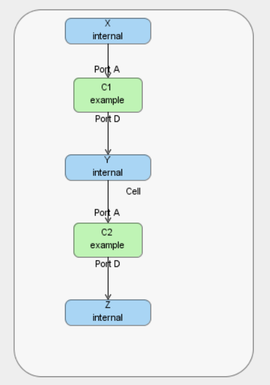
τ	υ	μ
0.2	C2	R1

$\langle \mathbf{a}, a_0 \rangle := computePropensities(M, \mathbf{x})$

C1				
t	A	B	C	D
0	5	10	0	0
0.1	4	9	1	0
0.3	4	9	0	1
0.5	4	9	0	0

Top			
t	X	Y	Z
0	5	0	0
0.1	4	0	0
0.3	4	1	0
0.5	4	0	0

C2				
t	A	B	C	D
0	0	10	0	0
0.1	0	10	0	0
0.3	1	10	0	0
0.5	0	9	1	0



Propensities				
C1		C2		Total
a_1	a_2	a_1	a_2	a_0
3.6	0	0	0	3.6

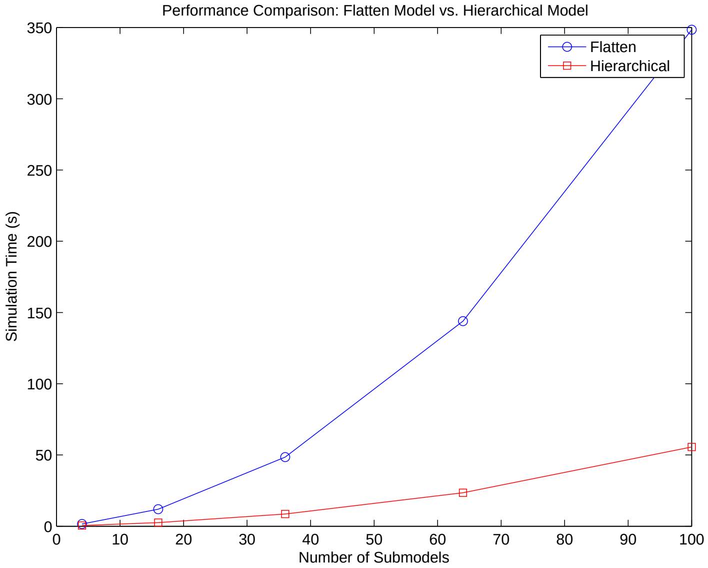
Always 0

τ	υ	μ
0.2	C2	R1

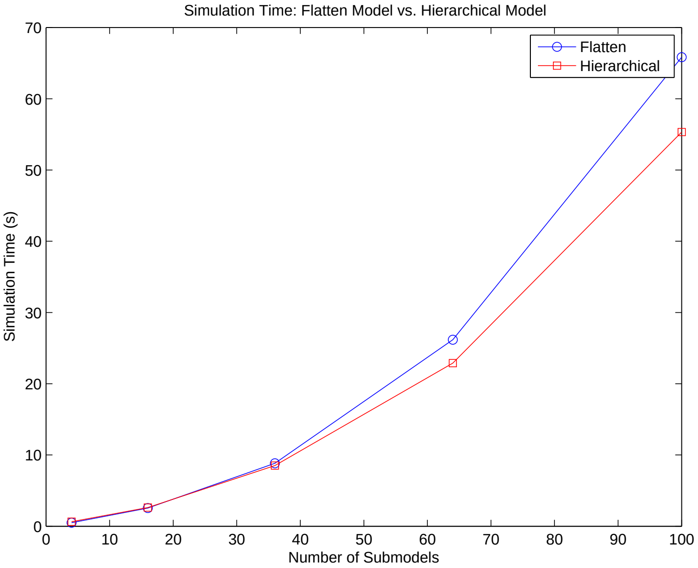
Results

- The first test
  - Top-level grid model populated with repressilator sub-models without replacements or deletions.
  - Size of 4, 16, 36, 64, and 100 sub-models.
- The second test
  - Top-level grid model populated with repressilator sub-models with replacements and deletions.
  - GFP protein is replaced by a top-level GFP protein that tracks the total amount across all sub-models.
  - Degradation reaction of the GFP reporter protein is deleted from all sub-models.
  - Size of 1, 4, 9, 15, 25, and 50 sub-models.

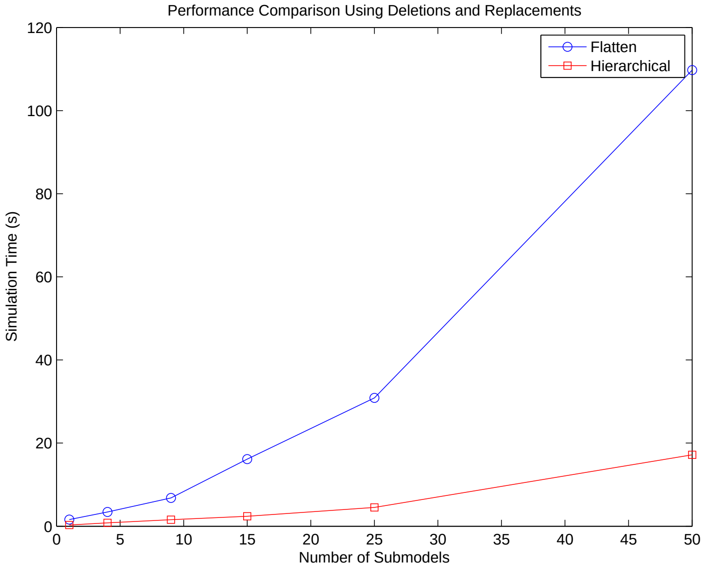
Performance Without Replacements/Deletions



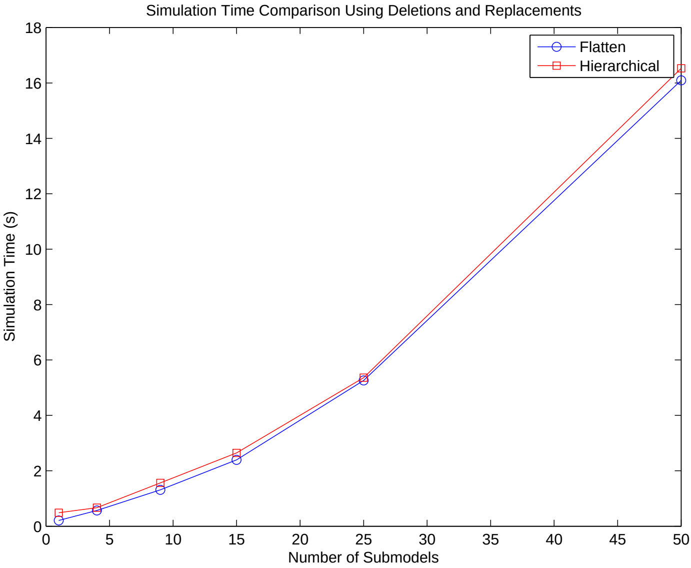
Simulation Time Without Replacements/Deletions



Performance With Replacements/Deletions



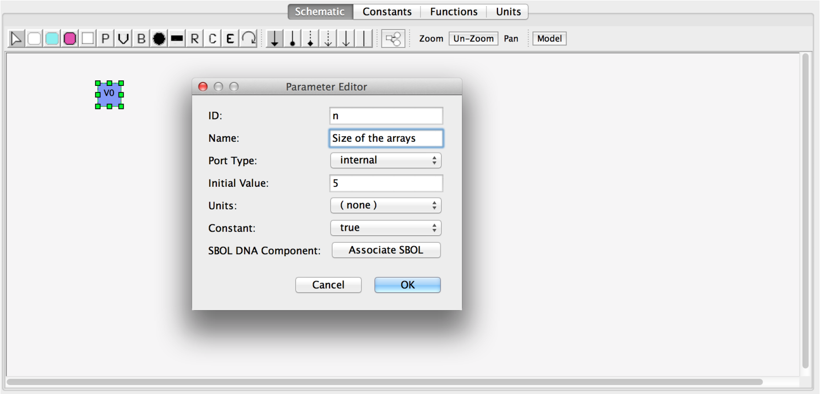
Simulation Time With Replacements/Deletions

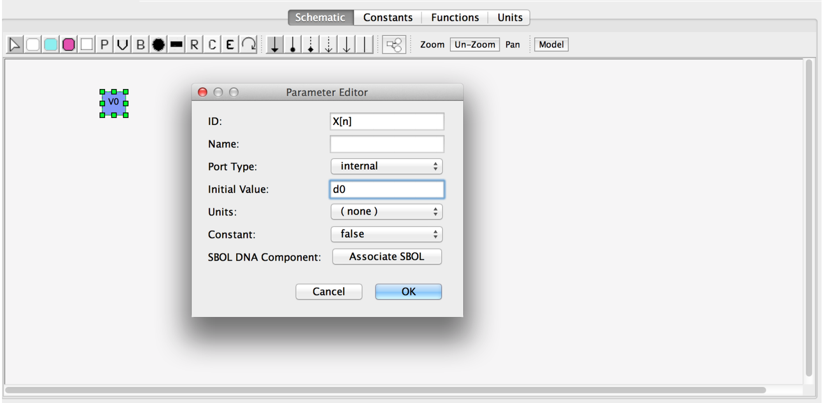


SBML Arrays Package

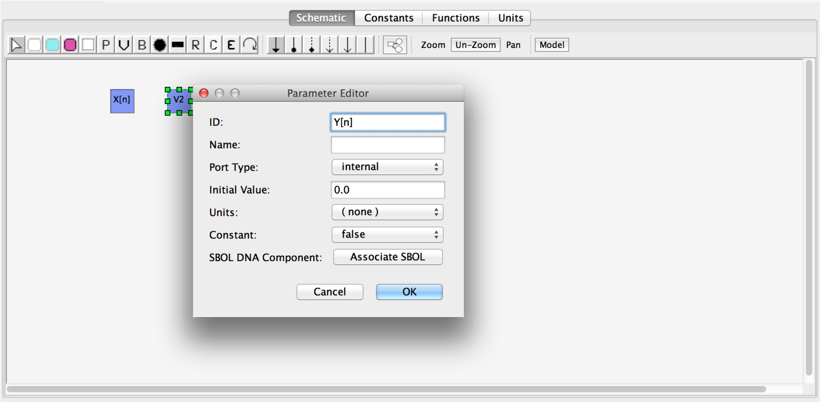
- Mathematical operations in SBML L3V1 core are restricted to operations on scalar values.
- Regular structures such as cellular populations cannot be represented efficiently.
- This motivated the development of the arrays package.

iBioSim and Arrays package (Creating Constant Parameter)

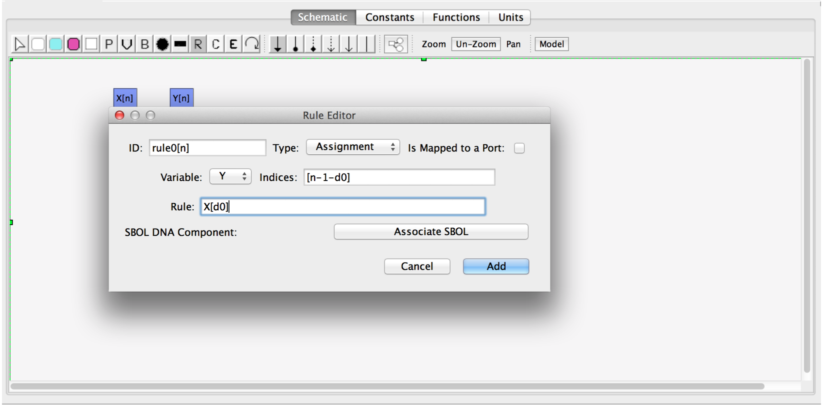



Schematic
Constants
Functions
Units
P
B
R
C
E
Zoom
Un-Zoom
Pan
Model
V0
Parameter Editor
ID:
X[n]
Name:
Port Type:
internal
Initial Value:
d0
Units:
(none)
Constant:
false
SBOL DNA Component:
Associate SBOL
Cancel
OK

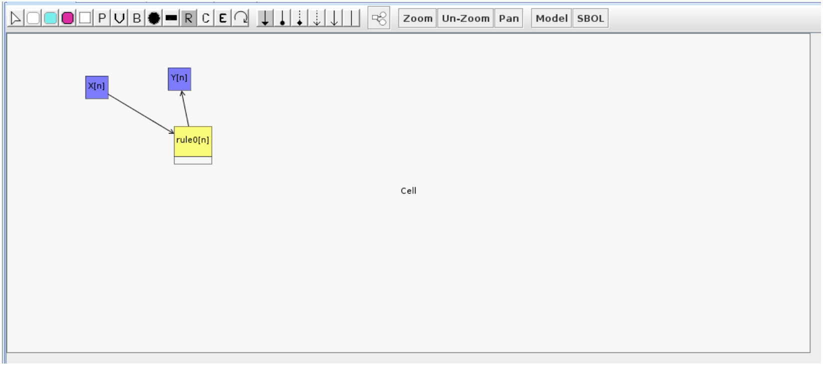
iBioSim and Arrays package (Creating Array Y)



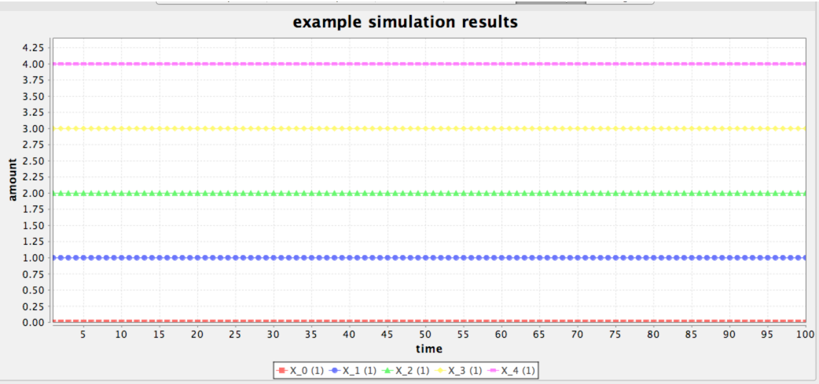
iBioSim and Arrays package (Creating Array of Rule)



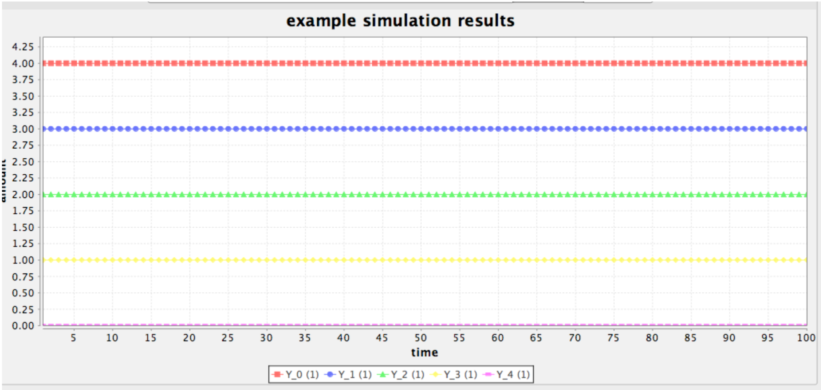
iBioSim and Arrays package (Full Model)



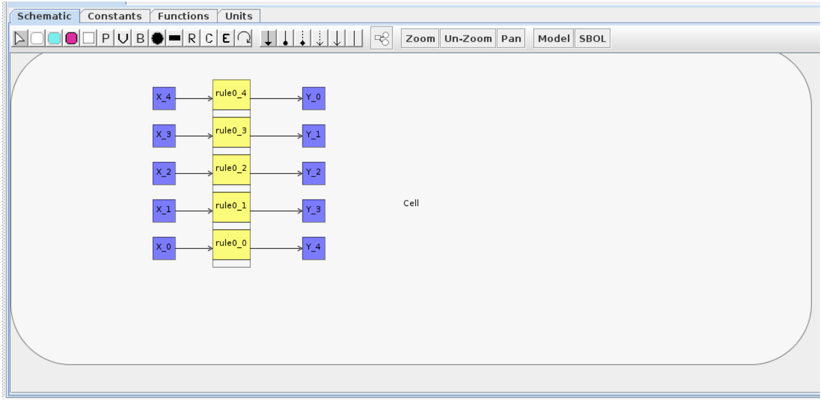
iBioSim Simulation Results



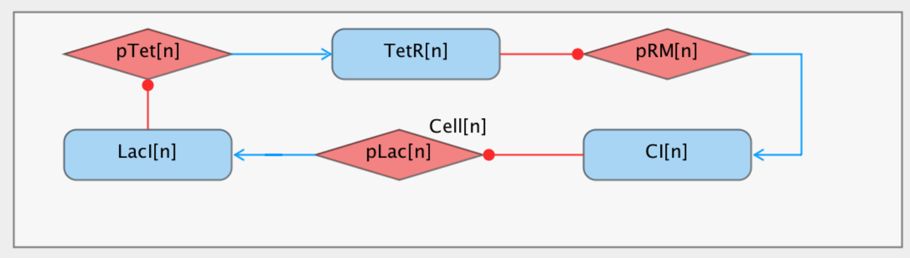
iBioSim Simulation Results (cont.)



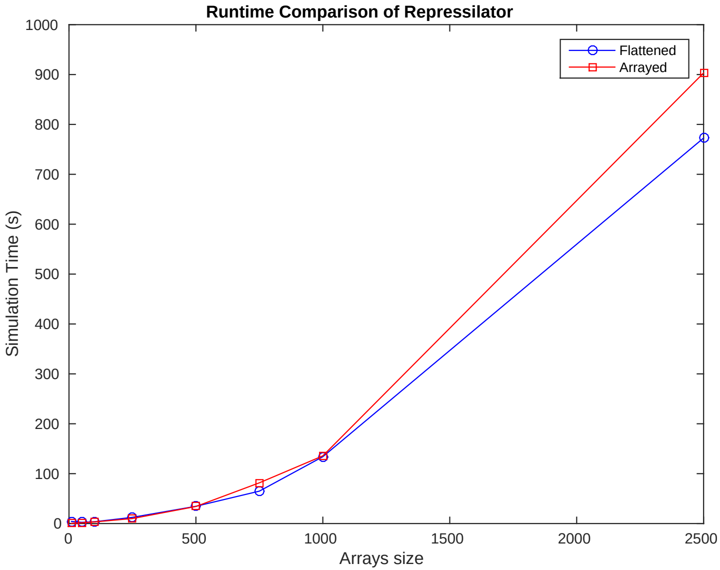
Flattened Model



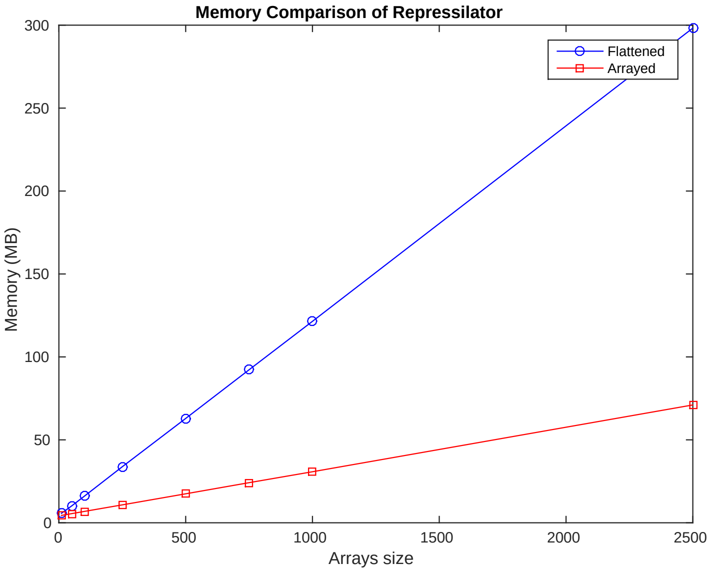
Population of Repressilator Circuits Using Arrays



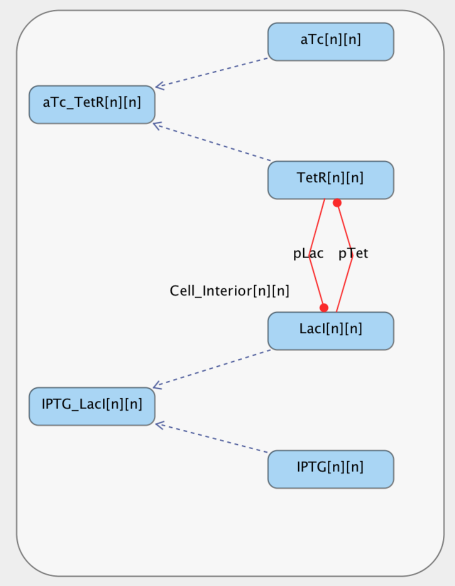
Runtime Comparison for Repressilator



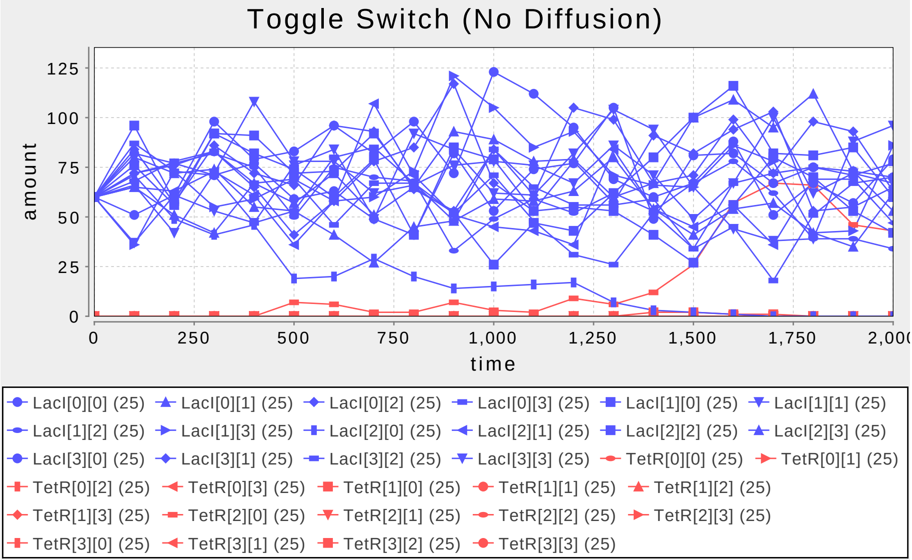
Memory Comparison for Repressilator



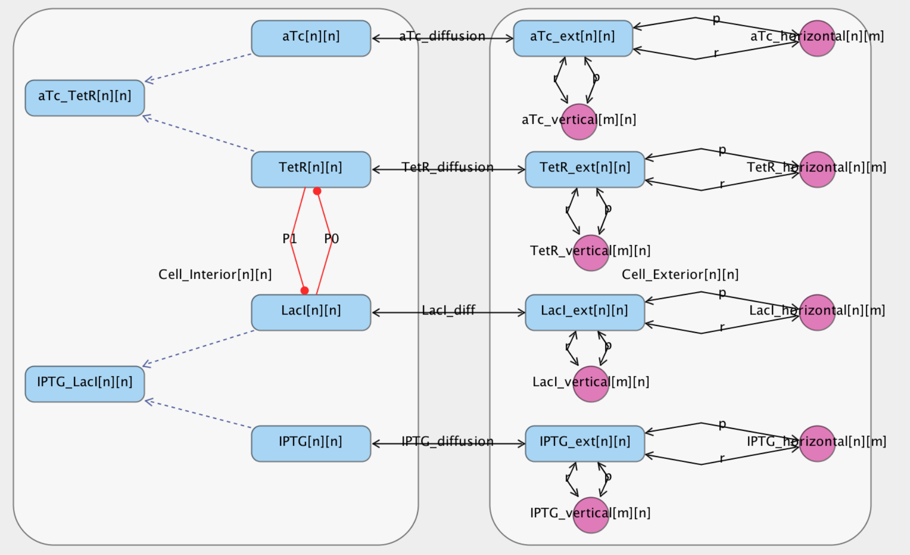
Population of Genetic Toggle Circuits Using Arrays



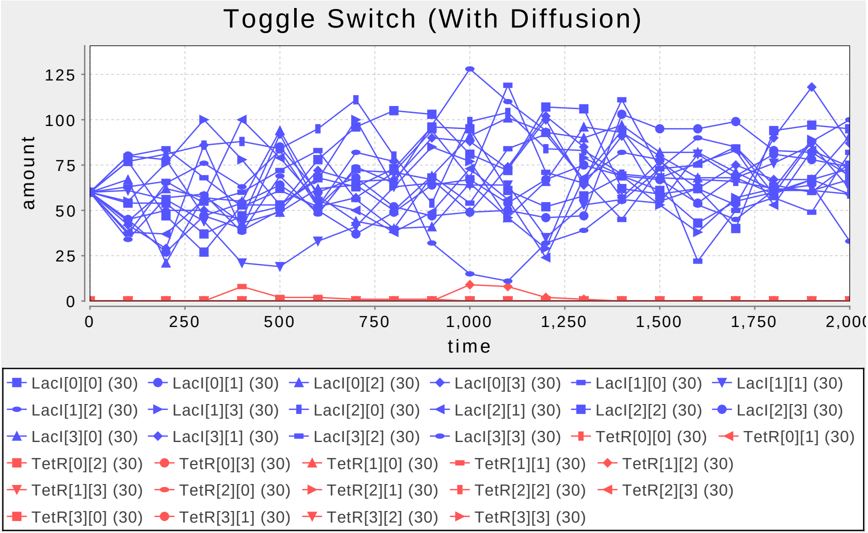
Simulation of Population of Genetic Toggle Circuits



Population of Genetic Toggle Circuits with Diffusion



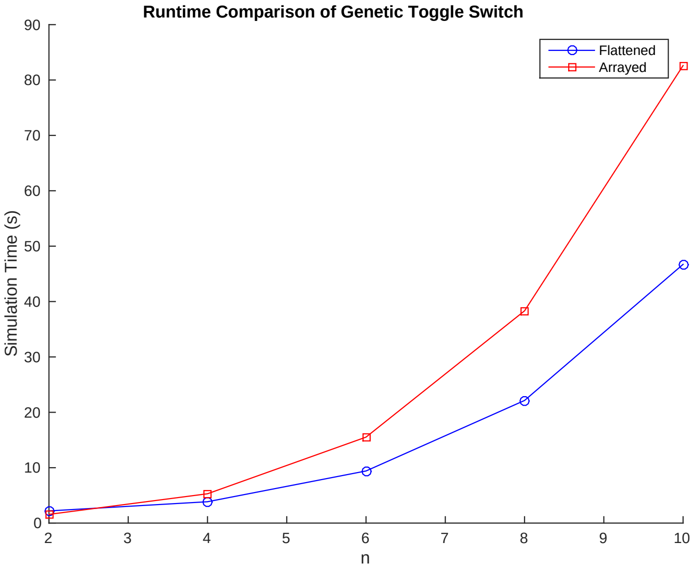
Simulation of Population of Genetic Toggle Circuits with Diffusion



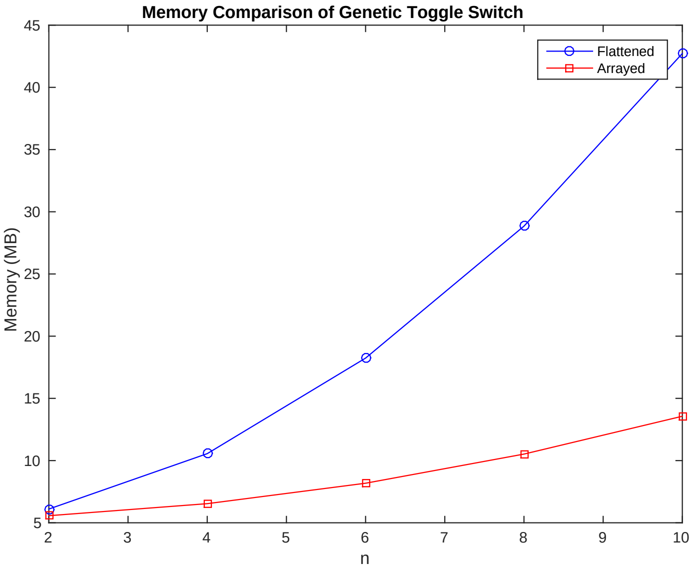
Probability of a Cell Entering a Bad State

Model	Number of Cells	Number of Failures	Probability
Without Diffusion	18,750	219	$\sim$ 1.2 %
With Diffusion	18,750	90	$\sim$ 0.5 %

Runtime Comparison for Genetic Toggle with Diffusion



Memory Comparison for Genetic Toggle with Diffusion



Discussion

- Results have shown that the hierarchical simulator scales better than the simulator with the SSA with flattening.
- While simulation time is equivalent, the flattening cost is avoided.
- Future Work
  - Support dynamic events.
  - Explore dynamic model abstraction.
  - Enable parallel processing.

‘

Motivation

- In biological systems, wide deviations from normal behavior may occur with extremely small probability.
- Rare events can have significant consequences in biological systems.
- Analysis of rare events can have significant computational costs.
- The *weighted SSA* (wSSA) targets this problem.

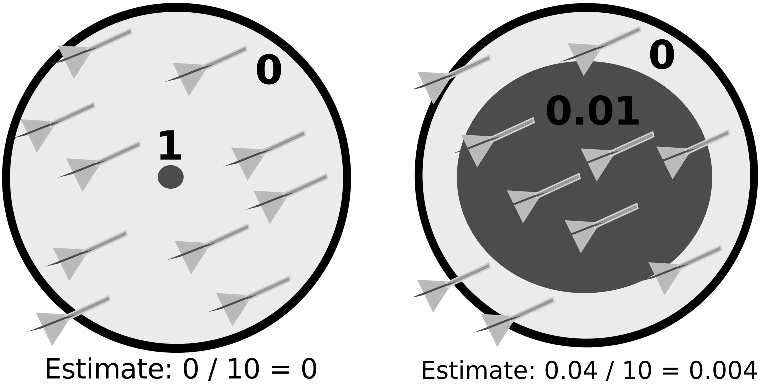
Background

- Consider $P_{t \leq t_{max}}(\mathbf{X} \rightarrow \mathcal{E} \mid \mathbf{x_0})$, the probability that $\mathbf{X}$ moves to a state in $\mathcal{E}$ within time limit t_{max}, given $\mathbf{X}(0) = \mathbf{x_0}$ where $\mathbf{x_0} \notin \mathcal{E}$.
- With SSA, generate n runs and report the *sample average*: $1/n \sum_{i=1}^{n} Y_i$ where $Y_i = 1$ if $\mathbf{X}(t)$ moves to a state in $\mathcal{E}$ before t_{max}, otherwise $Y_i = 0$.
- Finding probability of rare event requires a large number of runs.
- Example:
  - Switching rate from the lysogenic state to the lytic state in phage λ is experimentally estimated to be in the order of 10^{-7} per cell per generation.
  - Using SSA, this rare event occurs only once every 10^7 runs.
  - Therefore, more than 10^{11} simulation runs are needed to estimate the probability with a 95 percent confidence interval.

Importance Sampling

- The wSSA increases the chance of observing the rare events of interest by utilizing the *importance sampling* technique.
- Importance sampling manipulates the probability distribution to observe events of interest more often than when using conventional sampling.
- The outcome of each biased sampling is weighted by a likelihood factor to yield statistically correct and unbiased results.

Simple Example



Predilection Functions

- To observe rare events more often, the wSSA uses *predilection functions*, $b_j(\mathbf{x})$, rather than propensity functions, $a_j(\mathbf{x})$.
- The index of the next reaction is selected with the following probability:

$$Prob\{\text{the next reaction index is } j \text{ given } \mathbf{X} = \mathbf{x}\} = \frac{b_j(\mathbf{x})}{b_0(\mathbf{x})},$$

  where $b_0(\mathbf{x}) \equiv \sum_{j=1}^{m} b_j(\mathbf{x})$.
- To correct the sampling bias, each reaction is weighted as follows:

$$w(j, \mathbf{x}) = \frac{a_j(\mathbf{x}) b_0(\mathbf{x})}{a_0(\mathbf{x}) b_j(\mathbf{x})}.$$

Probability of a Reaction Sequence

- $P_k(\sigma \mid \mathbf{x_0})$ is the probability of *reaction sequence* $\sigma = (R_{j_1}, R_{j_2}, \ldots, R_{j_k})$ given that the initial state is x_0.
- Since $\mathbf{X}(t)$ is Markovian, the joint conditional probability is as follows:

$$\begin{aligned}
P_k(\sigma \mid \mathbf{x_0}) &= \prod_{h=1}^{k} \frac{a_{j_h}(\mathbf{x_{h-1}})}{a_0(\mathbf{x_{h-1}})} \\
&= \prod_{h=1}^{k} \left[\frac{a_{j_h}(\mathbf{x_{h-1}}) b_0(\mathbf{x_{h-1}})}{b_{j_h}(\mathbf{x_{h-1}}) a_0(\mathbf{x_{h-1}})} \right] \frac{b_{j_h}(\mathbf{x_{h-1}})}{b_0(\mathbf{x_{h-1}})} \\
&= \prod_{h=1}^{k} w(j_h, \mathbf{x_{h-1}}) \prod_{h=1}^{k} \frac{b_{j_h}(\mathbf{x_{h-1}})}{b_0(\mathbf{x_{h-1}})}.
\end{aligned}$$

where $\mathbf{x_h} = \mathbf{x_0} + \sum_{h'=1}^{h-1} \mathbf{v_{j_{h'}}}$.

Weighted Sample Average

- The estimate of $P_{t \leq t_{max}}(\mathbf{X} \rightarrow \mathcal{E} \mid \mathbf{x_0})$ is calculated by first defining the statistical weight of the i-th sample trajectory w_i such that:

$$w_i = \begin{cases} \prod_{h=1}^{k_i} w(j_h, \mathbf{x_{h-1}}) & \text{if } \mathbf{X}(t) \text{ moves to a state in } \mathcal{E} \text{ within the time limit,} \\ 0 & \text{otherwise,} \end{cases}$$

  where k_i is the number of jumps in the i-th sample trajectory.
- Then, $P_{t \leq t_{max}}(\mathbf{X} \rightarrow \mathcal{E} \mid \mathbf{x_0})$ is estimated by taking a sample average of w_i:

$$\frac{1}{n} \sum_{i=1}^{n} w_i.$$

Choice of Predilection Functions

- With adequate choice of predilection functions, wSSA increases the fraction of sample trajectories that result in the rare events.
- For each reaction R_j, $b_j(\mathbf{x})$ is defined as:

$$b_j(\mathbf{x}) = \alpha_j \times a_j(\mathbf{x}),$$

  where each $\alpha_j > 0$ is a constant.
- Example:
  - Determine probability that S transitions from θ_1 to θ_2 where $\theta_1 < \theta_2$.
  - Increase predilection functions for the production reactions of S and/or decrease the predilection functions for the degradation reactions of S.

wSSA Algorithm

1. Initialize $q = 0$ and $k = 1$.
2. Initialize: $w = 1$, $t = t_0$, and $\mathbf{x} = \mathbf{x}_0$.
   1. Evaluate $a_j(\mathbf{x})$, $a_0(\mathbf{x}) = \sum_{j=1}^{m} a_j(\mathbf{x})$, $b_j(\mathbf{x})$, and $b_0(\mathbf{x}) = \sum_{j=1}^{m} b_j(\mathbf{x})$.
   2. Draw two unit uniform random numbers, r_1 and r_2.
   3. Determine the time, τ, until the next reaction:
   $$\tau = \frac{1}{a_0(\mathbf{x})} \ln\left(\frac{1}{r_1}\right).$$
   4. Determine the next reaction, R_μ:
   $$\mu = \text{ the smallest integer satisfying } \sum_{j=1}^{\mu} b_j(\mathbf{x}) > r_2 b_0(\mathbf{x}).$$
   5. Determine the sequence weight: $w = w \times (a_\mu(\mathbf{x})/b_\mu(\mathbf{x})) \times (b_0(\mathbf{x})/a_0(\mathbf{x}))$.
   6. Determine the new state: $t = t + \tau$ and $\mathbf{x} = \mathbf{x} + v_\mu$.
   7. If $\mathbf{x} \in \mathcal{E}$ then $q = q + w$ else if $t \leq t_{max}$ then goto step 1.
3. $k = k + 1$, if $k \leq n$ then goto step 2 else report q/n as the probability.

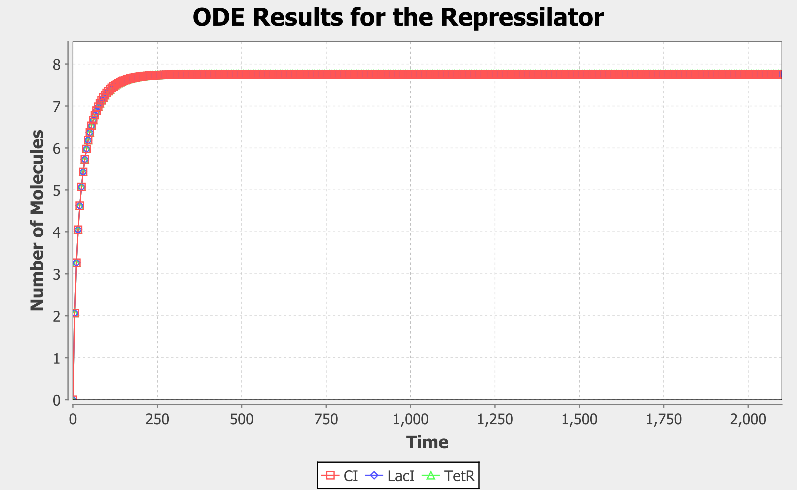
Discussion

- Key to success is the choice of predilections functions.
- A procedure to choose optimized α_j by running several test runs to compute the variance of the statistical weights has been proposed.
- More research is likely needed here.

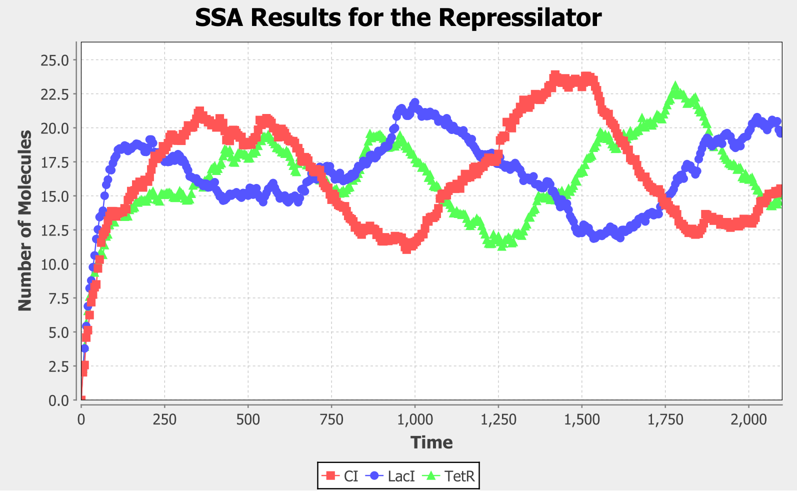
Incremental Stochastic Simulation Algorithm (iSSA)

- Built off of Gillespie's SSA.
- Performs simulation runs in small time-increments.
- Statistics are computed at the end of each increment.
- Computed statistics are then used to constrain the initial condition for the subsequent increment.
- Ensures that the generated sample paths are functionally coherent and yield meaningful statistics.

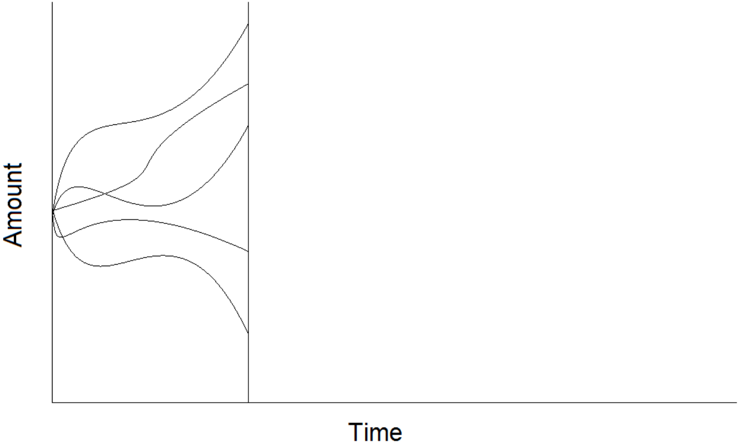
Repressilator: ODE Simulation



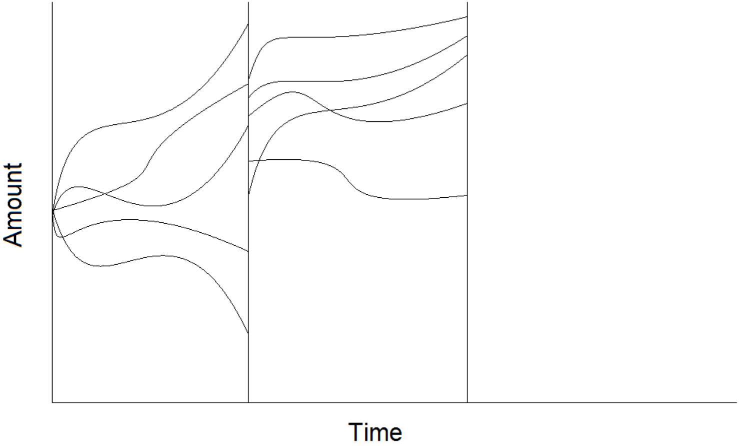
Repressilator: SSA Simulation



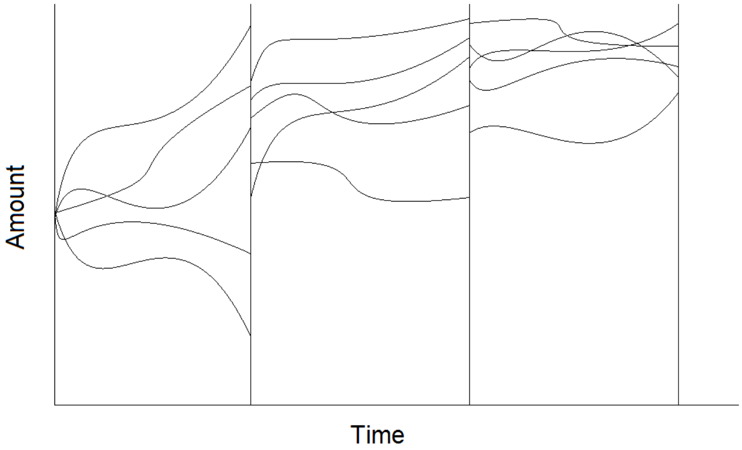
Incremental Stochastic Simulation Algorithm (iSSA)



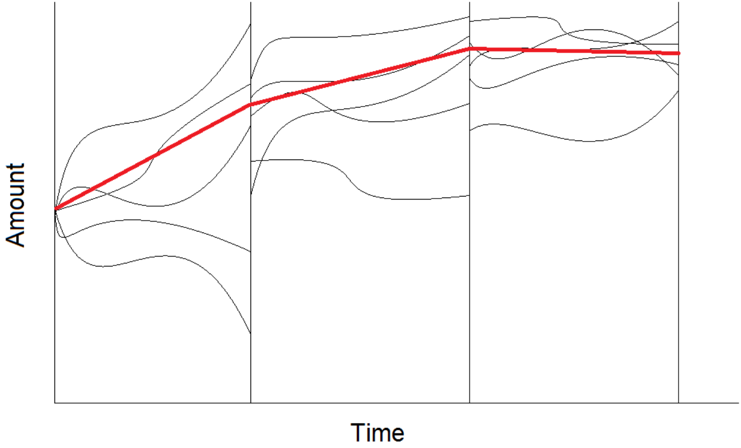
Incremental Stochastic Simulation Algorithm (iSSA)



Incremental Stochastic Simulation Algorithm (iSSA)



Incremental Stochastic Simulation Algorithm (iSSA)



iSSA

1. Initialize: $k = 1$ and $\mathbf{X}^{(0)} = \langle t_0, \mathbf{x}_0 \rangle$.
2. Set $i = 1$.
3. Set $\langle t, \mathbf{x} \rangle = \mathtt{select}(\mathbf{X}^{(k-1)})$ and $\mathtt{start} = t$.
4. Set $\mathtt{limit} = \mathtt{findLimit}(\mathtt{start}, t, \mathbf{x})$.
5. Execute a Gillespie SSA step.
6. If $t < \mathtt{limit}$ then go to step 4.
7. $\mathtt{record}(\mathbf{X}^{(k)}, t, \mathbf{x}, i)$.
8. If $i < \mathtt{maxRuns}$ then $i = i + 1$, go to step 3.
9. If $t < \mathtt{timeLimit}$ then $k = k + 1$, go to step 2.

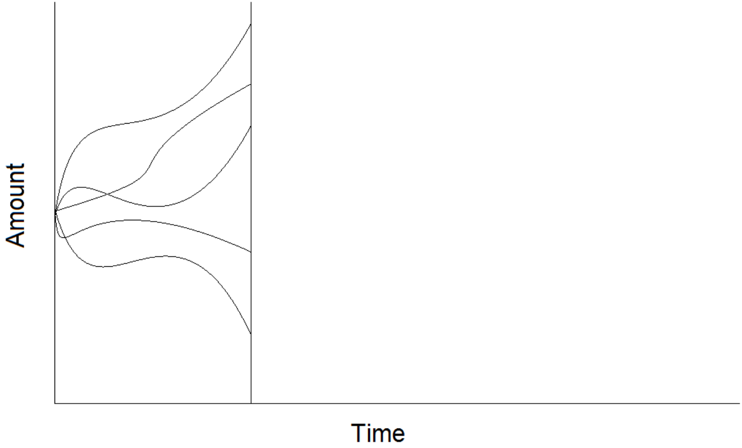
Variations

- Variants of iSSA are derived by altering:
  - How each time increment is calculated.
  - How starting states are selected in each increment.
  - What information is stored during each increment.

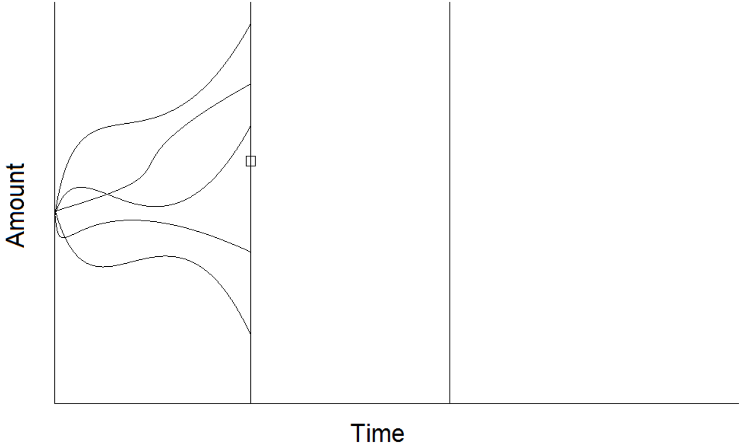
Marginal Probability Density Evolution (MPDE)

- Generates a probability distribution for each species.
- Defined as follows:
  - Stores the average and variance over all the species.
  - When all runs reach the end of an increment, a *probability density function* (pdf) is approximated for each species.
  - Uses the pdf to randomly generate a new starting state.
- MPDE can be used if known correlations are stated explicitly as constraints in the reaction model.
- Must reject any state that violates this constraint.

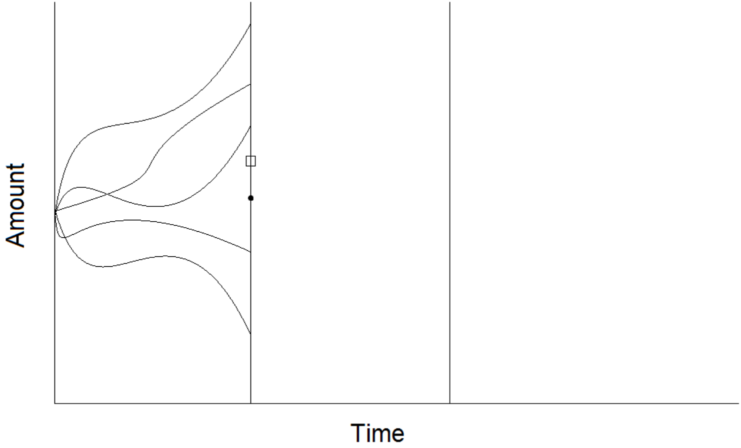
Marginal Probability Density Evolution (MPDE)



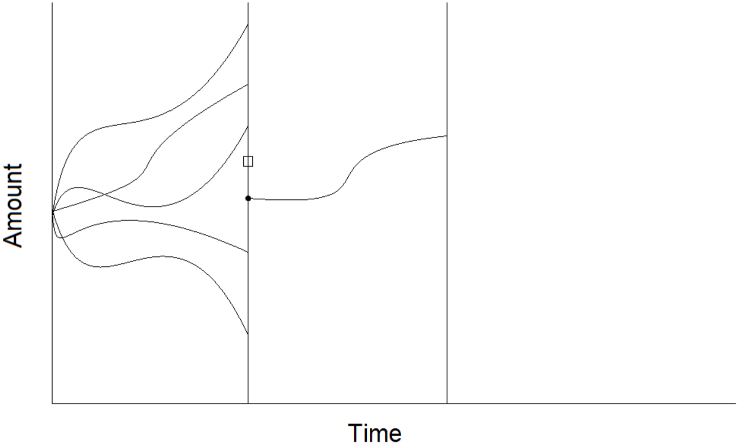
Marginal Probability Density Evolution (MPDE)



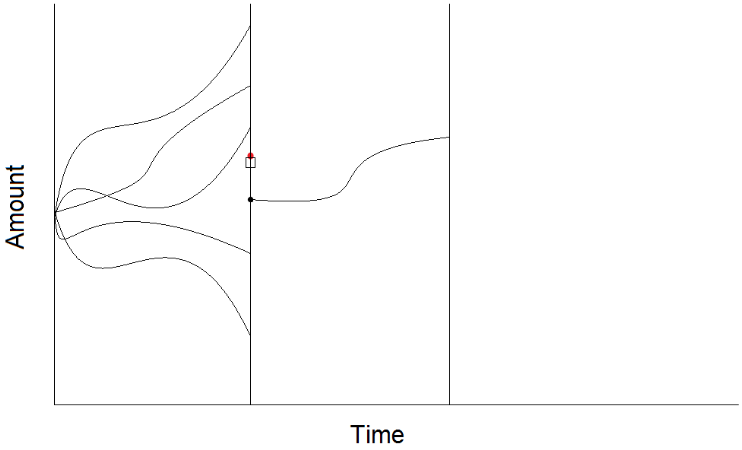
Marginal Probability Density Evolution (MPDE)



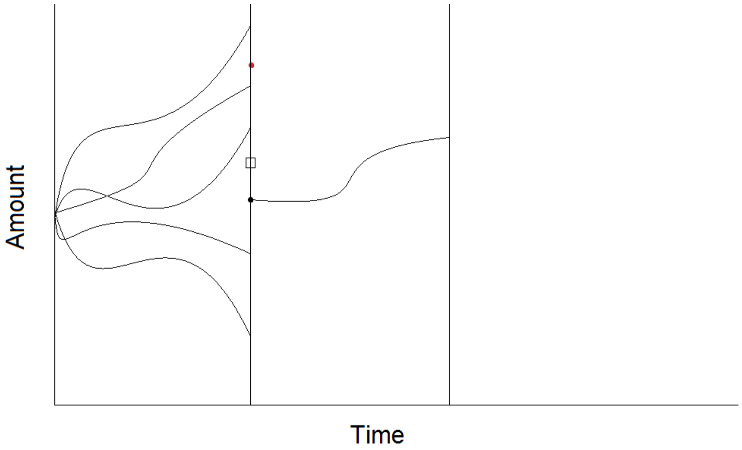
Marginal Probability Density Evolution (MPDE)



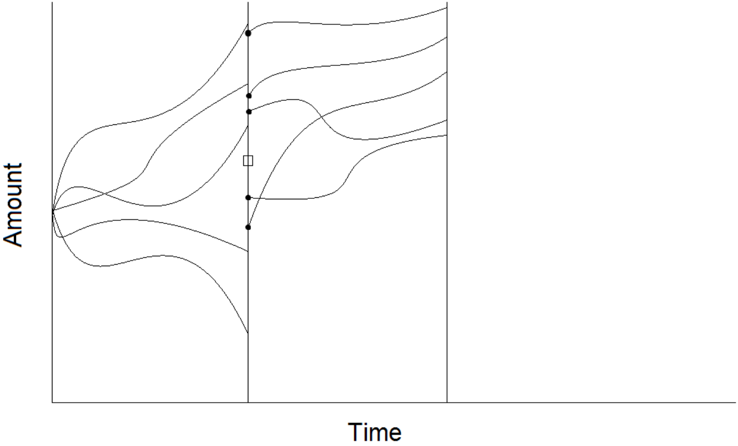
Marginal Probability Density Evolution (MPDE)



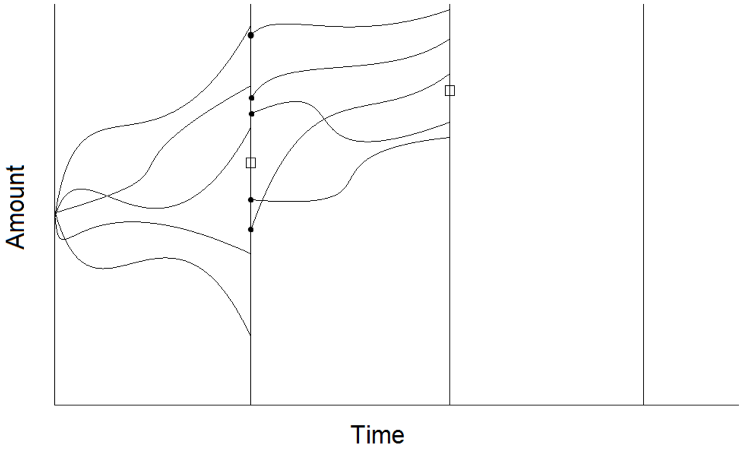
Marginal Probability Density Evolution (MPDE)



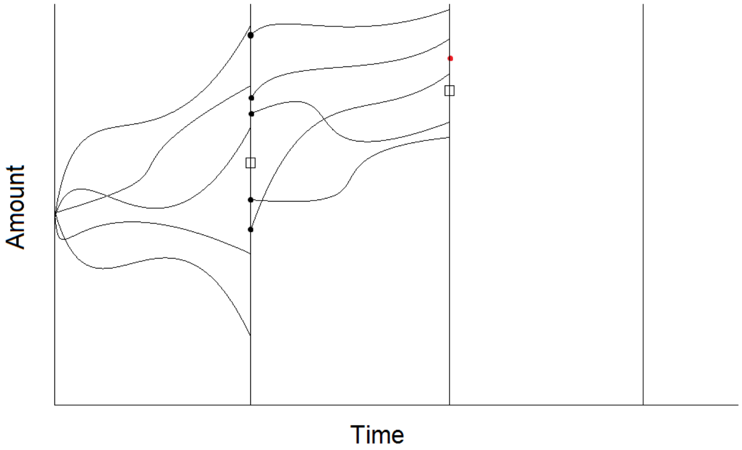
Marginal Probability Density Evolution (MPDE)



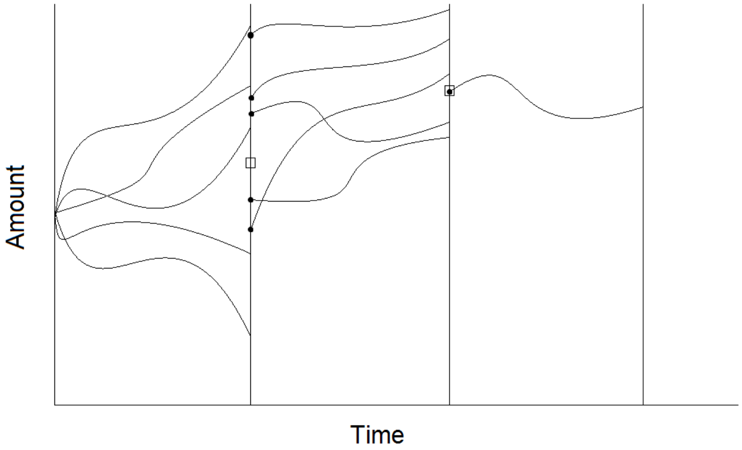
Marginal Probability Density Evolution (MPDE)



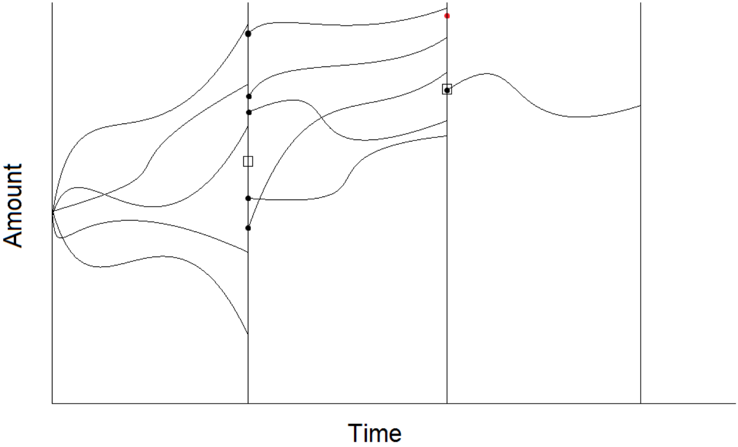
Marginal Probability Density Evolution (MPDE)



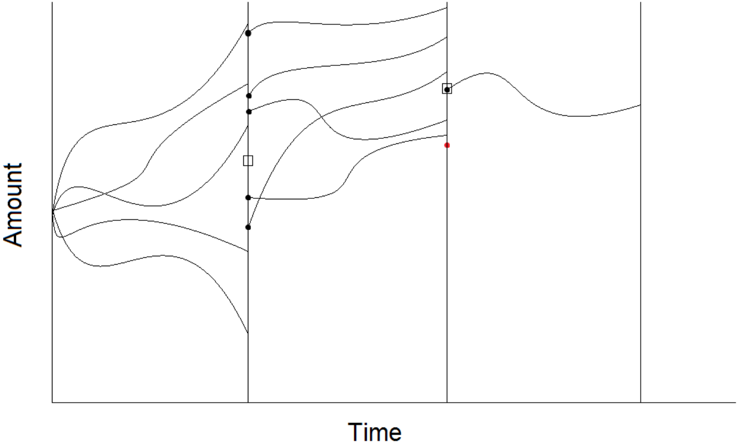
Marginal Probability Density Evolution (MPDE)



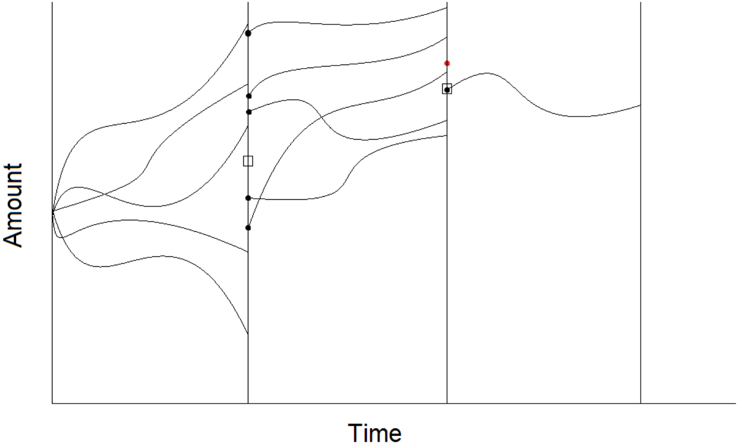
Marginal Probability Density Evolution (MPDE)



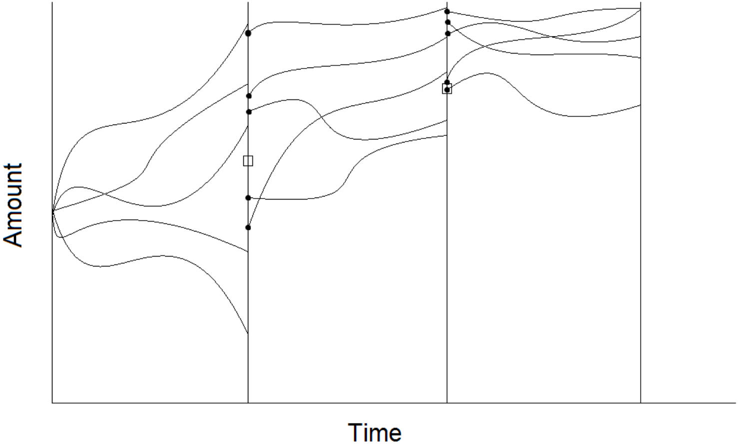
Marginal Probability Density Evolution (MPDE)



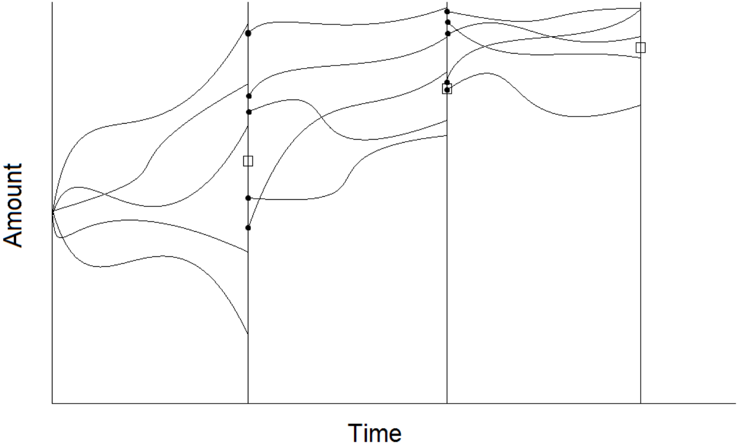
Marginal Probability Density Evolution (MPDE)



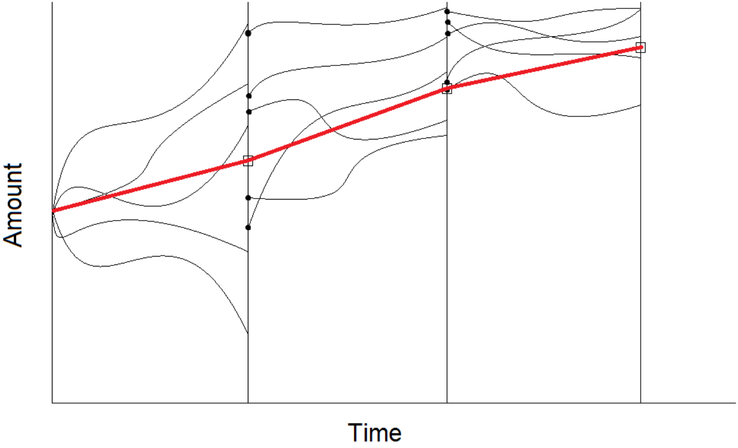
Marginal Probability Density Evolution (MPDE)



Marginal Probability Density Evolution (MPDE)



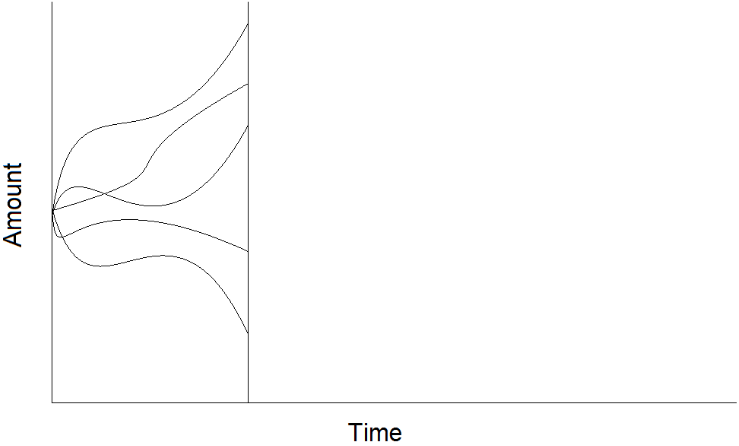
Marginal Probability Density Evolution (MPDE)



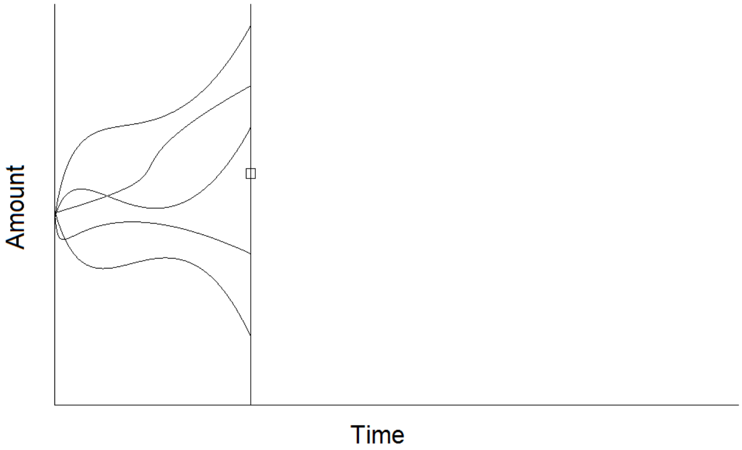
Mean Path

- MPDE relies on a statistical approximation that limits the conditions under which it can be trusted.
- An alternative method is mean path, which is defined as follows:
  - Stores the SSA states in the state table.
  - The average state is computed at the end of each increment.
  - Selects the state that has the smallest Euclidean distance from the average state as the starting state for the next increment.
- Produces an actual simulation trace of the mean path representing statistics on typical behavior.
- Eliminates the need for added constraints and allows reaction-based abstraction to be applied to improve simulation efficiency.

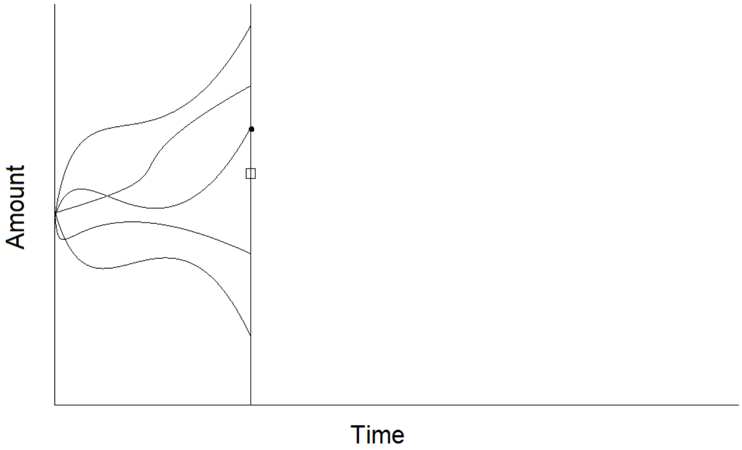
Mean Path



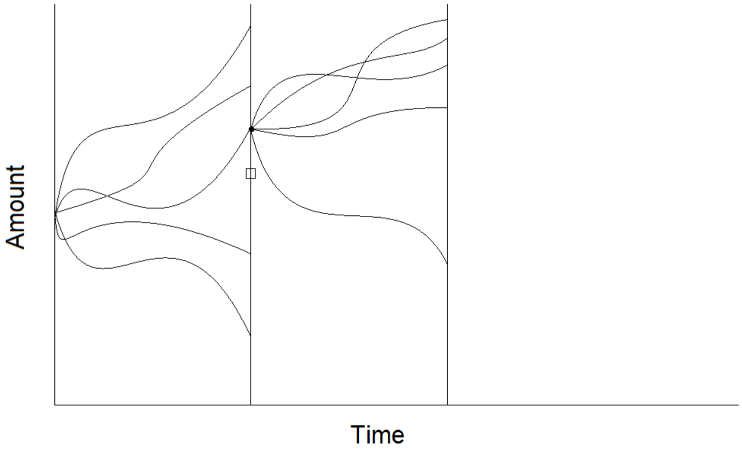
Mean Path



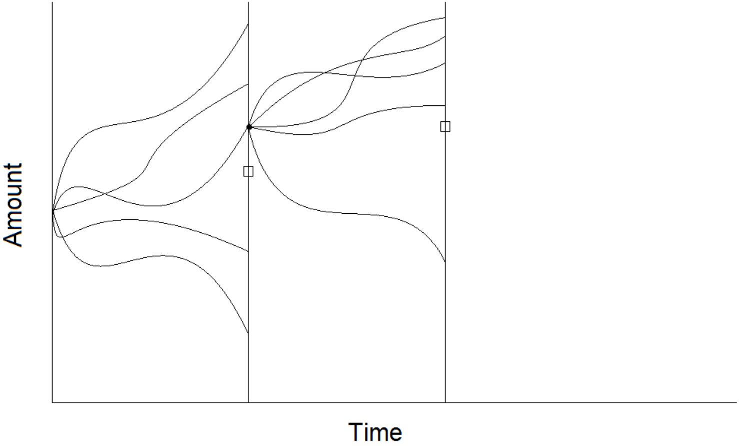
Mean Path



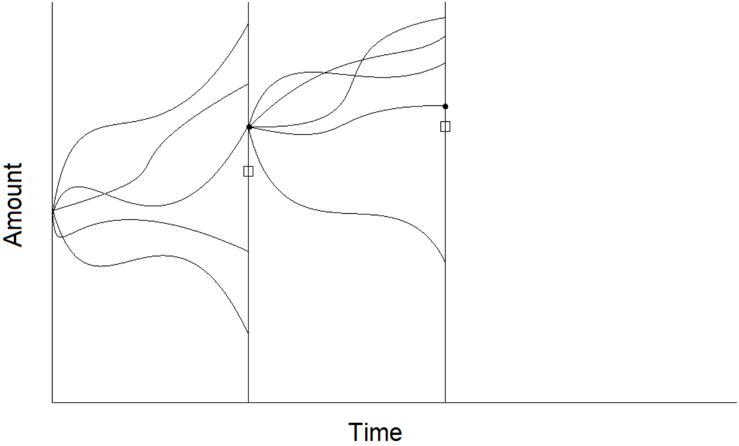
Mean Path



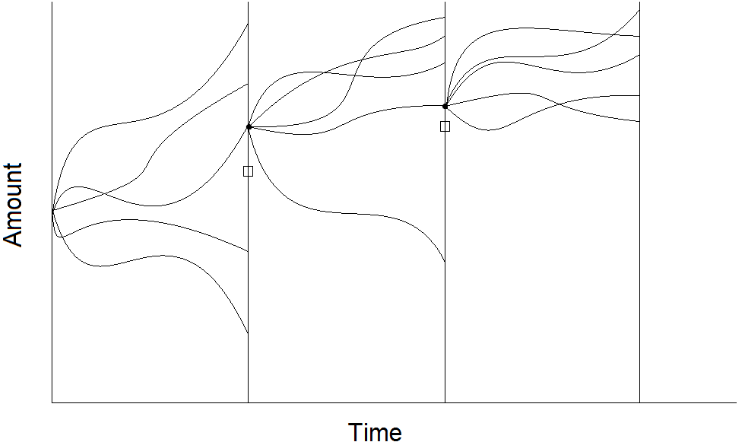
Mean Path



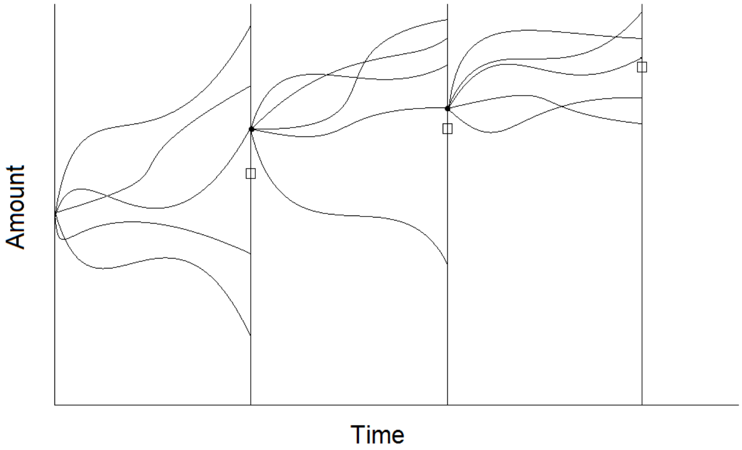
Mean Path



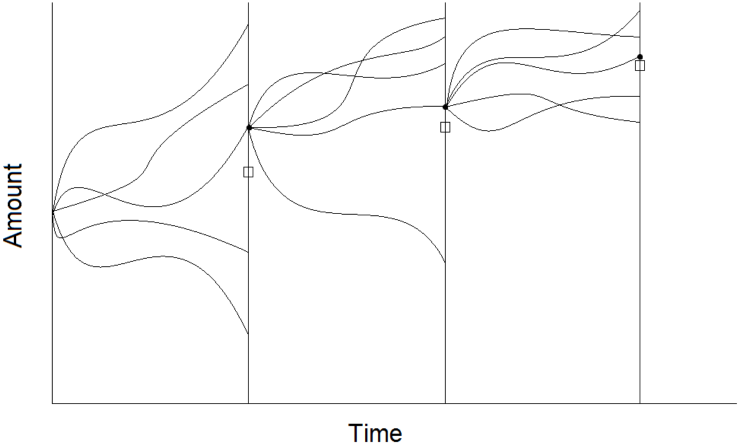
Mean Path



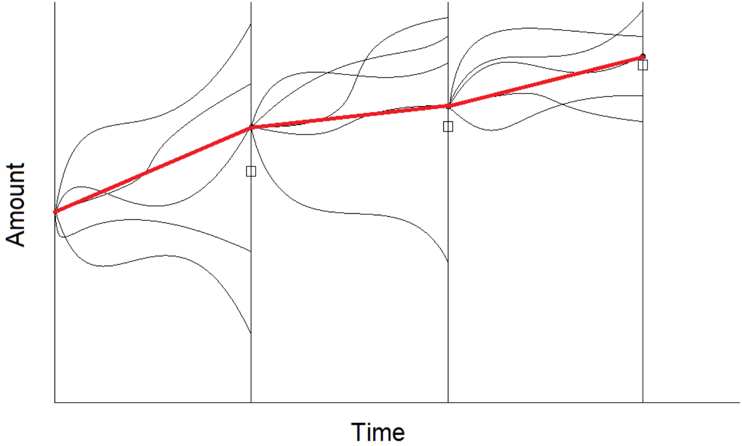
Mean Path



Mean Path



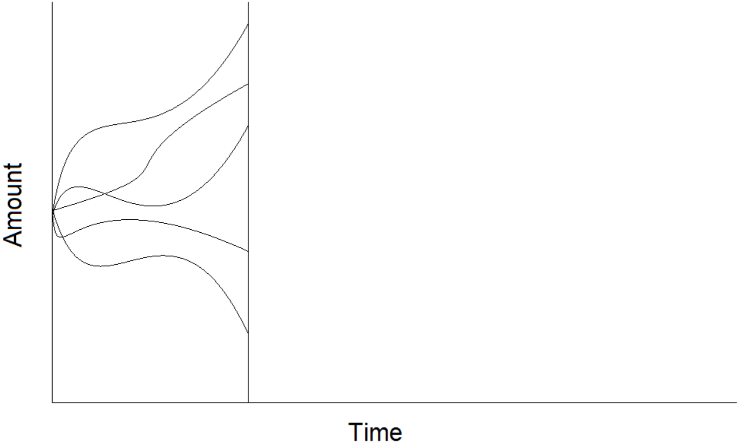
Mean Path



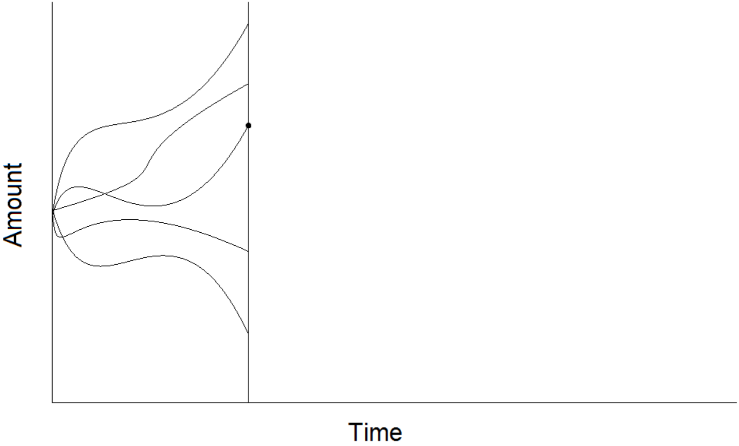
Median Path

- One or more simulation traces may diverge so much that the ending states become outliers in the average state calculation.
- The mean path method may end up selecting a state that does not represent the “typical" behavior of the system.
- Instead, find the state with the smallest Euclidean distance from the median state and use this state as the starting state in the subsequent time increment.

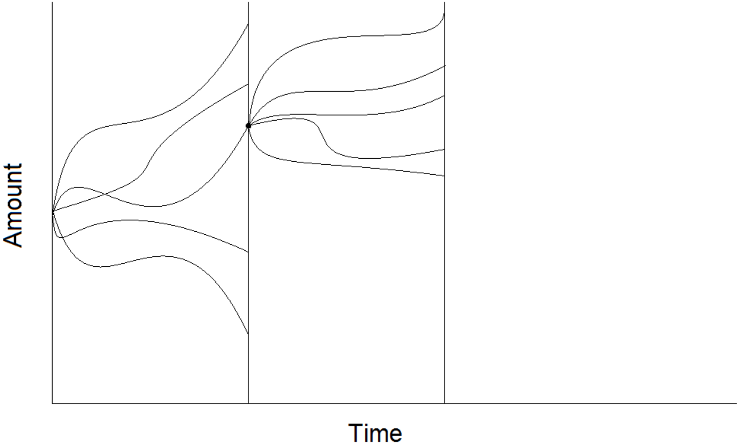
Median Path



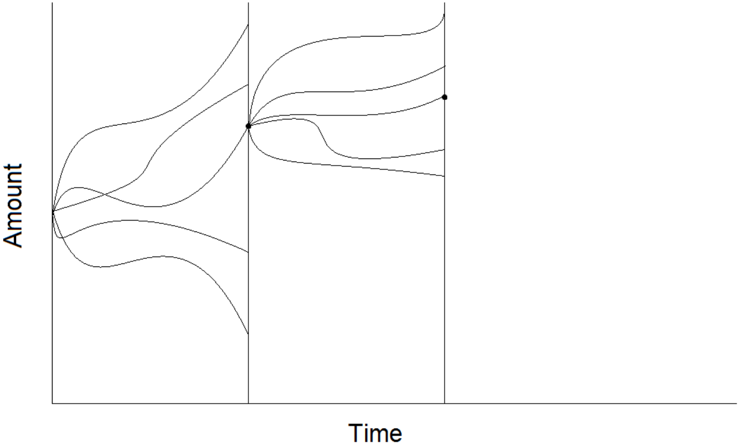
Median Path



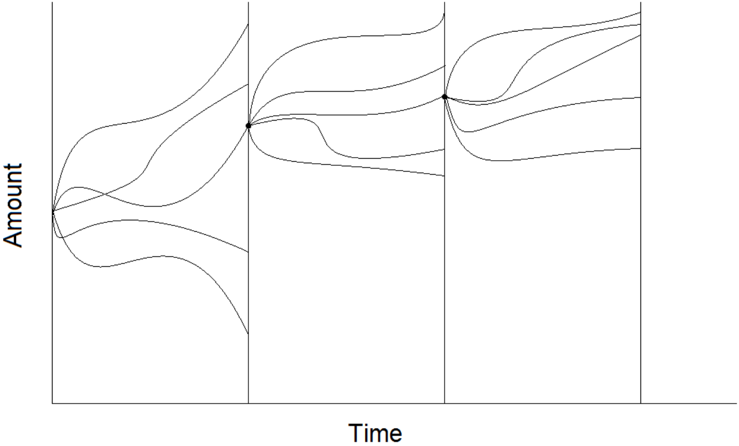
Median Path



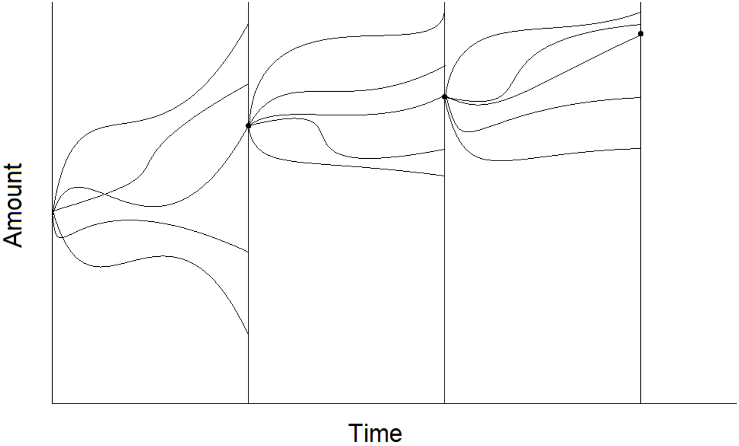
Median Path



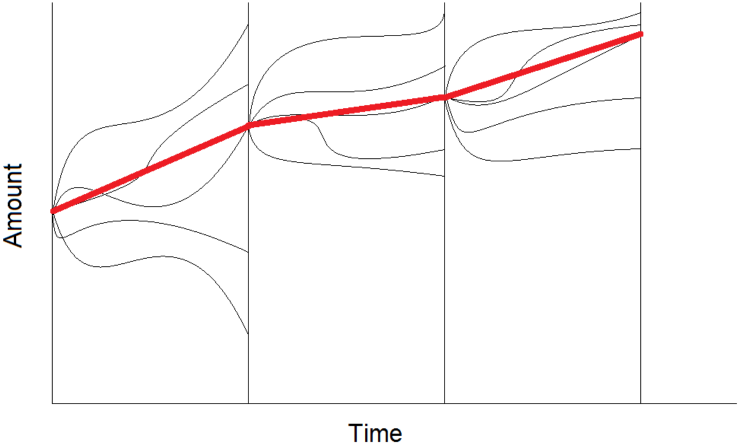
Median Path



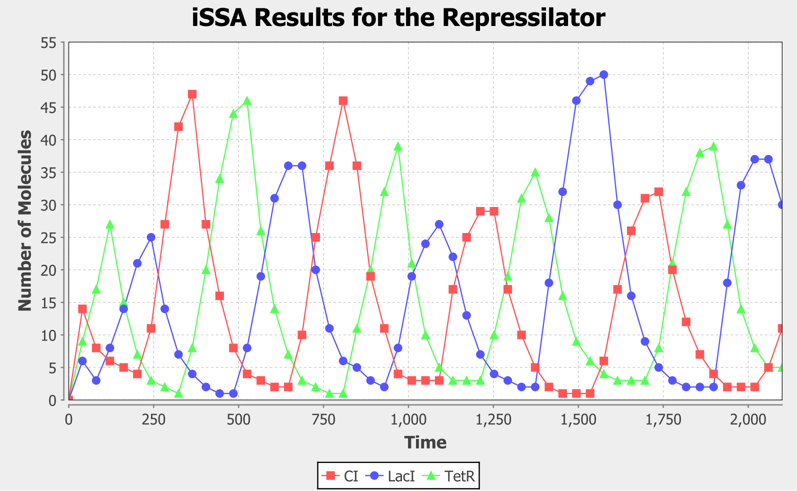
Median Path



Median Path



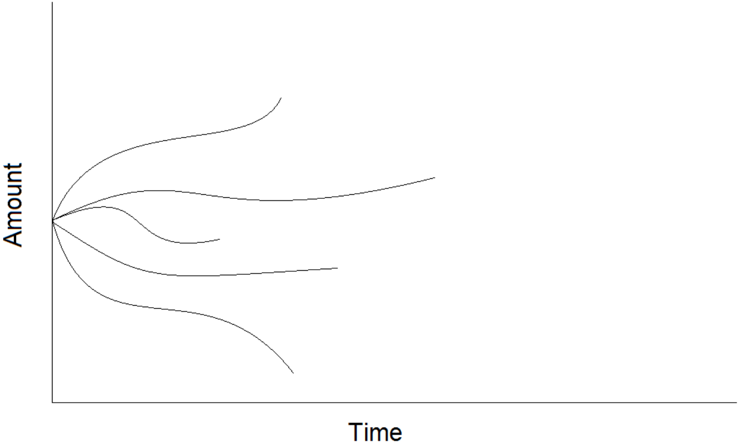
Repressilator: iSSA Simulation



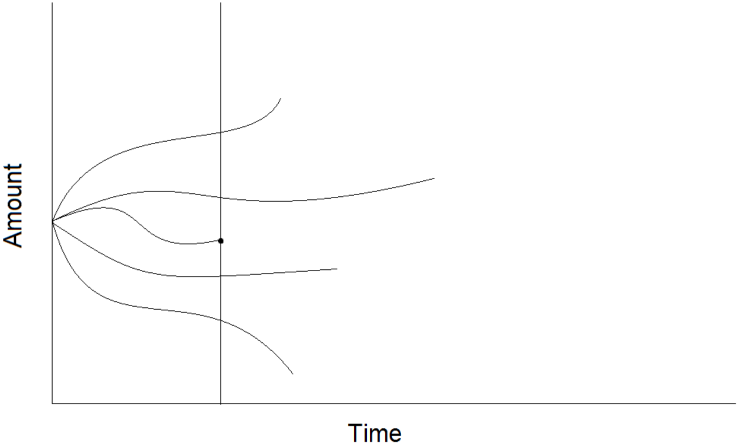
Adaptive Time Step

- For small increments, too few reactions are observed each increment.
- For large increments, too many reactions are observed each increment.
- Instead of specifying the size of the time increment, a user can specify a desired number of slow reaction events per increment.
- No matter how fast or slow the system evolves, the algorithm adjusts the time increment to capture approximately the same number of slow reaction events.

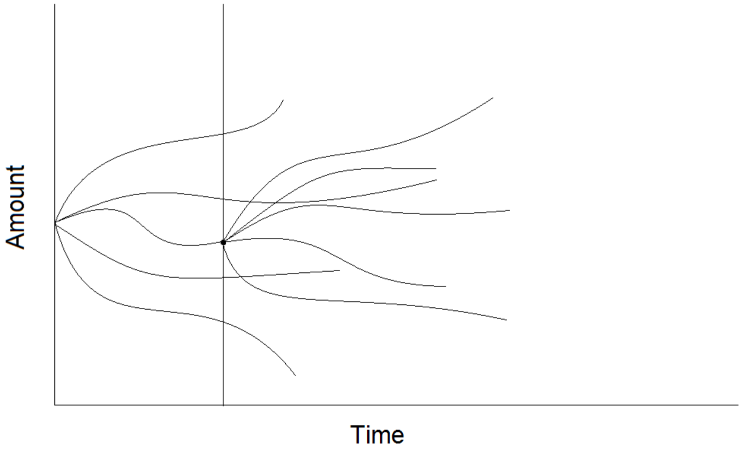
Adaptive Time Step



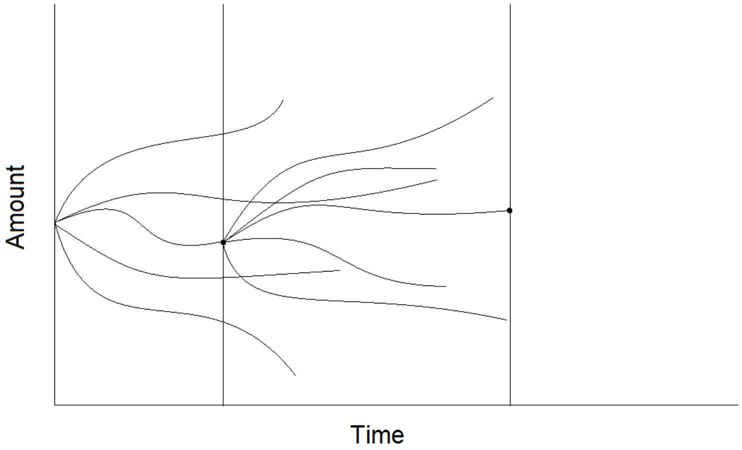
Adaptive Time Step



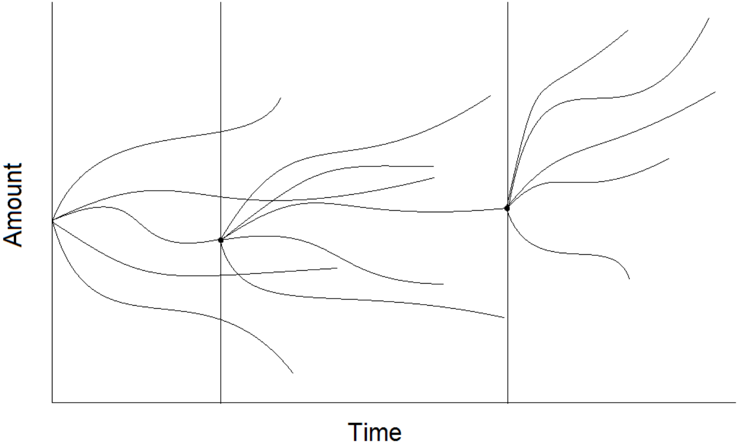
Adaptive Time Step



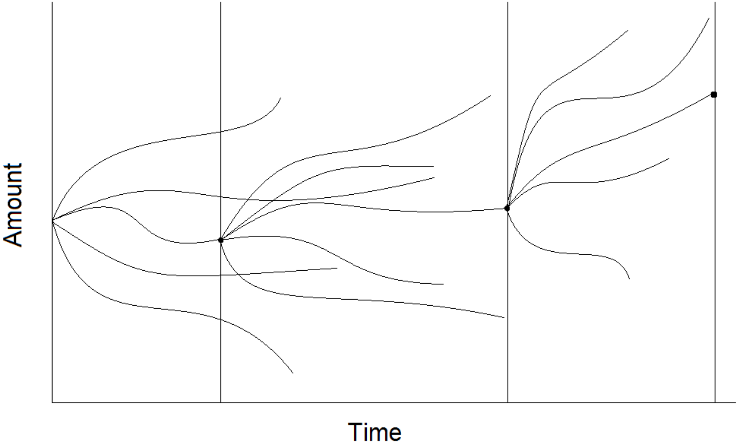
Adaptive Time Step



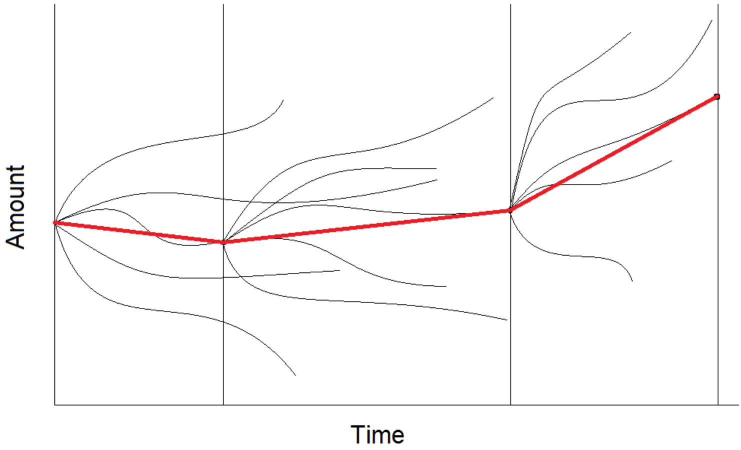
Adaptive Time Step



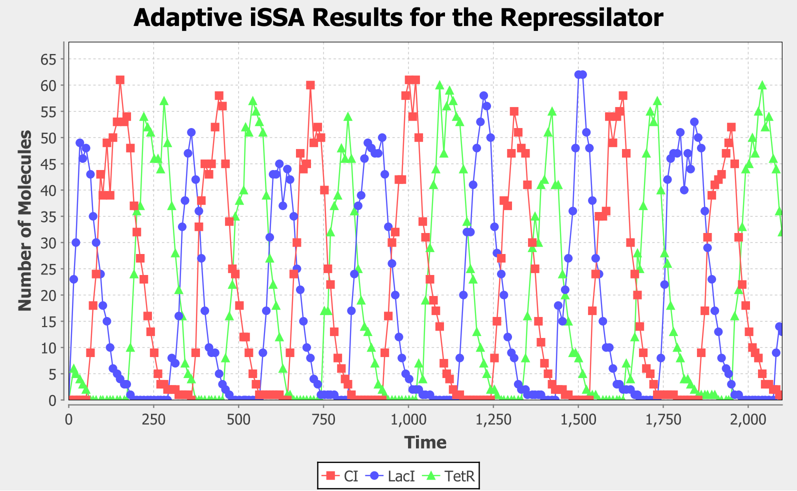
Adaptive Time Step



Adaptive Time Step



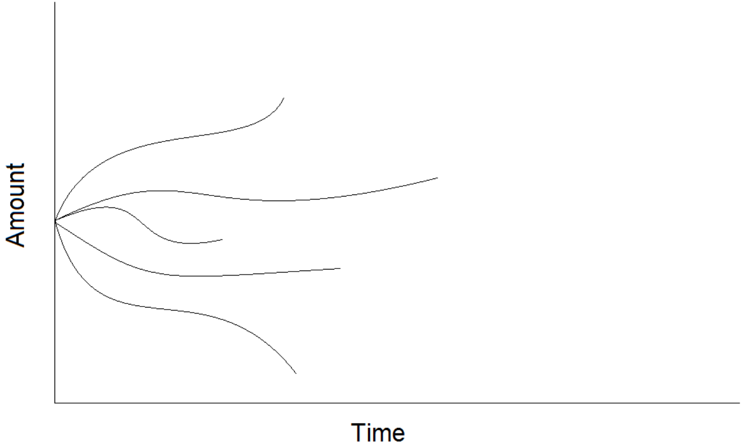
Repressilator: Adaptive iSSA Simulation



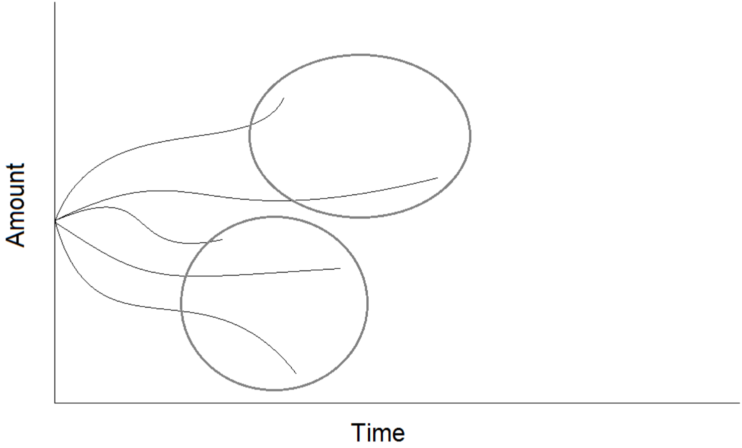
Multiple Paths

- Many systems have more than one typical behavior.
- Modify the selection process to use the k-means clustering algorithm to select starting states that are closest to the average of each grouping.
- The likelihood of each path is determined by how many states end up in each cluster.

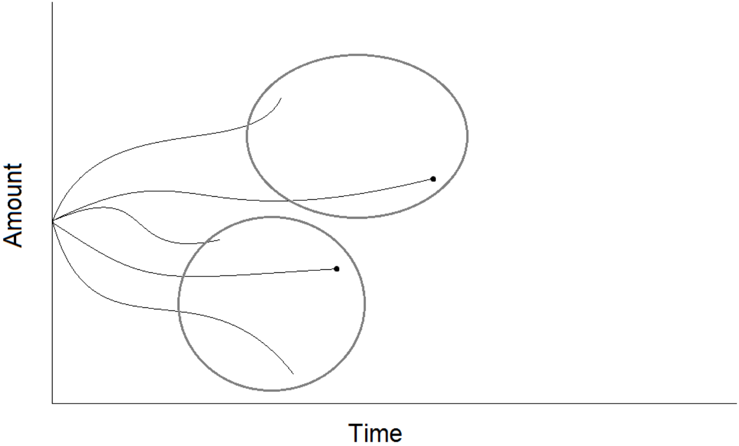
Multiple Paths



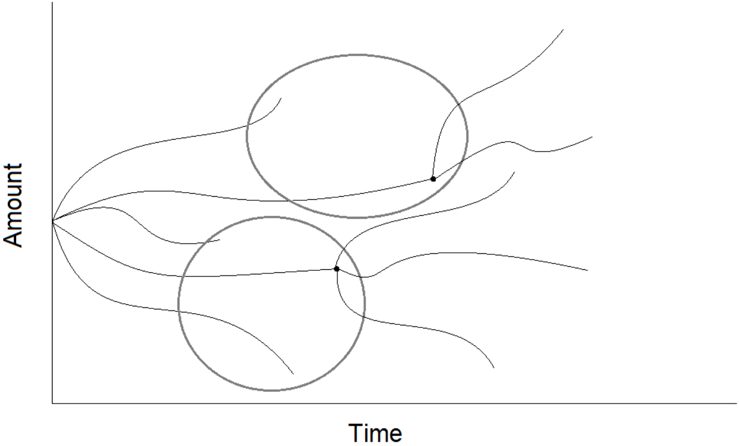
Multiple Paths



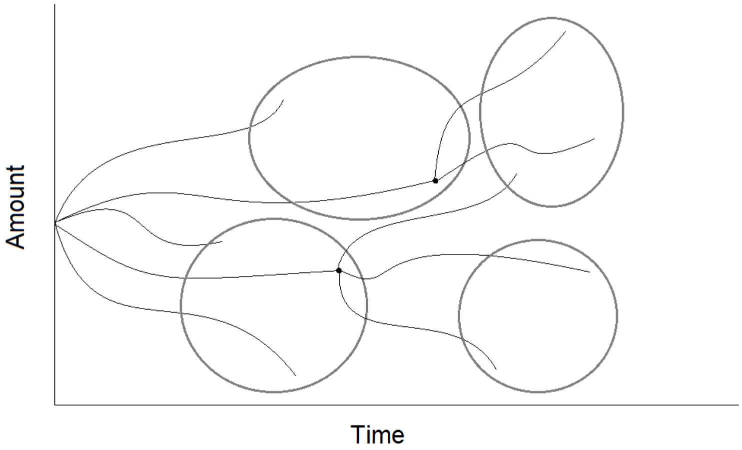
Multiple Paths



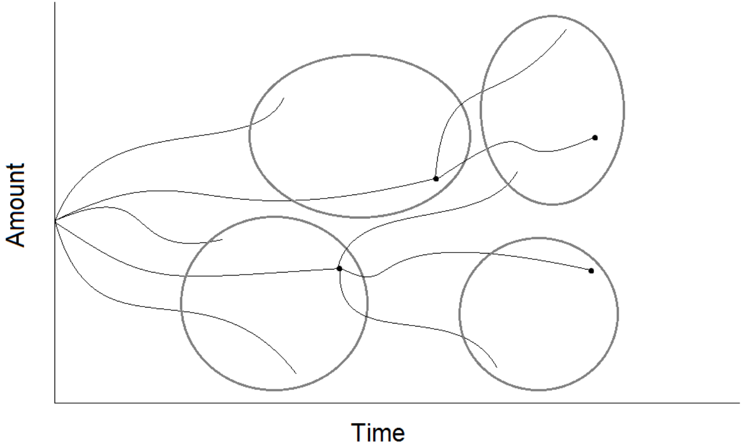
Multiple Paths



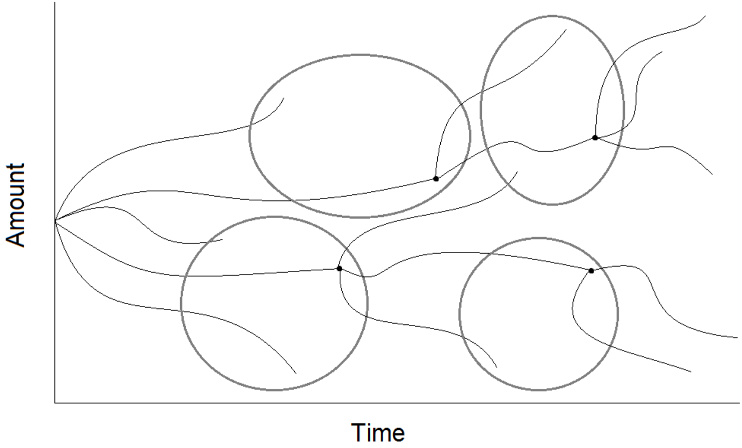
Multiple Paths



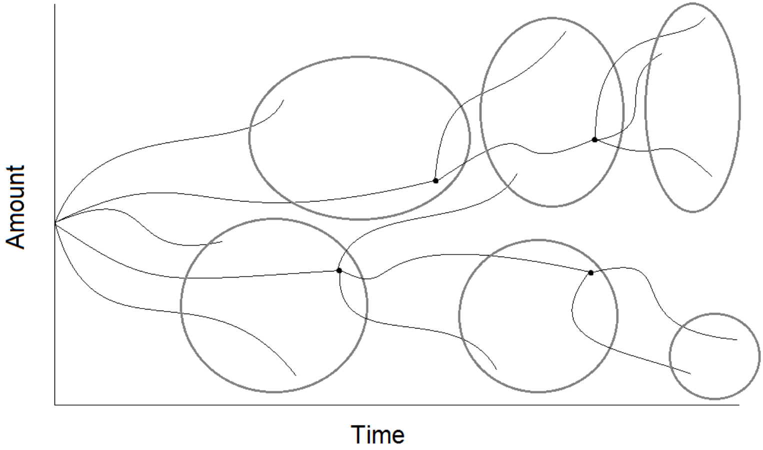
Multiple Paths



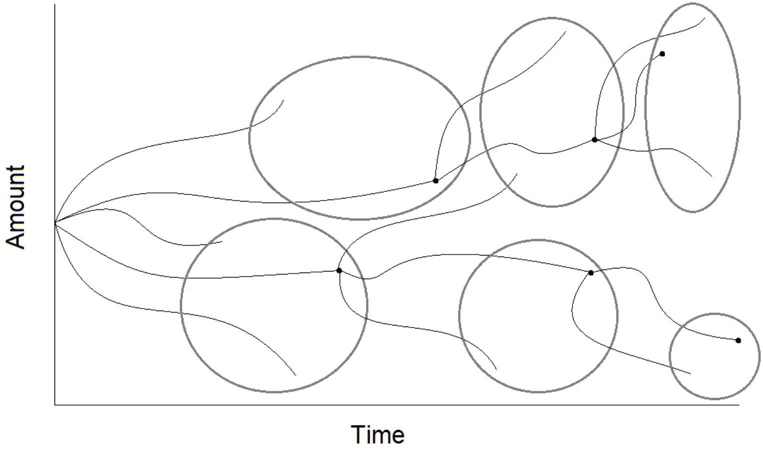
Multiple Paths



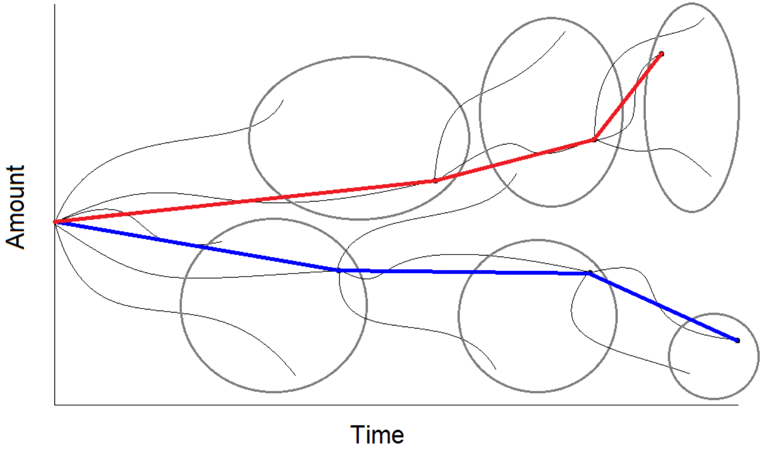
Multiple Paths



Multiple Paths



Multiple Paths



Summary

- hSSA is useful for hierarchical models, such as, cellular populations.
- wSSA is useful for analysis of rare events.
- iSSA is useful for extracting typical behavior.

Sources

- hSSA - Watanabe and Myers (2014).
- Arrays - Watanabe and Myers (2016).
- wSSA - Kuwahara and Mura (2008).
- iSSA - Winstead, Madsen et al. (2010).