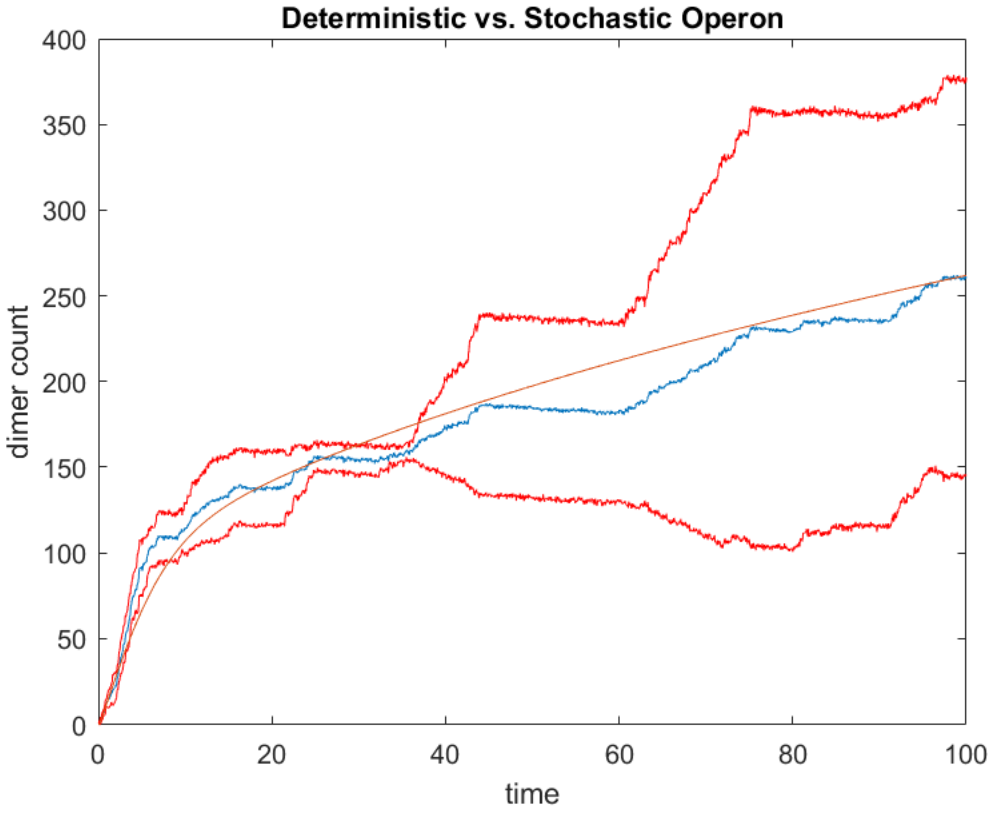
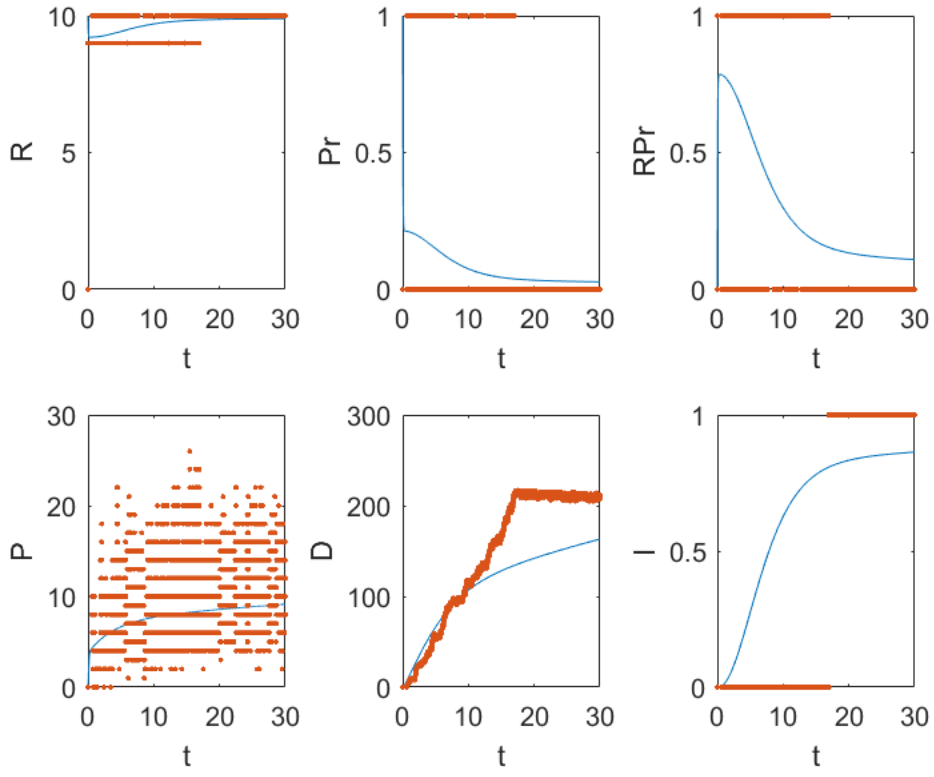
```
% Minjun Park, CAAM 210, 10/19/15
%
% This coding compares Deterministic and stochastic operon by plotting
 Dimer, and also subplots of other elements.
% It use Ordinary differential equations to compare these different
 operons
%
% USAGE: Choose what element to compare, and in this case we compare
 Dimer.
% Also, write down equations to see how they react, considering
 propensity
%
%
function mca2driver
rtab = {[1 -1 2 -1 3 1]%R + Pr -> RPr
    [3 -1 1 1 2 1] %RPr -> R + PR
    [3 -1 4 10 1 1 2 1]%RPr -> 10P + R + Pr
    [4 -2 5 1]%P + P -> D
    [5 -1 4 2]%D -> P + P
    [4 -1]%P ->
    [2 -1 5 -1 6 1]%Pr + D -> I
    [6 -1 2 1 5 1]};%I -> Pr + D

nr=4; %number of times we iternate
tinc = 0.05;
tfin = 100;
tvec = 0:tinc:tfin; %uniform time vector
x=[10 1 0 0 0 0]; %x = [R Pr RPr P D I]

c=[2 1 4 2 0.5 0.05 0 0]; %propensity (without reaction 7 and 8)
X=zeros(nr, length(tvec)); %preallocating
%for j=1:nr %lay Gillespie runs along the rows of matrix and average
 down the columns
 %   [t,dimer] = mygill(tfin,rtab,x,c);
  %  X(j,:)= interp1(t, dimer, tvec); %interpolation
%end
%avg = mean(X,1); %average
%dev = std(X); %standard deviation
%plot(tvec,avg) %plot time vs dimer count graph
%hold on
%plot(tvec,(avg+dev),'r') %plot with std dev added
%plot(tvec,(avg-dev),'r') %plot wiht std dev subtracted
%xlabel('time')
%ylabel('dimer count')
%hold off

%figure

c=[2 1 4 2 0.5 0.05 0.01 0.05]; %propensity (with reaction 7 and 8)
X2=zeros(nr, length(tvec)); %preallocating
for j=1:nr %lay Gillespie runs along the rows of matrix and average
 down the columns
```

```
    [t,dimer] = mygill(tfin,rtab,x,c);
    X2(j,:)= interp1(t, dimer(5, :), tvec); %interpolation
end
avg2 = mean(X2,1); %average
dev2 = std(X2,1); %standard deviation
plot(tvec,avg2) %plot time vs dimer count graph
hold on
plot(tvec,(avg2+dev2),'r') %plot with std dev added
plot(tvec,(avg2-dev2),'r') %plot wiht std dev subtracted
xlabel('time') % set x-axis as time
ylabel('dimer count') %set y-axis as dimer count
title('Deterministic vs. Stochastic Operon')

tspan = [0 tfin]; %from 0 to time final
yinit = x'; %this gives derivative of x
[T, X] = ode23(@(t1, x) mcaode(t1, x, c), tspan, yinit);%
plot(T,X(:,5)); %plots 5th column, which is dimer
hold off

figure
[t,dimer] = mygill(tfin,rtab,x,c); %we need to call mygill function to
 plot
k = {'R' 'Pr' 'RPr' 'P' 'D' 'I'};
for j =  1:6 %there are six elements

  subplot(2, 3, j); %(row, column, place to plot)
  plot(T, X(:, j)) %we plot jth column
  ylabel(k(j))%it gives y-axis
  xlabel('t'); %set x-axis as time
  hold on
  plot(t, dimer(j, :), '.') %we need to plot dimer corresponding to
 each plot
  xlim([0 30])
end
end
% This is mygill function.
% mygill function takes in tfin, rtab, x, and c, and results in [t,
 dimer].
% Within this function, update function is used.
% Example: mygill(100, rtab, [10 1 0 0 0 0], c=[2 1 4 2 0.5 0.05 0 0])
%code-run mygill
%set k
% [tnew, xnew] = ode23(@(variables(t,x)) funct, [0.tfin], x0)
% subplot(row,column,where to plot) e.g. (2,4,i) --use forloop!
% We plot deterministic as well as stochastic operon



function [t,all] = mygill(tfin, rtab, x, c)
t=[0]; %preallocating t
iter=1; %initial iter
dimer=[0]; %preallocating dimer
while t(iter)<tfin %run as long as t(iter) < tfin
    iter=iter+1; %counter
```

```
    a = c .* update(x); %likelihood of each reaction happening
    a0 = sum(a); %add components in a
    r1 = rand; %random number between 0 and 1
    tau = (1/a0)*log(1/r1); %calculate when it is likely to occur
    r2 = rand; %random number between 0 and 1
    cumchance = cumsum(a)/sum(a); %weighted probabilities
    b = find(cumchance > r2,1); %determining which reaction will occur
    x(rtab{b}(1:2:end)) =  x(rtab{b}(1:2:end)) + rtab{b}
(2:2:end); %update x according to which reaction occured
    t(iter) = t(iter-1) + tau; %update t(iter) by adding calculated
 tau to prior t(iter)
    all(1, iter) = x(1);
    all(2, iter) = x(2);
    all(3, iter) = x(3);
    all(4, iter) = x(4);
    all(5, iter) = x(5);
    all(6, iter) = x(6); %update the dimer count
end
end

function dx = mcaode(t,x,c)
dx = zeros(6,1);
dx(1,1) = -1*c(1)*x(1)*x(2)+c(2)*x(3)+c(3)*x(3); % 1, 2, 3rd equations
dx(2,1) = -1*c(1)*x(1)*x(2)+c(2)*x(3)+c(3)*x(3)-
c(7)*x(2)*x(5)+c(8)*x(6); %1, 2, 3, 7, 8th equations
dx(3,1) = c(1)*x(1)*x(2)-c(2)*x(3)-c(3)*x(3); %1, 2, 3rd equations
dx(4,1) = 10*c(3)*x(3)-c(4)*(x(4))^2+2*c(5)*x(5)-c(6)*x(4); %3, 4, 5,
 6th equations
dx(5,1) = (0.5)*c(4)*(x(4)^2)-c(5)*x(5)-c(7)*x(2)*x(5)+c(8)*x(6); %4,
 5, 7, 8th equations
dx(6,1) = c(7)*x(2)*x(5)-c(8)*x(6); %7, 8th equations
end
%{this gives derivative function of each reaction, considering
 reactants and resultants}

function h = update(x) %h is an updated probability
h(1) = x(1)*x(2);%reaction 1
h(2) = x(3);%reaction 2
h(3) = x(3);%reaction 3
h(4) = x(4)*(x(4)-1)/2;%reaction 4
h(5) = x(5);%reaction 5
h(6) = x(4);%reaction 6
h(7) = x(2)*x(5);%reaction 7
h(8) = x(6);%reaction 8
end
```


Deterministic vs. Stochastic Operon
dimer count
time
0
50
100
150
200
250
300
350
400
0
20
40
60
80
100


R
Pr
RPr
P
D
I
t
0
5
10
0.5
1
15
20
25
30
100
200
300

Published with MATLAB® R2015b