Continuous-space analysis of Bio-PEPA models

Stephen Gilmore
LFCS, University of Edinburgh

PEPA Club

August 22, 2008

Outline

1. Analysis with Bio-PEPA
2. Discrete-state analysis
   - Discrete state-space
   - Transient probability distribution
3. Continuous-space analysis
   - Differential equations and the Jacobian
   - Differential equation analysis

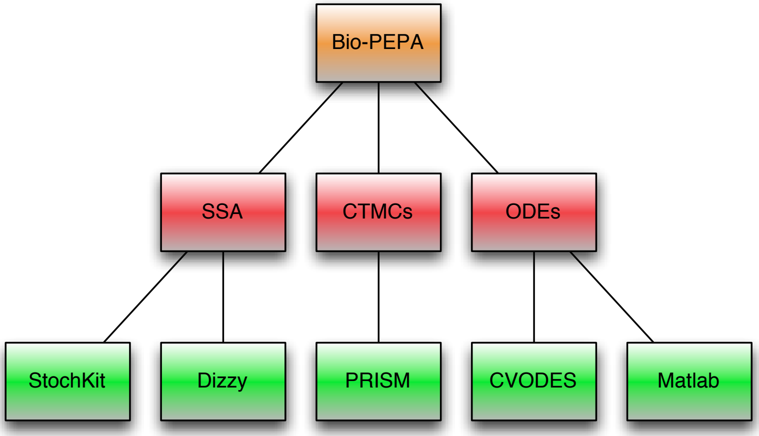
Outline

1. Analysis with Bio-PEPA

2. Discrete-state analysis
   - Discrete state-space
   - Transient probability distribution

3. Continuous-space analysis
   - Differential equations and the Jacobian
   - Differential equation analysis

Analysis with Bio-PEPA



Enzyme-Substrate example

Consider the simple Enzyme-Substrate reaction

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} E{:}S \overset{k_2}{\longrightarrow} E + P$$

Formulation in Bio-PEPA

The kinetic functions

r_1	$k_1 \times E \times S$
r_{-1}	$k_{-1} \times E{:}S$
r_2	$k_2 \times E{:}S$

The Bio-PEPA model

E	$r_1 \downarrow$	$+$	$r_{-1} \uparrow$	$+$	$r_2 \uparrow$
S	$r_1 \downarrow$	$+$	$r_{-1} \uparrow$		
$E{:}S$	$r_1 \uparrow$	$+$	$r_{-1} \downarrow$	$+$	$r_2 \downarrow$
P					$r_2 \uparrow$

Outline

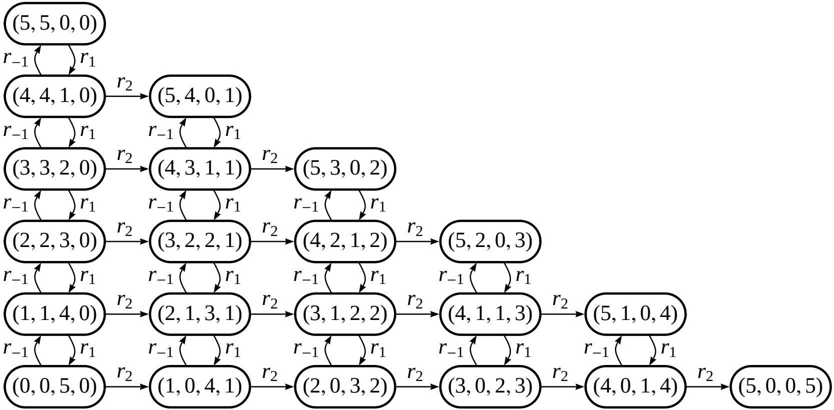
1. Analysis with Bio-PEPA
2. Discrete-state analysis
   - Discrete state-space
   - Transient probability distribution
3. Continuous-space analysis
   - Differential equations and the Jacobian
   - Differential equation analysis

Discrete state-space of Enzyme-Substrate example

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} E{:}S \overset{k_2}{\longrightarrow} E + P$$

- In the discrete stochastic domain we can consider the state-space generated from initial numbers of molecules. E.g. $(E, S, E{:}S, P) = (5, 5, 0, 0)$.
- We consider the effect of each of the three reactions on the four molecule counts E, S, $E{:}S$, P.

Discrete state-space of Enzyme-Substrate example

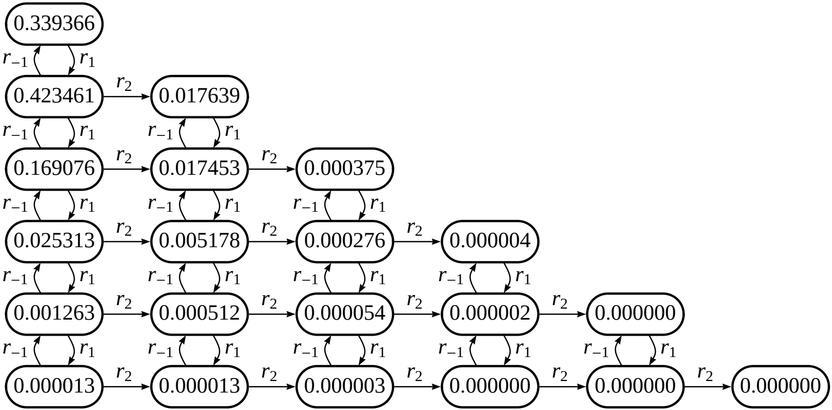


Probability distribution across the state-space

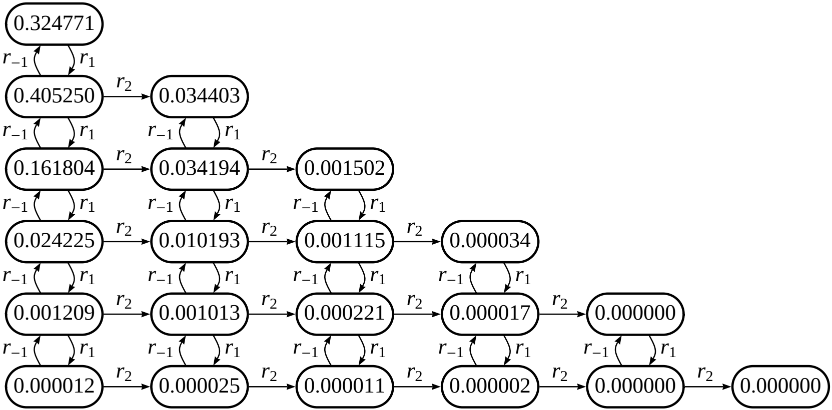
$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} E{:}S \overset{k_2}{\longrightarrow} E + P$$

- If we know the initial molecule counts and the values of the rate constants $k_1 = 1.0$, $k_{-1} = 20.0$ and $k_2 = 0.05$ we can compute the probability of being in each state of the state-space at all future time points.
- At time $t = 0$ we have $\Pr(5, 5, 0, 0) = 1$.

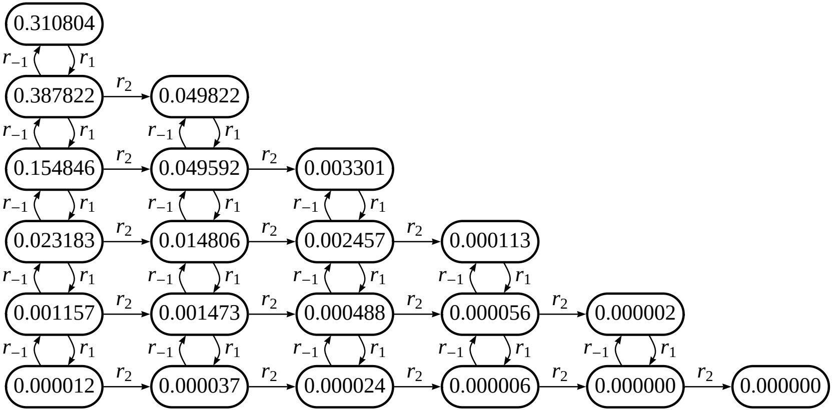
Transient probability distribution at $t = 1$



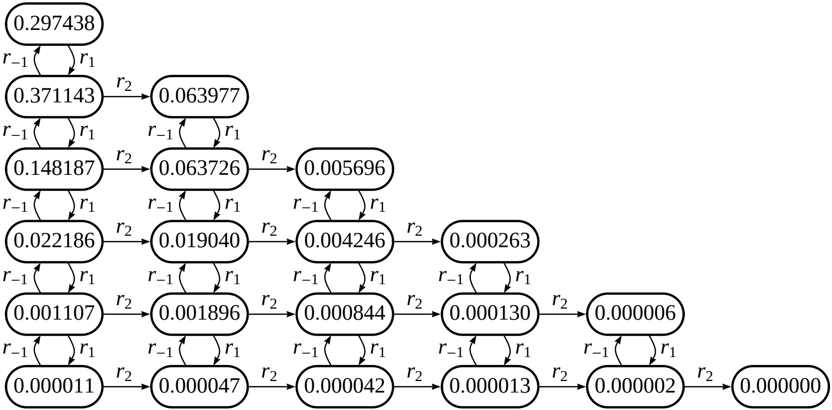
Transient probability distribution at $t = 2$



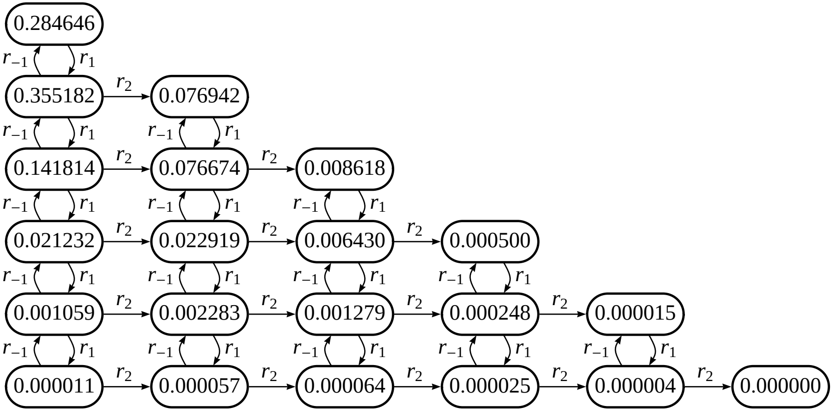
Transient probability distribution at $t = 3$



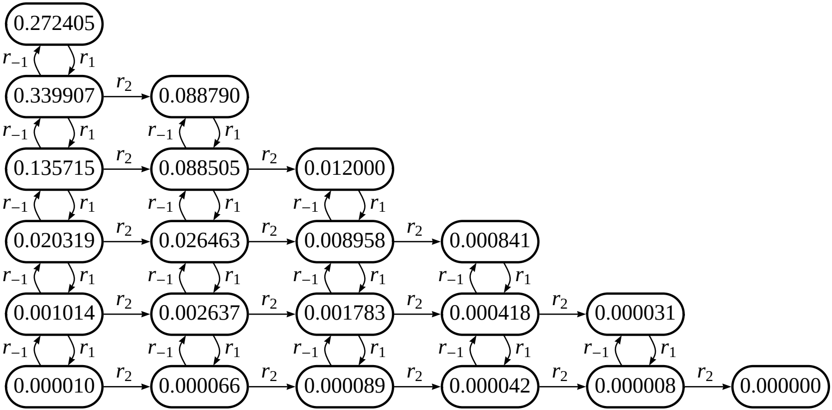
Transient probability distribution at $t = 4$



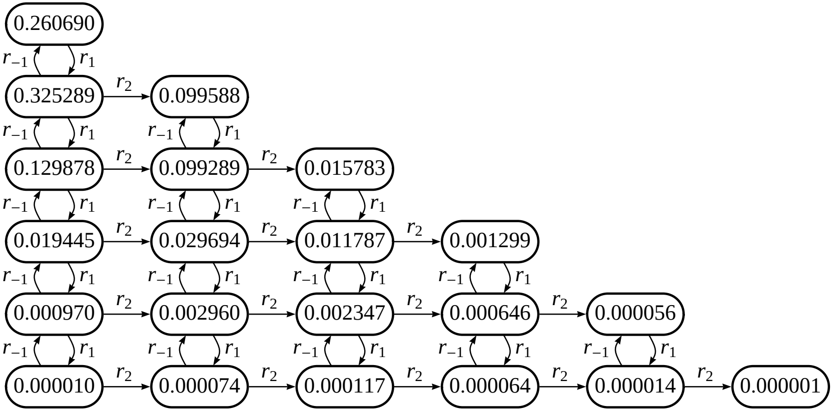
Transient probability distribution at $t = 5$



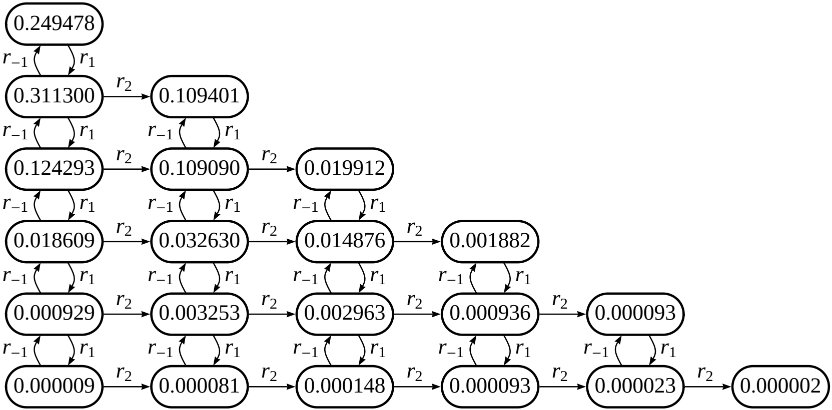
Transient probability distribution at $t = 6$



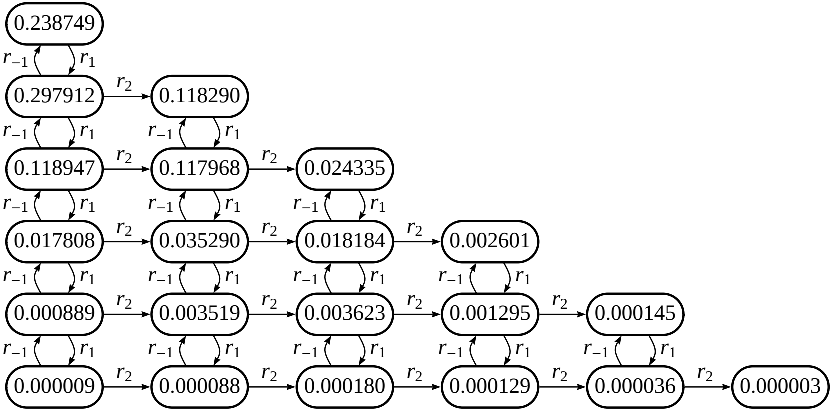
Transient probability distribution at $t = 7$



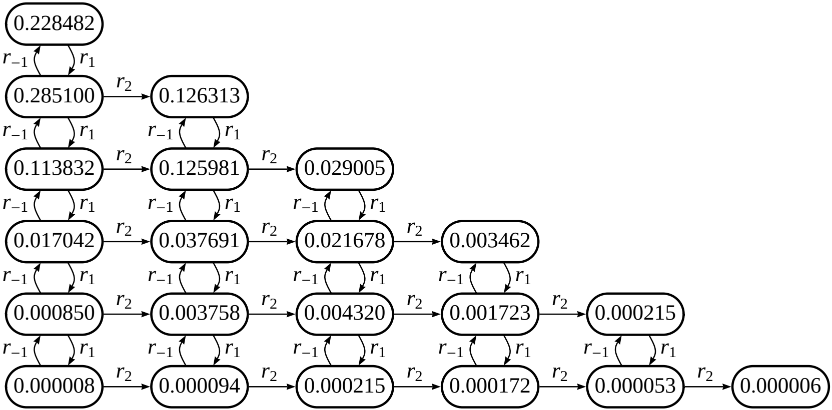
Transient probability distribution at $t = 8$



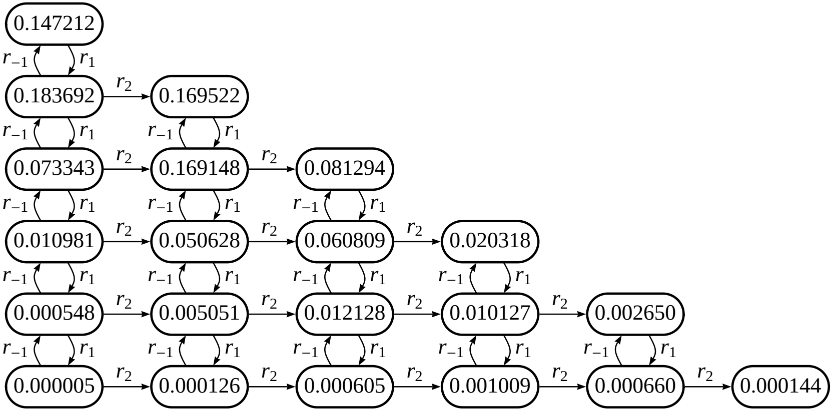
Transient probability distribution at $t = 9$



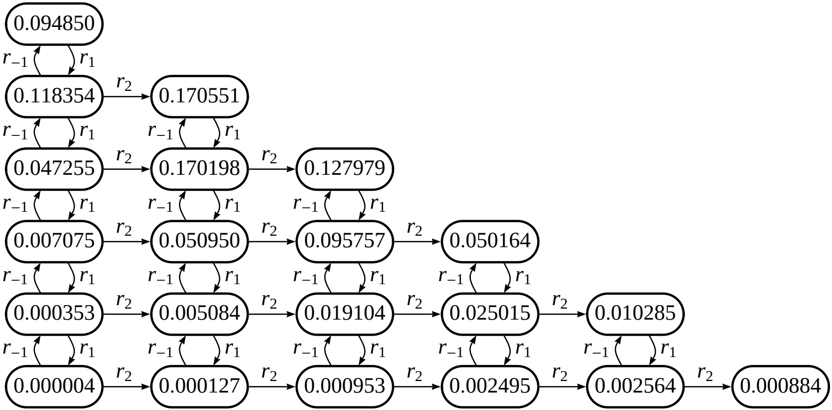
Transient probability distribution at $t = 10$



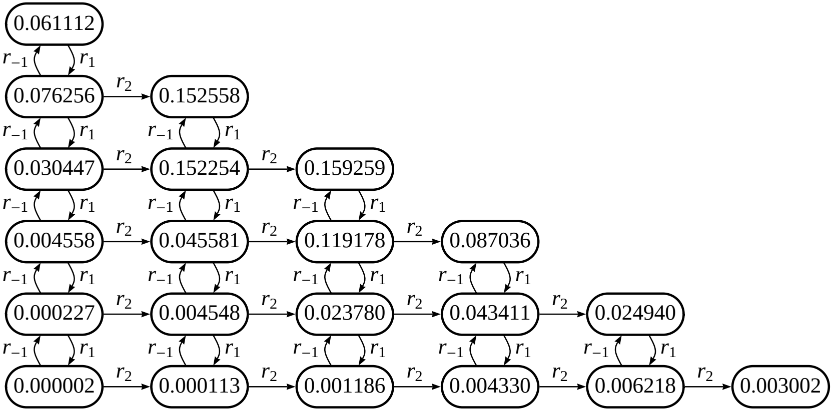
Transient probability distribution at $t = 20$



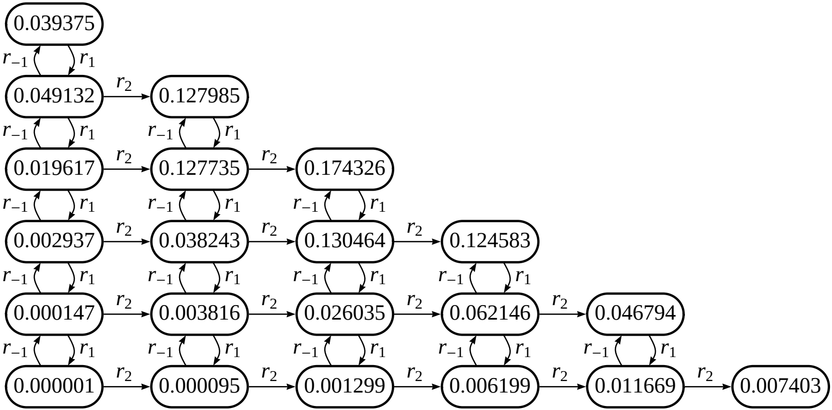
Transient probability distribution at $t = 30$



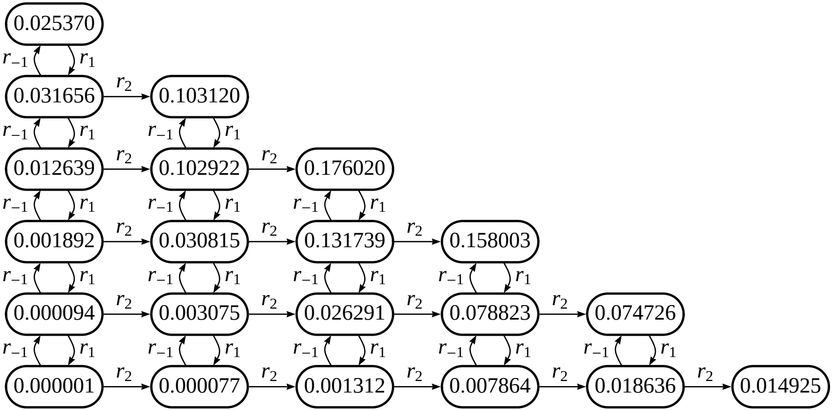
Transient probability distribution at $t = 40$



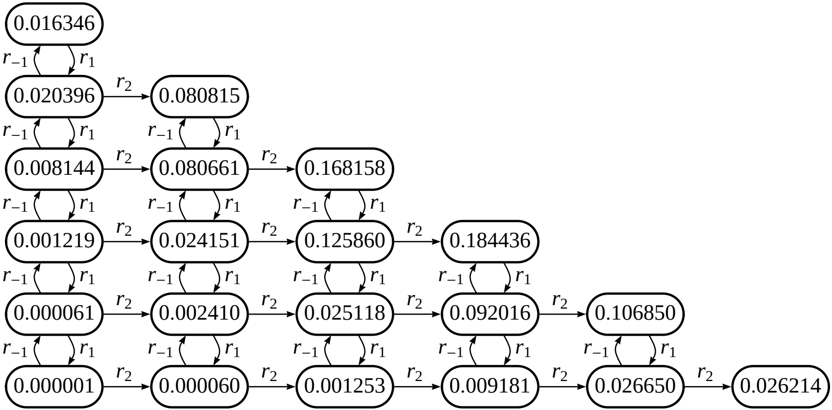
Transient probability distribution at $t = 50$



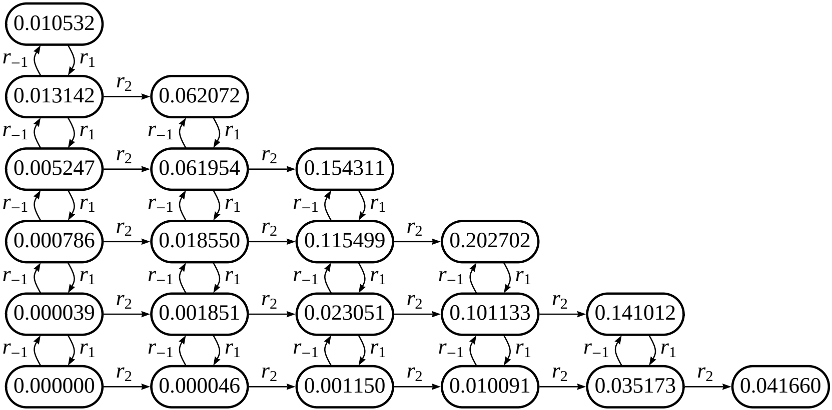
Transient probability distribution at $t = 60$



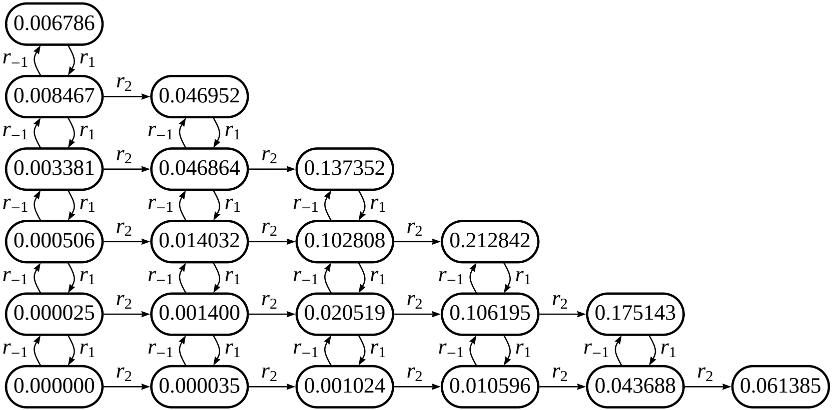
Transient probability distribution at $t = 70$



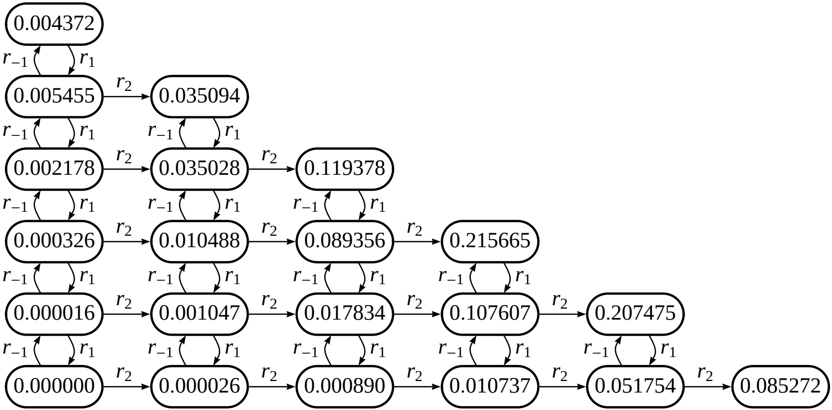
Transient probability distribution at $t = 80$



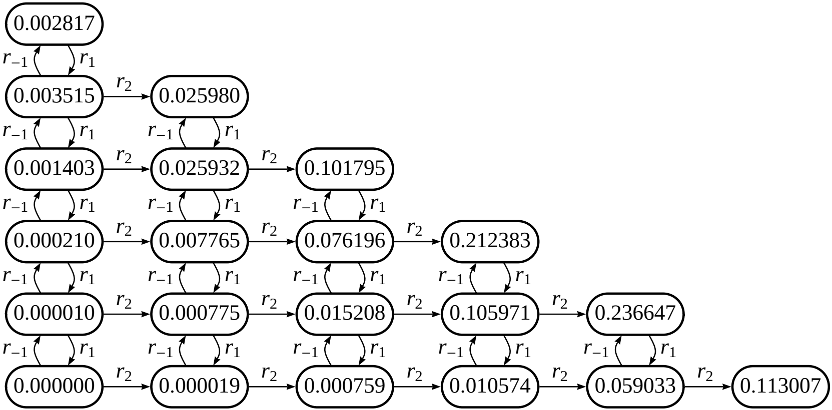
Transient probability distribution at $t = 90$



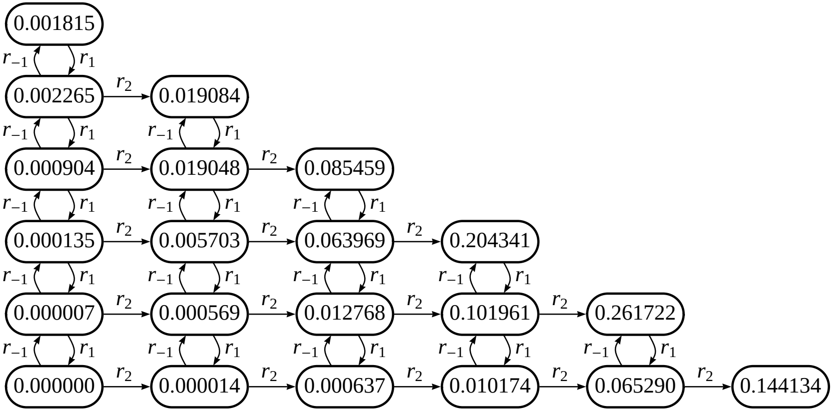
Transient probability distribution at $t = 100$



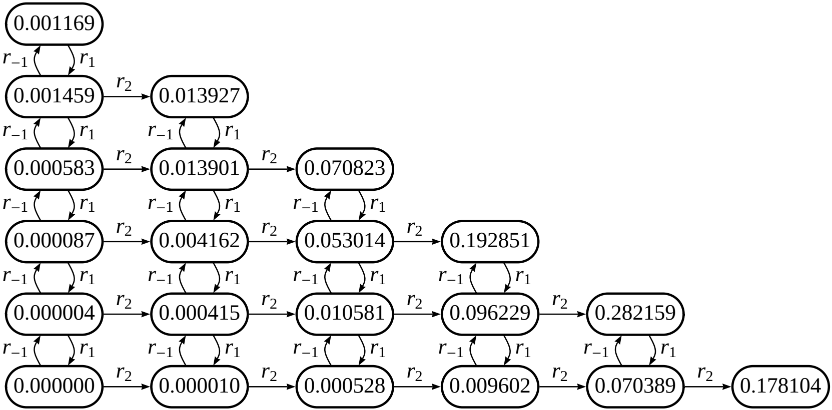
Transient probability distribution at $t = 110$



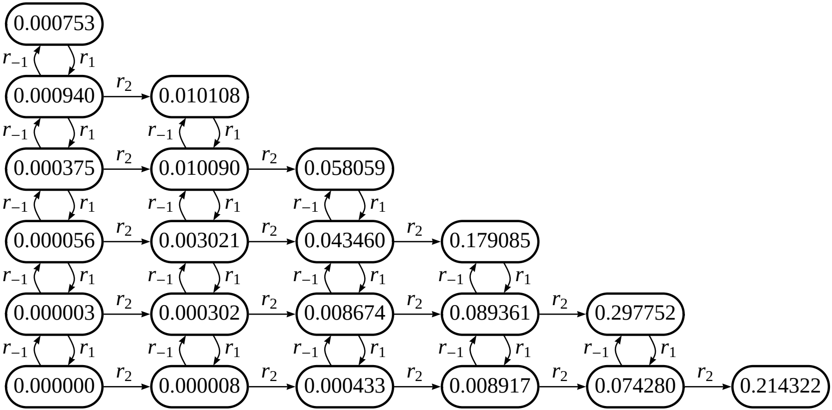
Transient probability distribution at $t = 120$



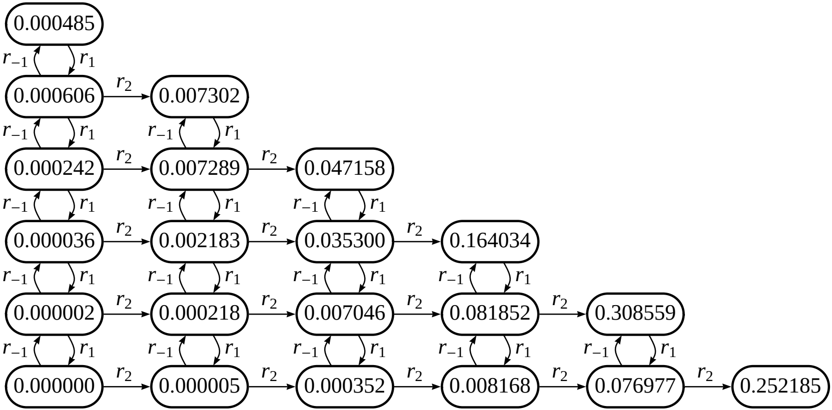
Transient probability distribution at $t = 130$



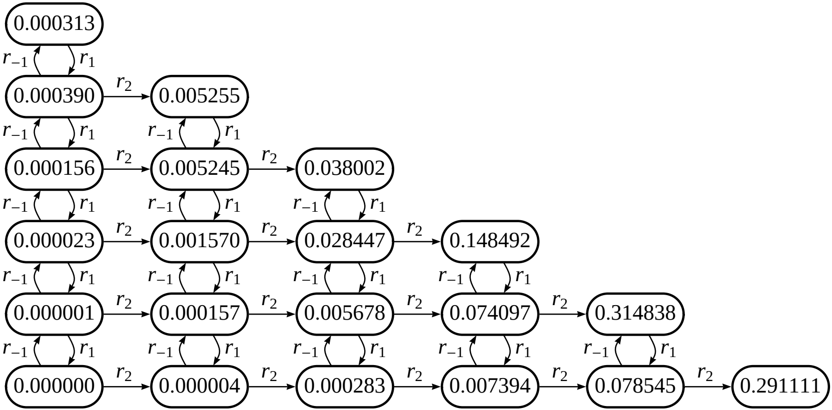
Transient probability distribution at $t = 140$



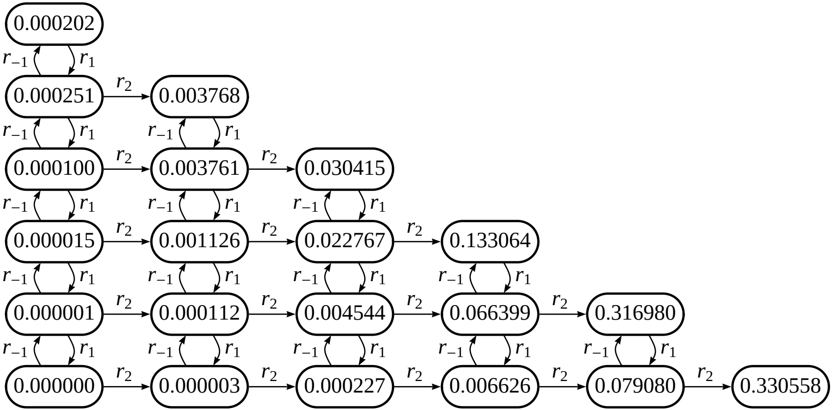
Transient probability distribution at $t = 150$



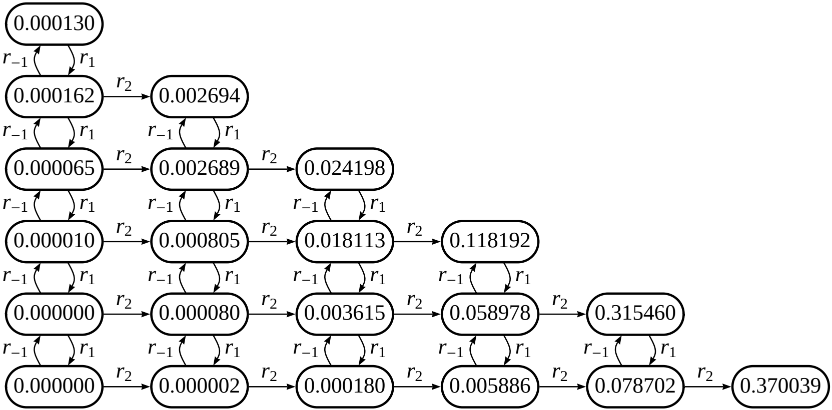
Transient probability distribution at $t = 160$



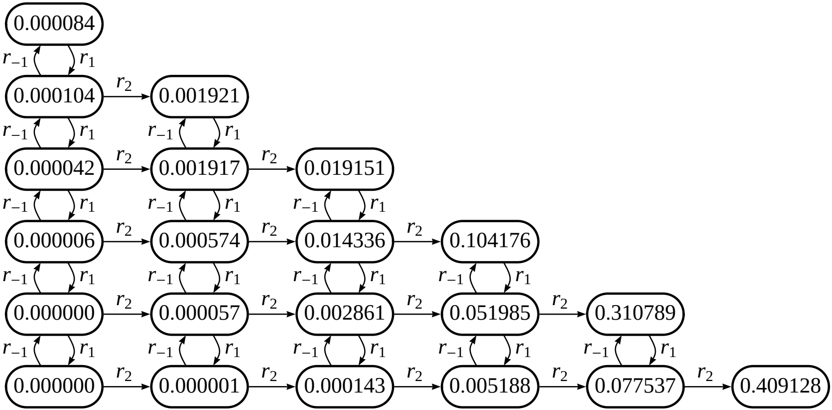
Transient probability distribution at $t = 170$



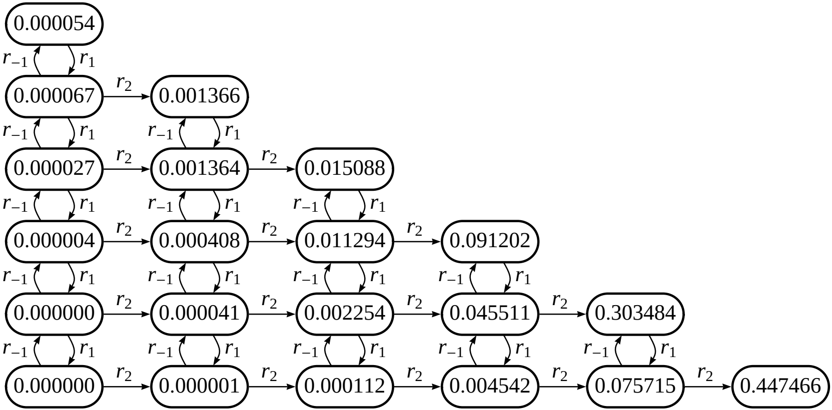
Transient probability distribution at $t = 180$



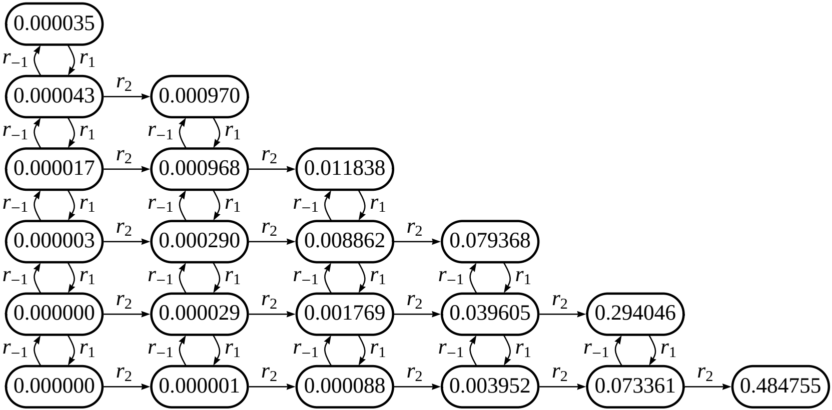
Transient probability distribution at $t = 190$



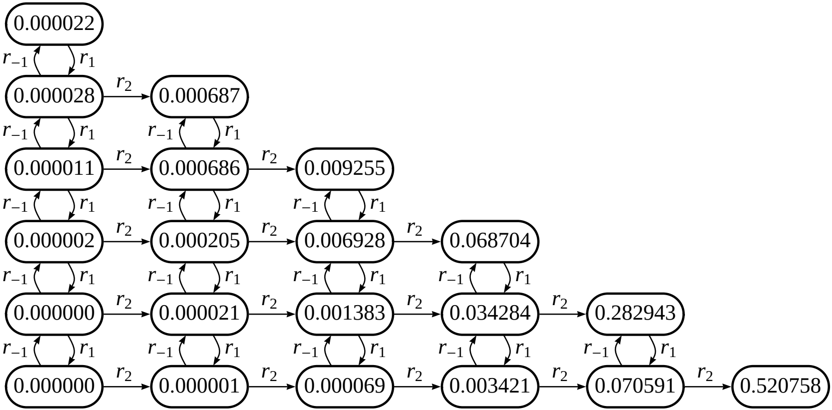
Transient probability distribution at $t = 200$



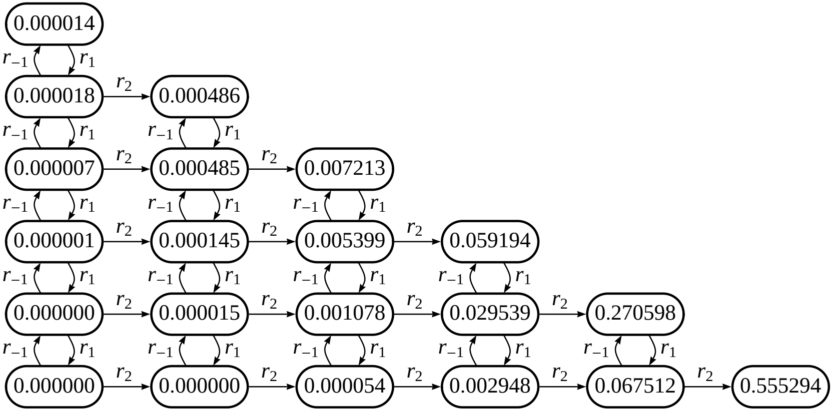
Transient probability distribution at $t = 210$



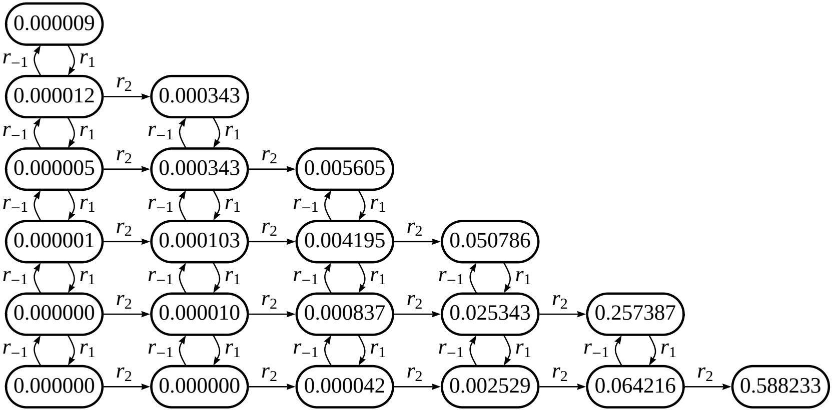
Transient probability distribution at $t = 220$



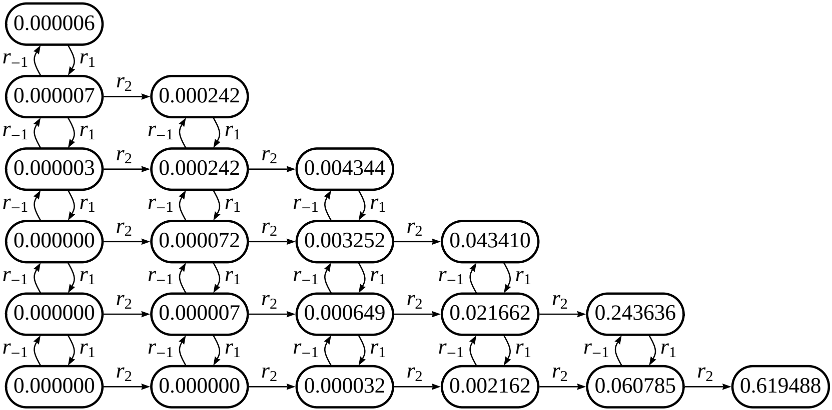
Transient probability distribution at $t = 230$



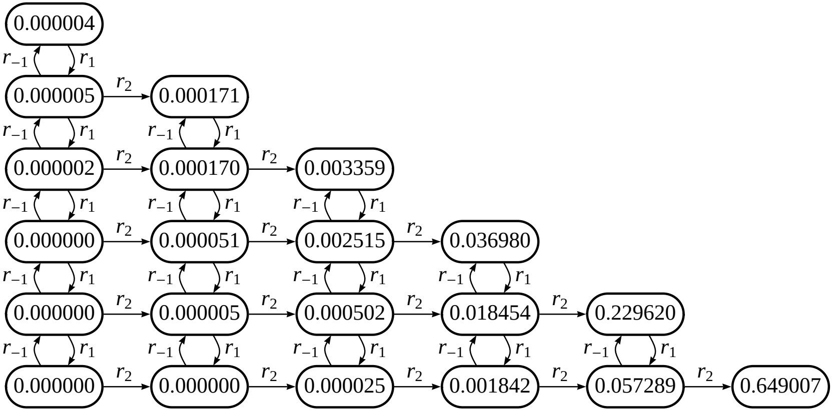
Transient probability distribution at $t = 240$



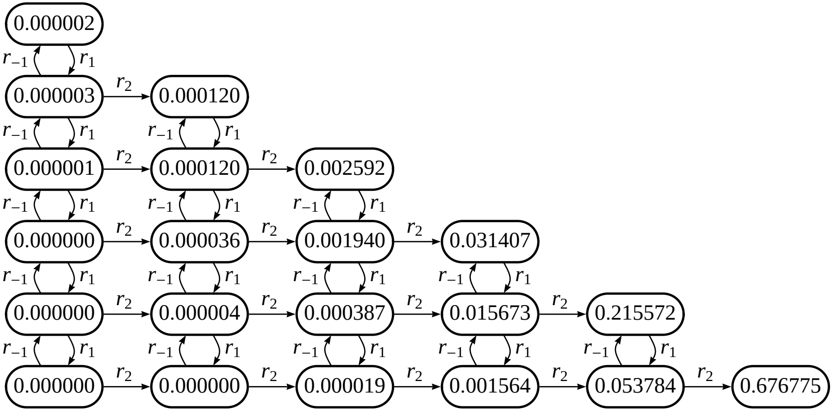
Transient probability distribution at $t = 250$



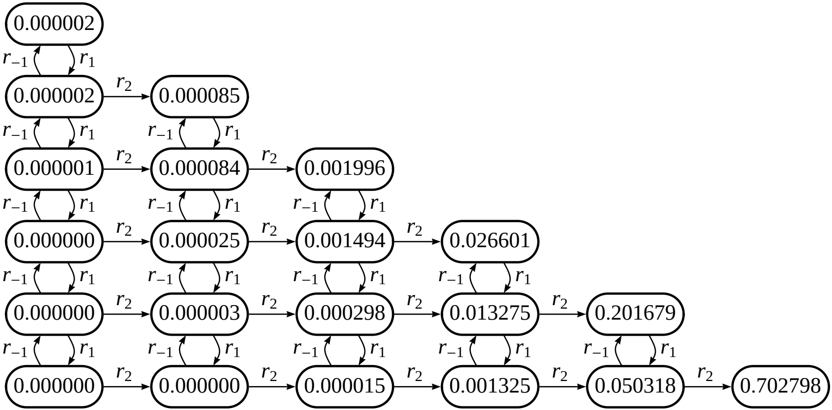
Transient probability distribution at $t = 260$



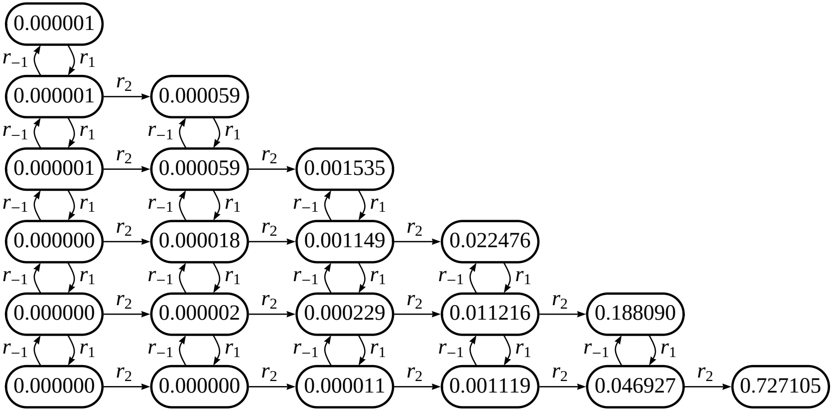
Transient probability distribution at $t = 270$



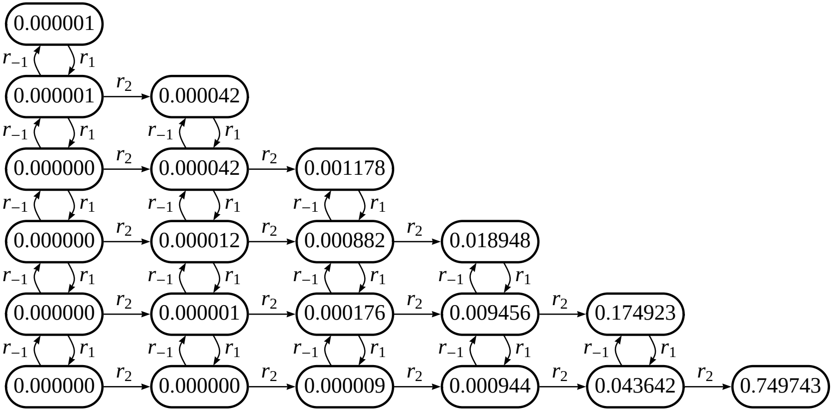
Transient probability distribution at $t = 280$



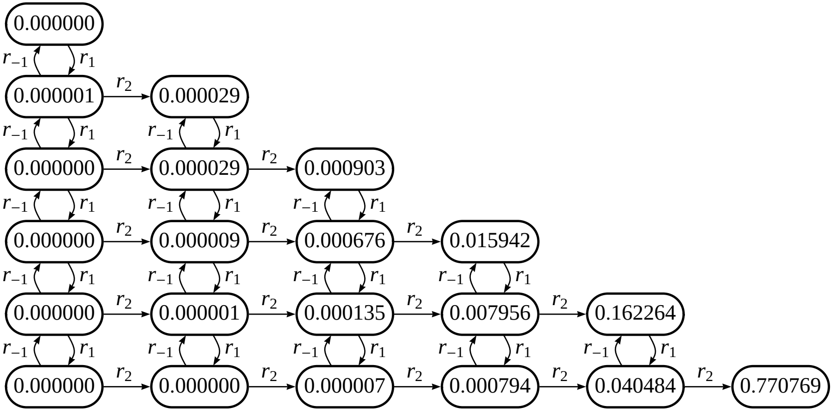
Transient probability distribution at $t = 290$



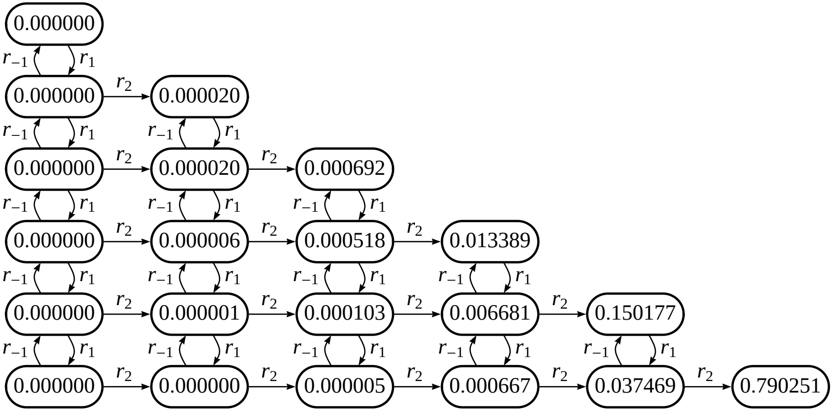
Transient probability distribution at $t = 300$



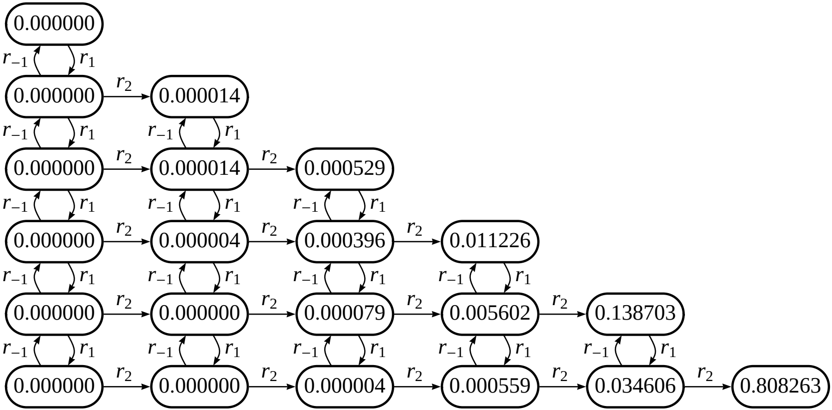
Transient probability distribution at $t = 310$



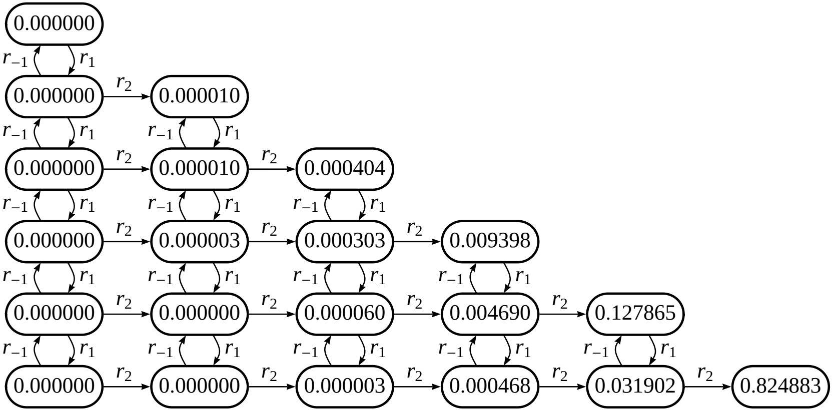
Transient probability distribution at $t = 320$



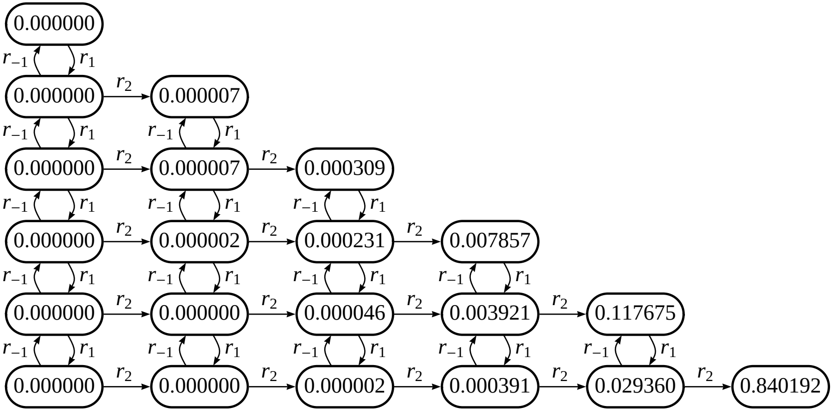
Transient probability distribution at $t = 330$



Transient probability distribution at $t = 340$



Transient probability distribution at $t = 350$



Outline

1. Analysis with Bio-PEPA

2. Discrete-state analysis
   - Discrete state-space
   - Transient probability distribution

3. Continuous-space analysis
   - Differential equations and the Jacobian
   - Differential equation analysis

The kinetic functions and the Bio-PEPA model

The kinetic functions

r_1	$k_1 \times E \times S$
r_{-1}	$k_{-1} \times E{:}S$
r_2	$k_2 \times E{:}S$

The Bio-PEPA model

E	$r_1 \downarrow$	$+$	$r_{-1} \uparrow$	$+$	$r_2 \uparrow$
S	$r_1 \downarrow$	$+$	$r_{-1} \uparrow$		
$E{:}S$	$r_1 \uparrow$	$+$	$r_{-1} \downarrow$	$+$	$r_2 \downarrow$
P					$r_2 \uparrow$

The Bio-PEPA model and the differential equations

The Bio-PEPA model

E	$r_1 \downarrow$	$+$	$r_{-1} \uparrow$	$+$	$r_2 \uparrow$
S	$r_1 \downarrow$	$+$	$r_{-1} \uparrow$		
$E{:}S$	$r_1 \uparrow$	$+$	$r_{-1} \downarrow$	$+$	$r_2 \downarrow$
P					$r_2 \uparrow$

The differential equations

dE/dt	$-r_1$	$+$	r_{-1}	$+$	r_2
dS/dt	$-r_1$	$+$	r_{-1}		
$dE{:}S/dt$	r_1	$-$	r_{-1}	$-$	r_2
dP/dt					r_2

The differential equations and the Jacobian

The differential equations

dE/dt	$-k_1 \times E \times S + k_{-1} \times E{:}S + k_2 \times E{:}S$
dS/dt	$-k_1 \times E \times S + k_{-1} \times E{:}S$
$dE{:}S/dt$	$k_1 \times E \times S - k_{-1} \times E{:}S - k_2 \times E{:}S$
dP/dt	$k_2 \times E{:}S$

The Jacobian

	E	S	$E{:}S$	P
E	$\partial f_E/\partial E$	$\partial f_E/\partial S$	$\partial f_E/\partial E{:}S$	$\partial f_E/\partial P$
S	$\partial f_S/\partial E$	$\partial f_S/\partial S$	$\partial f_S/\partial E{:}S$	$\partial f_S/\partial P$
$E{:}S$	$\partial f_{E:S}/\partial E$	$\partial f_{E:S}/\partial S$	$\partial f_{E:S}/\partial E{:}S$	$\partial f_{E:S}/\partial P$
P	$\partial f_P/\partial E$	$\partial f_P/\partial S$	$\partial f_P/\partial E{:}S$	$\partial f_P/\partial P$

The differential equations and the Jacobian

The differential equations

dE/dt	$-k_1 \times E \times S + k_{-1} \times E{:}S + k_2 \times E{:}S$
dS/dt	$-k_1 \times E \times S + k_{-1} \times E{:}S$
$dE{:}S/dt$	$k_1 \times E \times S - k_{-1} \times E{:}S - k_2 \times E{:}S$
dP/dt	$k_2 \times E{:}S$

The Jacobian

	E	S	$E{:}S$	P
E	$-k_1 \times S$	$-k_1 \times E$	$k_{-1} + k_2$	
S	$-k_1 \times S$	$-k_1 \times E$	k_{-1}	
$E{:}S$	$k_1 \times S$	$k_1 \times E$	$-k_{-1} - k_2$	
P			k_2	

Obtaining and using the Jacobian

- The Jacobian (and Hessian and higher derivatives) are computed automatically from the differential equations using *symbolic differentiation*.

Obtaining and using the Jacobian

- The Jacobian (and Hessian and higher derivatives) are computed automatically from the differential equations using *symbolic differentiation*.
- ODE solvers generally use *finite differences* to approximate the Jacobian matrix if it is not supplied, but an implementation of the analytically derived Jacobian can improve the speed, accuracy and reliability of the program.

Obtaining and using the Jacobian

- The Jacobian (and Hessian and higher derivatives) are computed automatically from the differential equations using *symbolic differentiation*.
- ODE solvers generally use *finite differences* to approximate the Jacobian matrix if it is not supplied, but an implementation of the analytically derived Jacobian can improve the speed, accuracy and reliability of the program.
- Programs that compute *bifurcations* will use the Hessian and higher derivatives.

Differential equation analysis

Several different differential equation solvers exist.

- SUNDIALS — ODE integrators in C

Differential equation analysis

Several different differential equation solvers exist.

- SUNDIALS — ODE integrators in C
- Matlab — numerical computing platform

Differential equation analysis

Several different differential equation solvers exist.

- SUNDIALS — ODE integrators in C
- Matlab — numerical computing platform
- MatCont — Matlab toolbox for continuation analysis

Differential equation analysis

Several different differential equation solvers exist.

- SUNDIALS — ODE integrators in C
- Matlab — numerical computing platform
- MatCont — Matlab toolbox for continuation analysis
- AUTO — C and Fortran package for numerical continuation

Differential equation analysis

Several different differential equation solvers exist.

- SUNDIALS — ODE integrators in C
- Matlab — numerical computing platform
- MatCont — Matlab toolbox for continuation analysis
- AUTO — C and Fortran package for numerical continuation

Differential equation analysis

Several different differential equation solvers exist.

- SUNDIALS — ODE integrators in C
- Matlab — numerical computing platform
- MatCont — Matlab toolbox for continuation analysis
- AUTO — C and Fortran package for numerical continuation

Different formats and languages for problem description.

VFGEN: A Vector Field File Generator

- VFGEN is a vector field file generator for differential equation solvers and other computational tools.

VFGEN: A Vector Field File Generator

- VFGEN is a vector field file generator for differential equation solvers and other computational tools.
- VFGEN lets you define your vector field once (using XML), and export the vector field in several formats.

VFGEN: A Vector Field File Generator

- VFGEN is a vector field file generator for differential equation solvers and other computational tools.
- VFGEN lets you define your vector field once (using XML), and export the vector field in several formats.
- VFGEN uses a C++ symbolic algebra library (GiNaC) to generate Jacobians and higher derivatives automatically.

VFGEN: A Vector Field File Generator

- VFGEN is a vector field file generator for differential equation solvers and other computational tools.
- VFGEN lets you define your vector field once (using XML), and export the vector field in several formats.
- VFGEN uses a C++ symbolic algebra library (GiNaC) to generate Jacobians and higher derivatives automatically.

VFGEN: A Vector Field File Generator

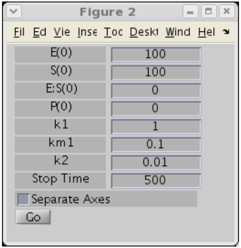
- VFGEN is a vector field file generator for differential equation solvers and other computational tools.
- VFGEN lets you define your vector field once (using XML), and export the vector field in several formats.
- VFGEN uses a C++ symbolic algebra library (GiNaC) to generate Jacobians and higher derivatives automatically.

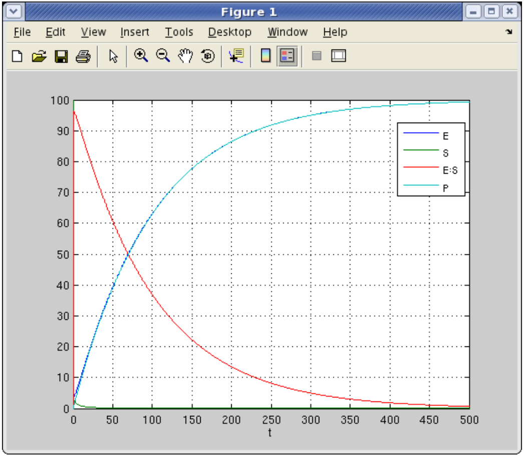
Warren Weckesser.
VFGEN: A Code Generation Tool.
Journal of Numerical Analysis, Industrial and Applied Mathematics, Volume 3(1-2):151–165, 2008.

VFgen representation of the Enzyme-Substrate model

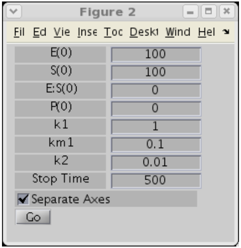
```
<?xml version="1.0"?>
<!-- VFgen model compiled from Bio-PEPA input file "mm" by
 Bio-PEPA Workbench Version 0.9.8 "Hillel Slovak" [20-June-2008] -->
<VectorField Name="mm001">

<Parameter Name="k1_" Description="k1" Latex="k_1" DefaultValue="1"/>
<Parameter Name="km1_" Description="km1" Latex="k_{-1}" DefaultValue="0.1"/>
<Parameter Name="k2_" Description="k2" Latex="k_2" DefaultValue="0.01"/>

<Expression Name="r1_" Description="r1" Latex="r_1" Formula=" k1_ * E_ * S_ "/>
<Expression Name="rm1_" Description="rm1" Latex="r_{-1}" Formula=" km1_ * E_colon_S_ "/>
<Expression Name="r2_" Description="r2" Latex="r_2" Formula=" k2_ * E_colon_S_ "/>

<StateVariable Name="E_" Description="E" Latex="E" DefaultInitialCondition="100"
   Formula=" - r1_ + rm1_ + r2_"/>

<StateVariable Name="S_" Description="S" Latex="S" DefaultInitialCondition="100"
   Formula=" - r1_ + rm1_"/>

<StateVariable Name="E_colon_S_" Description="E:S" Latex="\hbox{\textit{E:S}}" DefaultInitialCondition="0"
   Formula="r1_ - rm1_ - r2_"/>

<StateVariable Name="P_" Description="P" Latex="P" DefaultInitialCondition="0"
   Formula="r2_"/>

<!-- End VFgen model compiled from mm -->
</VectorField>
```

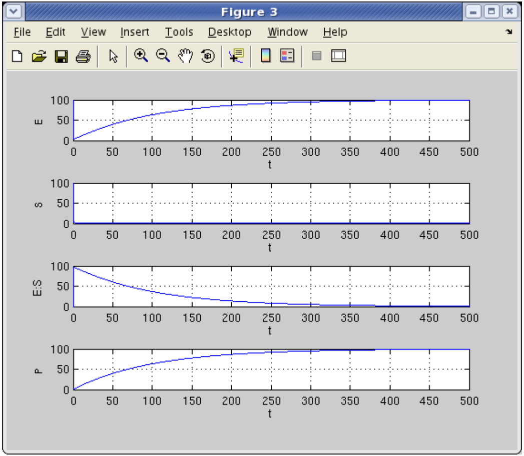
Analysing Bio-PEPA models with Matlab





Analysing Bio-PEPA models with Matlab





Conclusions

- We now have some support for more general analysis of ODE models generated from Bio-PEPA descriptions.

Conclusions

- We now have some support for more general analysis of ODE models generated from Bio-PEPA descriptions.
- We can now perform bifurcation analysis and continuation analysis on Bio-PEPA models.

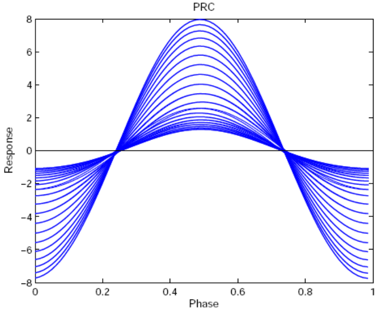
Conclusions

- We now have some support for more general analysis of ODE models generated from Bio-PEPA descriptions.
- We can now perform bifurcation analysis and continuation analysis on Bio-PEPA models.
  - Useful for studying systems which oscillate.

Conclusions

- We now have some support for more general analysis of ODE models generated from Bio-PEPA descriptions.
- We can now perform bifurcation analysis and continuation analysis on Bio-PEPA models.
  - Useful for studying systems which oscillate.
- Can compute *phase response curves* ▸ Defn

Phase response curve

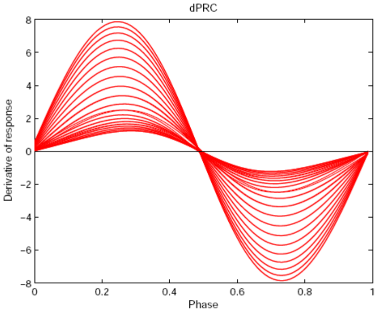


A. Dhooge, W. Govaerts, Yu.A. Kuznetsov, W. Mestrom, A.M. Riet and B. Sautois

MATCONT and CL_MATCONT: Continuation toolboxes in Matlab

December 2006.

Derivative of the phase response curve



A. Dhooge, W. Govaerts, Yu.A. Kuznetsov, W. Mestrom, A.M. Riet and B. Sautois

MATCONT and CL_MATCONT: Continuation toolboxes in Matlab

December 2006.

Phase response curve

The phase response curve of a limit cycle, or PRC, is a curve, defined over the period of the cycle, that expresses, at each time of that period, the effect of a small input vector on the cycle. In experimental circumstances, this may correspond to injected current, to the addition of more chemical agents, etc. A positive value means that the current cycle is shortened in time, a negative value means that the period is prolonged.

A. Dhooge, W. Govaerts, Yu.A. Kuznetsov, W. Mestrom, A.M. Riet and B. Sautois
MATCONT and CL_MATCONT: Continuation toolboxes in Matlab
December 2006.