1. **Coupled oscillators**

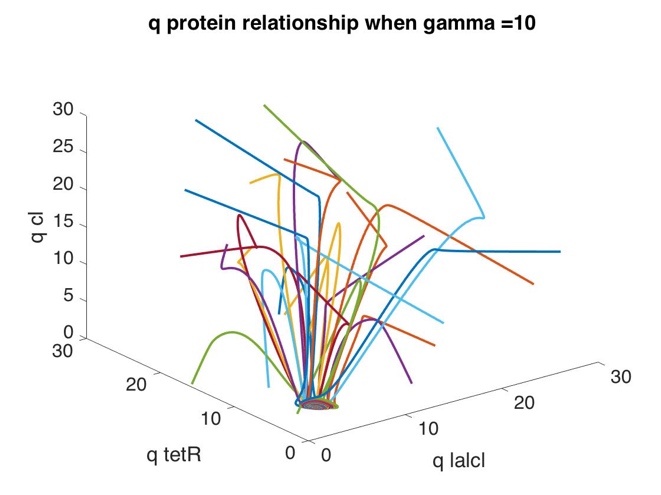
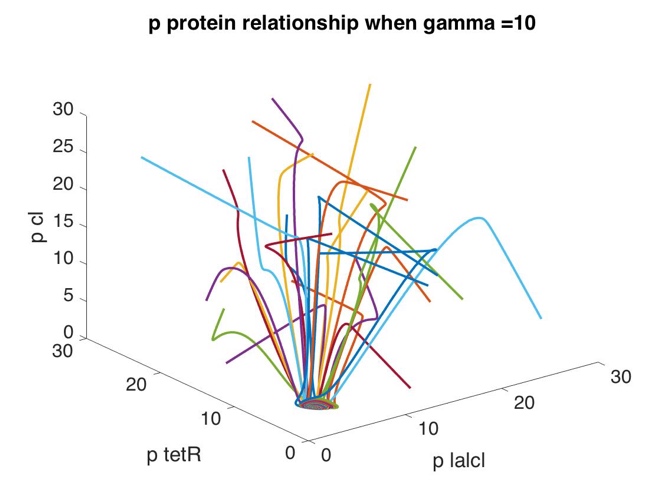
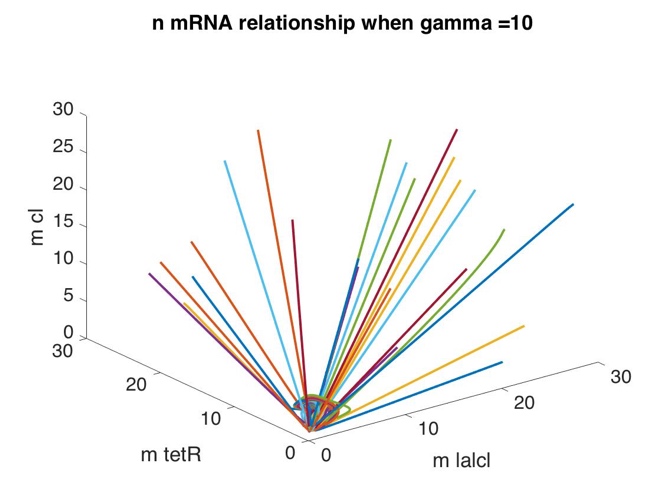
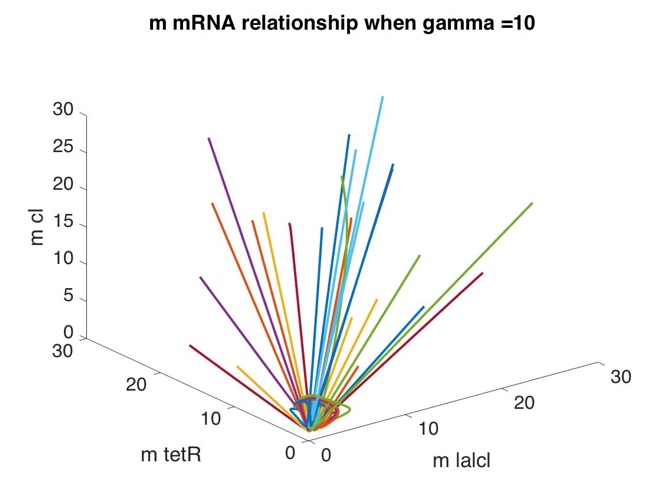
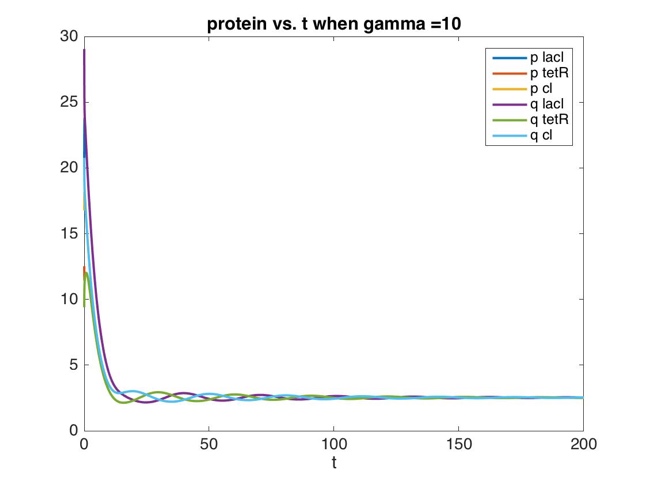
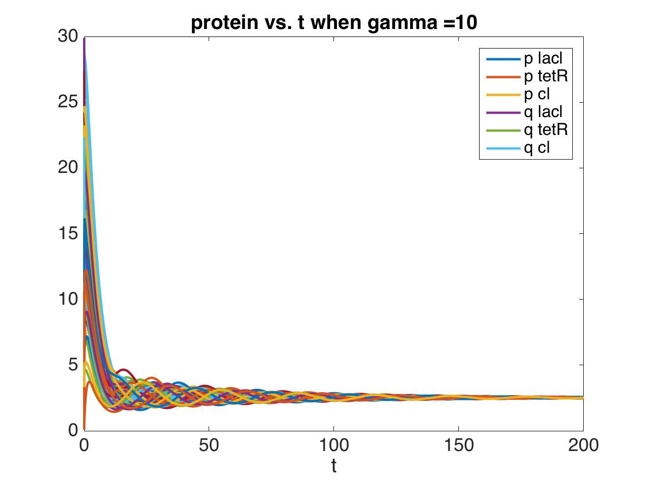
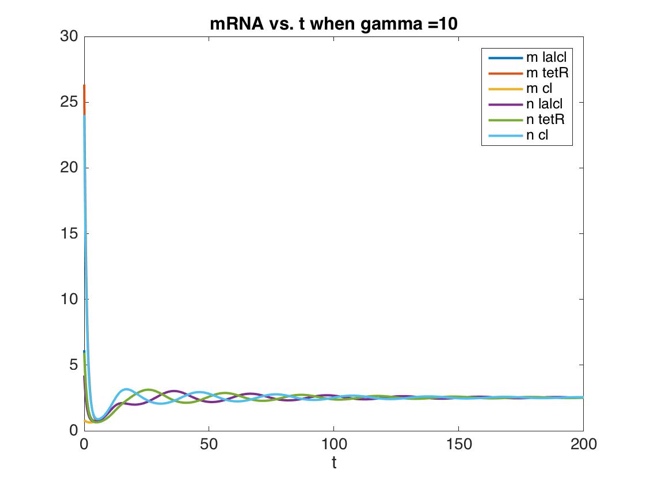
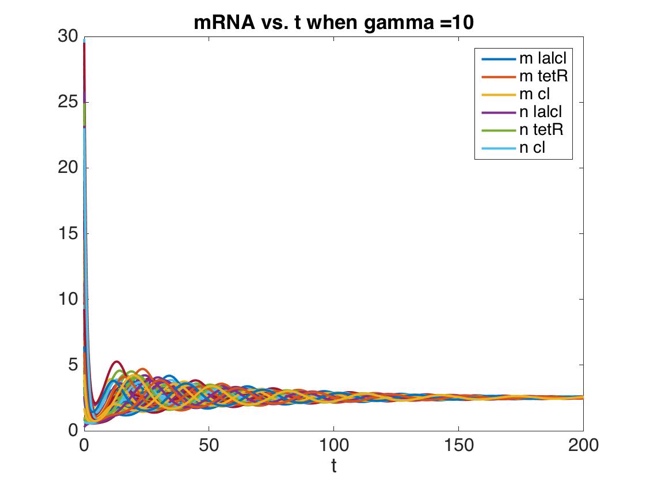
**Question a:** Do nonzero value of gamma serve to synchronize the two oscillators, in that when we start each oscillator with different initial conditions, the trajectories eventually converge over time?

Based on Figure 1, when gamma is positive, the trajectories did eventually converge over time. To be precise, it continues to oscillate (in a seemingly constant period) with decreasing fluctuation. They did look synchronizing with each other.

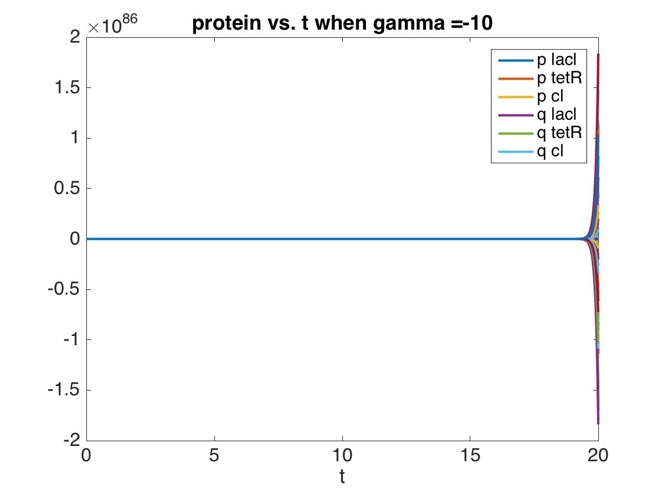
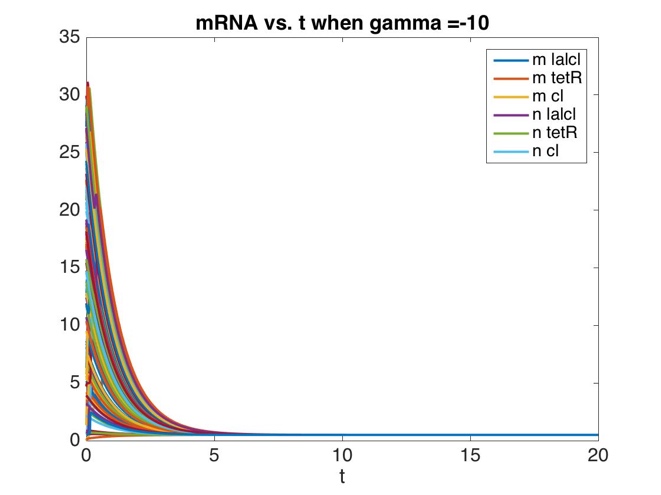
Based on Figure 2, when gamma is negative, the trajectories did eventually converge over time. In a few rounds, the oscillation disappears and reach a steady state without oscillation.

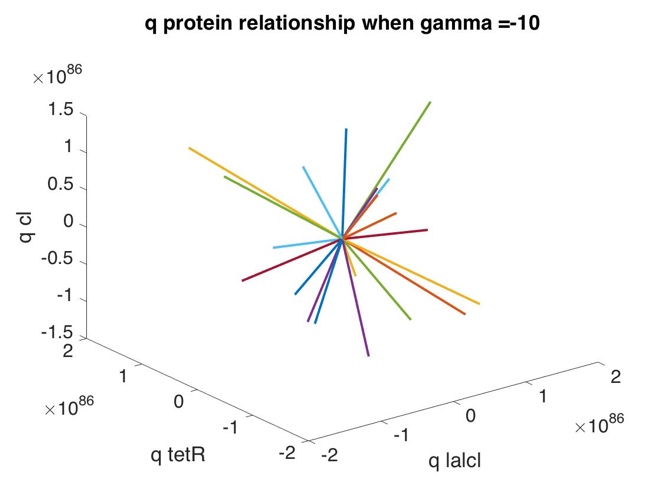
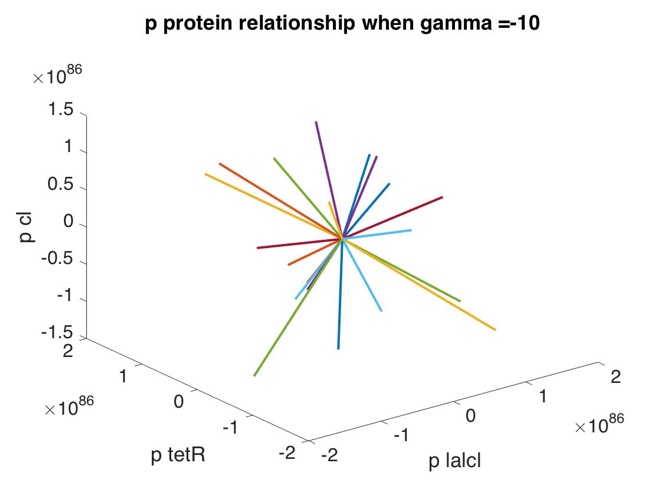
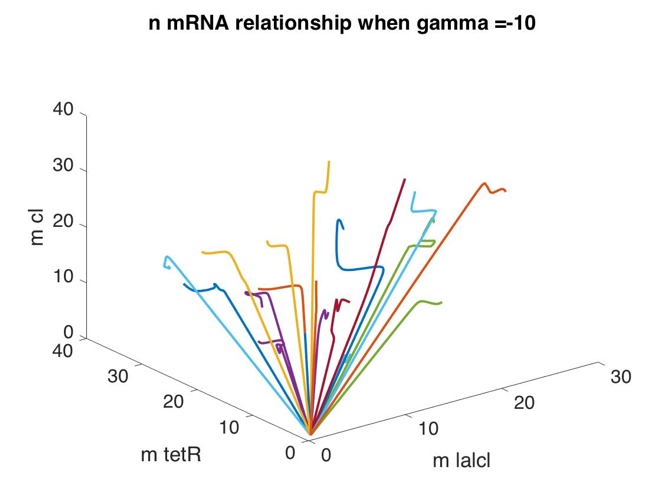
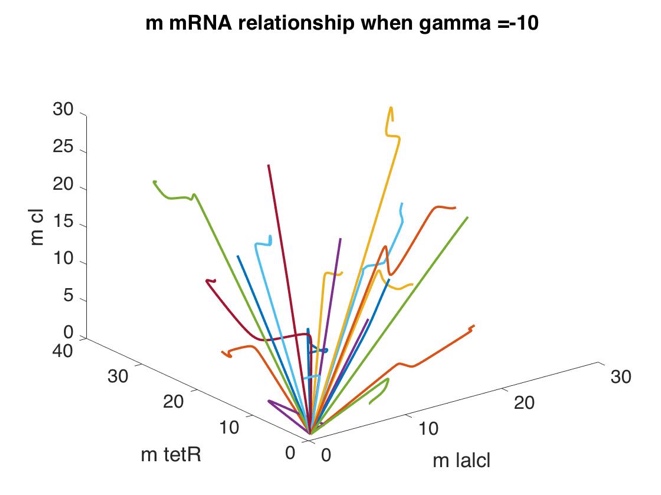
**Question b:** Does it matter whether gamma is positive or negative?

Based on the intuition, it converges no matter gamma is positive or negative. Looking at the equation, a positive gamma means the rate of repressilator 1 protein synthesis would increases if repressilator 2 has more protein, and vice verse; a negative gamma means the rate of repressilator 1 protein synthesis would decreases if repressilator 2 has more protein, and vice verse. This implies that a positive gamma would be a negative feedback control system and negative gamma would be a positive feedback control system. This intuition is supported by my simulation, that with a negative feedback loop and positive gamma, the system continues oscillating till converge, while the positive feedback with negative gamma decays directly to converge.

****

**Figure 1:** when gamma = 10, (a) multiple trials of mRNA dynamics, (b) mRNA dynamics, (c) multiple trials of protein dynamics, (d) protein dynamics, (e) mRNA relationship of repressilator 1 m, (f) mRNA relationship of repressilator 1 n, (g) protein relationship of repressilator 2 q, (h) protein relationship of repressilator 2 p.

****

****

**Figure 2:** when gamma = -10, (a) mRNA dynamics, (b) protein dynamics (c) mRNA relationship of repressilator 1 m, (d) mRNA relationship of repressilator 1 n, (e) protein relationship of repressilator 2 q, (f) protein relationship of repressilator 2 p

%% I. Coupled oscillators

alpha=15;

alpha0=0.5;

beta=0.2;

n=2;

gamma=10;

% gamma=-10;

for numreps=1:25

p = [alpha,alpha0,beta,n,gamma];

x0 = 30\*rand(12,1) ;

Tmax=200;

[T,Y] = ode45(@coupleOsci,[0 Tmax],x0,[],p);

figure(1)

set(gca,'FontSize',16)

plot(T,Y(:,1:3),'LineWidth',2) ; hold on

plot(T,Y(:,7:9),'LineWidth',2) ; hold on;

legend('m lalcl','m tetR','m cl','n lalcl','n tetR','n cl')

xlabel('t') ;

title(strcat('mRNA vs. t when gamma = ',num2str(gamma)));

figure(2)

set(gca,'FontSize',16)

plot(T,Y(:,4:6),'LineWidth',2) ; hold on

plot(T,Y(:,10:12),'LineWidth',2) ; hold on;

legend('p lacl','p tetR','p cl','q lacl','q tetR','q cl')

xlabel('t') ;

title(strcat('protein vs. t when gamma = ',num2str(gamma)));

figure(3)

set(gca,'FontSize',16)

plot3(Y(:,1),Y(:,2),Y(:,3),'LineWidth',2) ; hold on

xlabel('m lalcl');ylabel('m tetR');zlabel('m cl');

title(strcat('m mRNA relationship when gamma = ',num2str(gamma)));

figure(4)

set(gca,'FontSize',16)

plot3(Y(:,7),Y(:,8),Y(:,9),'LineWidth',2) ; hold on

xlabel('m lalcl');ylabel('m tetR');zlabel('m cl');

title(strcat('n mRNA relationship when gamma = ',num2str(gamma)));

figure(5)

set(gca,'FontSize',16)

plot3(Y(:,4),Y(:,5),Y(:,6),'LineWidth',2) ; hold on

xlabel('p lalcl');ylabel('p tetR');zlabel('p cl');

title(strcat('p protein relationship when gamma = ',num2str(gamma)));

figure(6)

set(gca,'FontSize',16)

plot3(Y(:,10),Y(:,11),Y(:,12),'LineWidth',2) ; hold on

xlabel('q lalcl');ylabel('q tetR');zlabel('q cl');

title(strcat('q protein relationship when gamma = ',num2str(gamma)));

end

**coupleOsci.m**

function dy = coupleOsci(t,y,p)

dy = zeros(12,1);

dy(1) = -y(1) + p(1)/(1.+y(6)^p(4))+ p(2);

dy(2) = -y(2) + p(1)/(1.+y(4)^p(4))+ p(2);

dy(3) = -y(3) + p(1)/(1.+y(5)^p(4))+ p(2);

dy(4) = -p(3)\*(y(4)-y(1)) + p(5)\*(y(10)-y(4));

dy(5) = -p(3)\*(y(5)-y(2)) + p(5)\*(y(11)-y(5));

dy(6) = -p(3)\*(y(6)-y(3)) + p(5)\*(y(12)-y(6));

dy(7) = -y(7) + p(1)/(1.+y(12)^p(4))+ p(2);

dy(8) = -y(8) + p(1)/(1.+y(10)^p(4))+ p(2);

dy(9) = -y(9) + p(1)/(1.+y(11)^p(4))+ p(2);

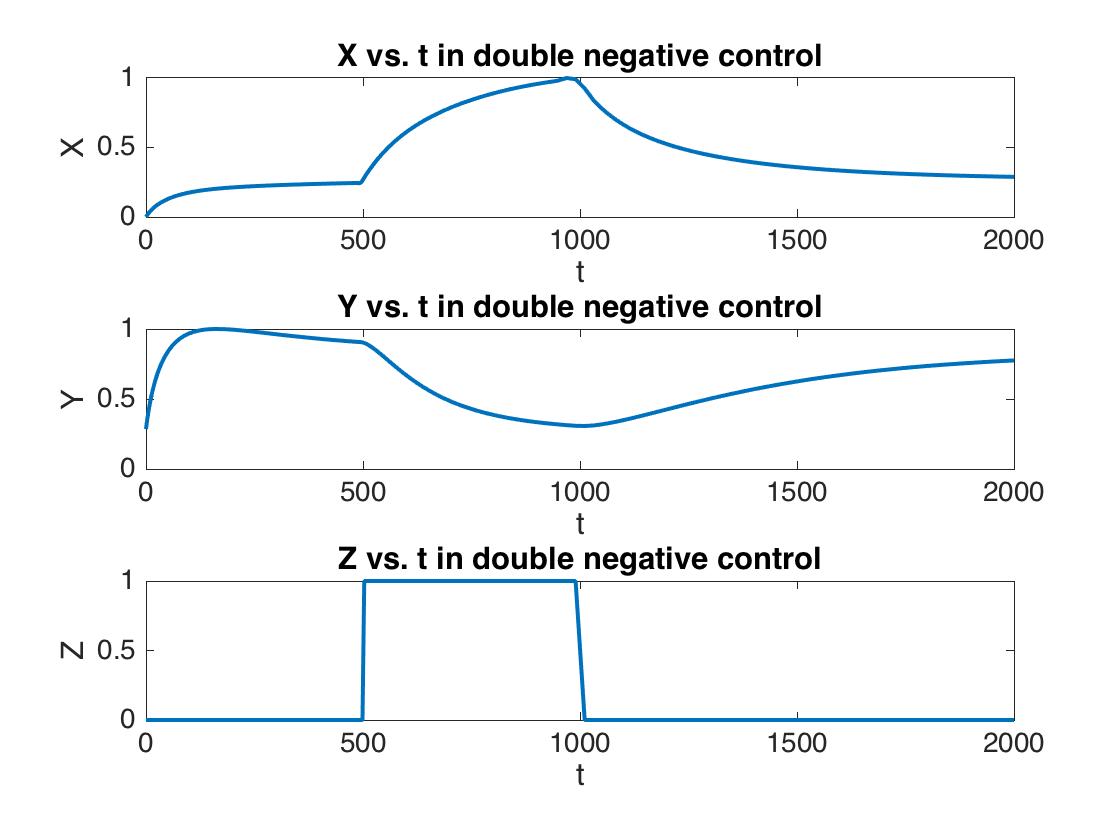
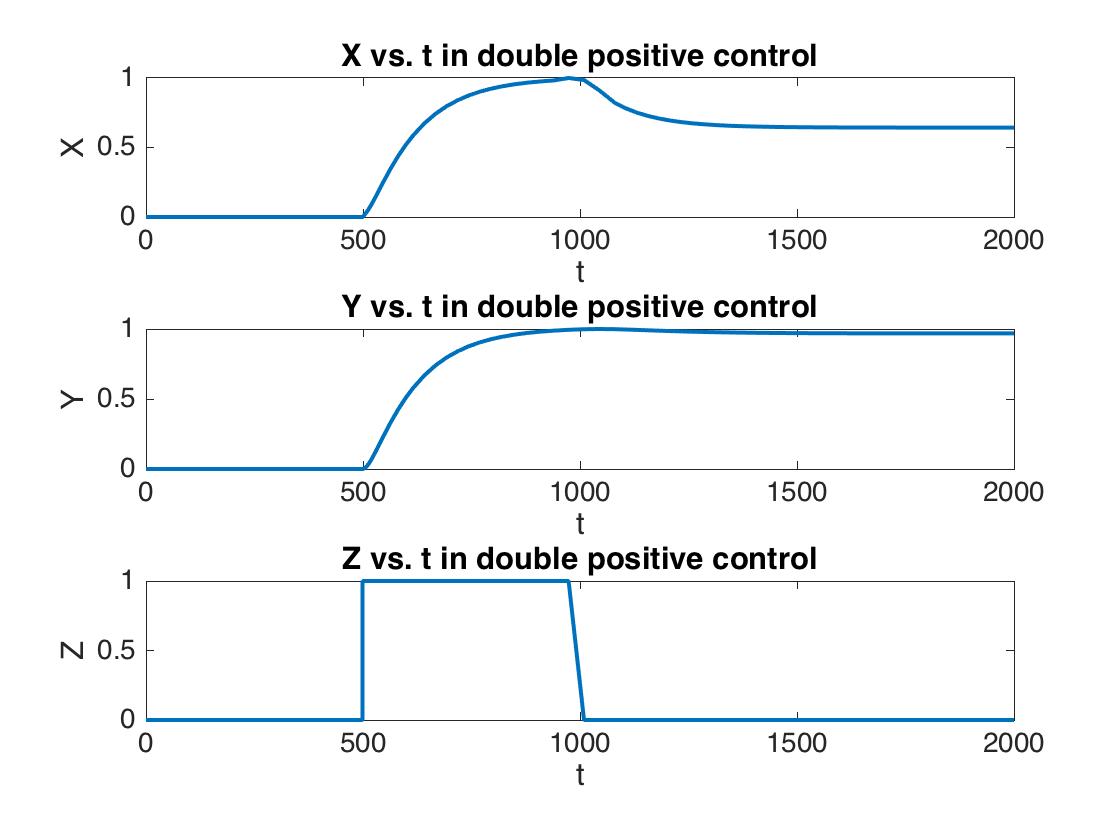
dy(10) = -p(3)\*(y(10)-y(1)) + p(5)\*(y(4)-y(10));

dy(11) = -p(3)\*(y(11)-y(2)) + p(5)\*(y(5)-y(11));

dy(12) = -p(3)\*(y(12)-y(3)) + p(5)\*(y(6)-y(12));

end

1. **Systems biology and network motifs**



**Figure 3 & 4:** dynamics of X, Y, Z in double positive and negative feedback loop

%% II. Systems Biology and Network Motifs

alpha1=0.1;

alpha2=0.1;

n=1;

% ctrl=1; % positive

ctrl=0; % negative

alpha3=0.001;

beta=0.01;

for numreps=1:1

p = [alpha1,alpha2,n,ctrl,alpha3,beta];

% x0 = [1; 0; 0];

x0 = [1; 0; 1];

Tmax=2000;

[T,Y] = ode45(@doubleCtrl,[0 Tmax],x0,[],p);

X1 = Y(:,2)./max(Y(:,2));

Y1 = Y(:,3)./max(Y(:,3));

Z1 = [Y(:,1) > 500] - [Y(:,1) > 1000];

figure(1)

set(gca,'FontSize',16)

subplot(3,1,1)

plot(T,X1,'LineWidth',2) ; hold on

ylabel('X');

xlabel('t');

ylim([0 1]);

% title('X vs. t in double positive control');

title('X vs. t in double negative control');

subplot(3,1,2)

plot(T,Y1,'LineWidth',2) ; hold on

ylabel('Y');

xlabel('t');

ylim([0 1]);

% title('Y vs. t in double positive control');

title('Y vs. t in double negative control');

subplot(3,1,3)

plot(T,Z1,'LineWidth',2) ; hold on

ylabel('Z');

xlabel('t');

ylim([0 1]);

% title('Z vs. t in double positive control');

title('Z vs. t in double negative control');

end

**doubleCtrl.m**

function dy = doubleCtrl(t,y,p)

dy = zeros(3,1);

dy(1) = 1;

if y(1) > 500 && y(1) < 1000

z = 1;

else

z = 0;

end

zx = (z^p(3)) \* p(1)/(1.+z^p(3));

zy = (z^p(3))^p(4) \* p(5)/(1.+z^p(3));

dy(2) = (y(3)^p(3))^p(4) \* p(1)/(1.+y(3)^p(3)) + zx - y(2)\*p(6)\*[y(2) > 0];

dy(3) = (y(2)^p(3))^p(4) \* p(2)/(1.+y(2)^p(3)) + zy - y(3)\*p(6)\*[y(3) > 0];

end

1. **Python**

**# Exercise 1**

rn = np.random.RandomState() # initialize a new RandomState object

rn.seed(100)

F = 5+rn.randn(1,100) \* 2 \*\* 0.5

F.mean() # 4.8526872367575509

F.var() # 1.8808398082683166

**# Exercise 2**

import scipy.integrate as si

def onedim(y, t):

yprime = np.array([y[0]])

return yprime

t = np.arange(0, 10.01, .01) # time points on which to solve

yzero = np.array([1.])

print (len(yzero))

y = si.odeint(onedim, yzero, t)

plt.plot(t[:], y[:, 0])

plt.xlabel('t')

plt.ylabel('x')

plt.title('Figure: 1D ODE simulation')

