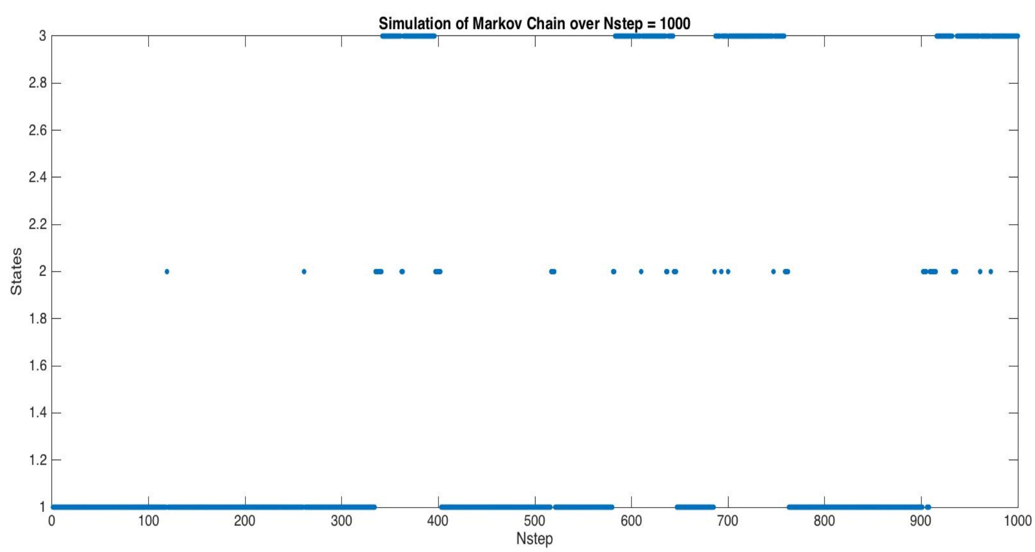
```
%% HW2 for AMATH422
% @Author: Baihan Lin
% @Date: Nov 2016

clear all; close all; clc;

rng(1);
```

%% I. Dwell time distributions: theory

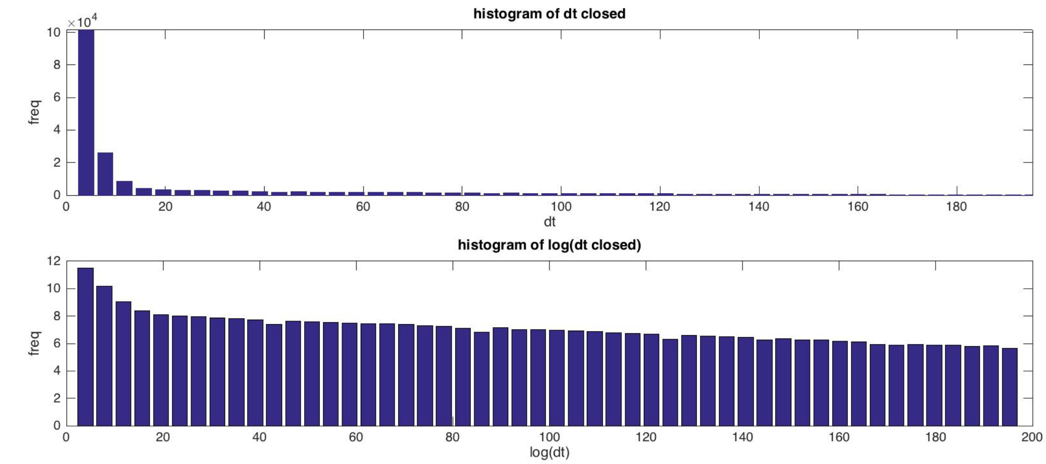
```
% ion channel with 4 open states and 2 closed states
% Mathematical arguement:
%
% Assume our Markov Matrix for this system, 6 X 6 matrix,
% is power-positive. Then, for any vector v,
% sum_i([A*v]) = sum_i(sum_k([a_ik * v_k])) = sum_k(v_k)
%
% For v, the dominant eigenvalue is:
% sum_i([A*v]) = sum_i(lamda*v_i) = lamda*sum_k(v_k)
% This implies lamda = 1, using sum_k(v_k) != 0
%
% Then, there exists, lamda as positive real number, and v > 0
% such that A*v = lamda*v, and |lamda| > |lamad_l| for all other
% eigenvalue lamda_l.
%
% Based on power positive stochastic 6 X 6 matrix A,
% lamda = 1 as dominant eigenvalue.
% p(k) -(k->inf)-> c_1*lamda^k * v = c_1 * v_1 = pi
% which is the equilibirium distribution for this Markov chain
%
% Since power-positive, lim (k->inf) (A^k * p(0)) = pi for any p(0)
% Thus, the answer to our question, the probability distribution for
% dwell time in open states, should be exponential in a certain type,
% but as a combination of four exponential function.

% [P1(k+1);P2(k+1);P3(k+1);P4(k+1);P5(k+1);P6(k+1);] as P(k+1)
% P(k+1) = A*P(k);
% If P1, P2, P3, P4 means open states.
% P_open(k+1) = P1(k+1)+P2(k+1)+P3(k+1)+P4(k+1)
%             = sum_j(P(k)*(a1j+a2j+a3j+a4j))
%             = sum_j(P(1)*(a1j+a2j+a3j+a4j)^(k-1))
% Q.E.D.!
```

```
%% II. Simulating Markov Chains and dwell times

% Markov Matrix
A = [0.98 0.1 0;
    0.02 0.7 0.05;
    0 0.2 0.95 ];

% across the entire time frame
Nstep = 10^7;
S = zeros(1,Nstep);  % states
S(1) = 1; % from C1

% simulation of Markov Chains
for k=1:Nstep-1
    rd=rand ;
    if rd < A(1,S(k))
        S(k+1) = 1;  % S(k+1) = S1 (C1)
    elseif rd < A(1,S(k))+A(2,S(k))
        S(k+1) = 2; % S(k+1) = S2 (C2)
    else
        S(k+1) = 3; % S(k+1) = S3 (O)
    end
end;

% plot the firstatet 1000 steps since 10^7 is too long
fig1 = figure
set(gca,'FontSize',18);
plot(1:1000,S(1:1000),'.','Markersize',20);
xlabel('Nstep','FontSize',16);
ylabel('States','FontSize',16);
title('Simulation of Markov Chain over Nstep = 1000');
```

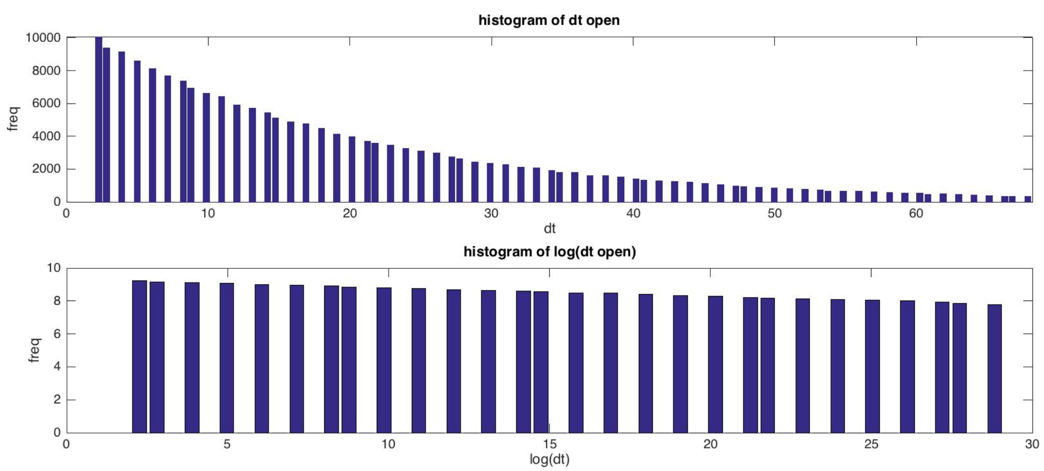



```
% Calculate dominant eigenvector
[V,D] = eig(A);
lamdas = diag(D);
jmax=find(abs(lamdas) == max(abs(lamdas))) ;
```

```
dom_eigenvec=V(:,jmax);
rescaled_dom_eigenvec = dom_eigenvec/sum(dom_eigenvec)
% Output of rescaled_dom_eigenvec:
%    0.5000
%    0.1000
%    0.4000

% fraction of time in each state with outputs:
S1 = length(find(S==1))/Nstep  % 0.4999
S2 = length(find(S==2))/Nstep  % 0.1000
S3 = length(find(S==3))/Nstep  % 0.4001

% Here the fractions match with the rescaled dominant eigenvector.

% reduced states
rstate = S;
rstate(find(S == 2)) = 1; % reduction

% Find dwell times in the different reduced states.
dt_closed=[];
dt_open=[];

% Initial Condition
S_0 = rstate(1);
Nstep_0=1;

for k = 2:length(rstate)
    if rstate(k) ~= S_0
        dt = k- Nstep_0 + 1;
        if S_0==1
            dt_closed = [dt_closed dt];
        else
            dt_open = [dt_open dt];
        end
        S_0=rstate(k);
        Nstep_0=k;
    end
end

%plot histograms
fig2 = figure

subplot(211)
[Nc,xc]=hist(dt_closed,200);
bar(xc,Nc)
axis([0 max(xc)/4 -Inf Inf])
xlabel('dt')
ylabel('freq')
title('histogram of dt_ closed')

subplot(212)
bar(xc(1:50),log(Nc(1:50)))
xlabel('log(dt)')
ylabel('freq')
title('histogram of log(dt_ closed)')
```

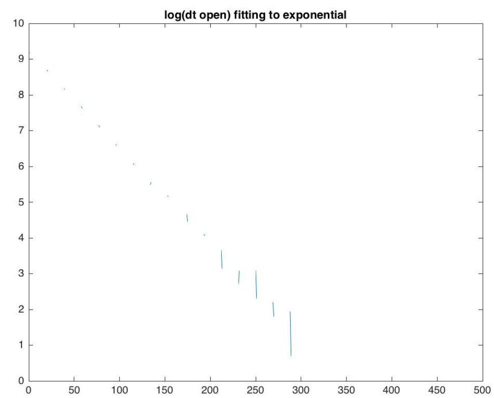


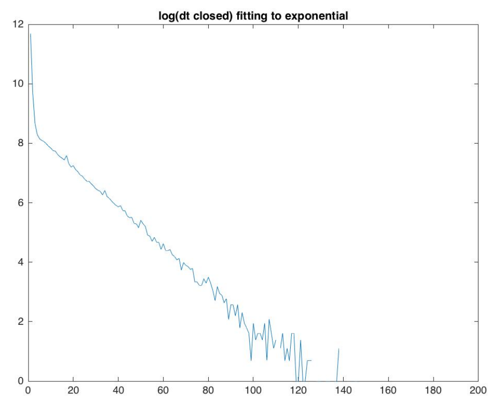

```
fig3 = figure

subplot(211)
[No,xo]=hist(dt_open,500);
bar(xo,No)
axis([0 max(xo)/4 -Inf Inf])
xlabel('dt')
ylabel('freq')
title('histogram of dt_ open')

subplot(212)
bar(xo(1:50),log(No(1:50)))
xlabel('log(dt)')
ylabel('freq')
title('histogram of log(dt_ open)')
```




```
% Based on the hypothesis, here we fit it to a single exponential:
fig4 = figure;
plot(1:length(No),log(No))  %see what we're fitting
title('log(dt open) fitting to exponential')
parao = polyfit(1:50,log(No(1:50)),1) % -0.0666    8.7731
E_ajj = exp(parao(1)) % 0.9355
```

```
% Based on the hypothesis, here we fit it to a single exponential:
fig5 = figure;
plot(1:length(Nc),log(Nc))  %see what we're fitting
title('log(dt closed) fitting to exponential')
parac = polyfit(1:50,log(Nc(1:50)),1) % -0.0666    8.7731
E_ajj = exp(parac(1)) % 0.9355

% We see the distribution of dwell times for the open state
% fits exponential well (log graph almost linear). But the
% distribution for closed states is not, perhaps due to two
% different close states.
```

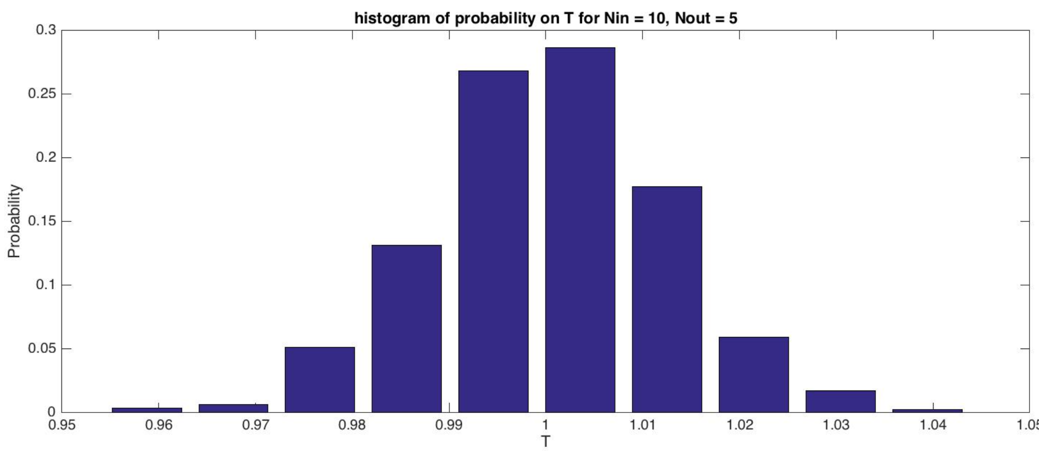





```
%% III. Simulating Markov Chains and neural spiking

Trial = 1000;
Nin = 10;
Nout = 5;

Snet = zeros(1, Trial);

% Markov Matrices
Ain = [0.98 0.1 0;
    0.02 0.7 0.05;
    0 0.2 0.95 ];
Aout = [0.9 0.1 0;
    0.1 0.6 0.1;
    0 0.3 0.9 ];

% across the entire time frame
Nstep = 10^6;

for t = 1:Trial

   Sin = zeros(Nin,Nstep);
    Sout = zeros(Nout,Nstep);

    Sin(:,1) = 1;
    Sout(:,1) = 1;
```

```
% simulation of Markov Chains
for k=1 : Nstep-1
    % Get states of inward channels
    for n = 1:Nin
        rd=rand;
        if rd < Ain(1,Sin(n,k))
            Sin(n,k+1) = 1; % S(k+1) = S1 (C1)
        elseif rd < Ain(1,Sin(n,k))+Ain(2,Sin(n,k))
            Sin(n,k+1) = 2; % S(k+1) = S2 (C2)
        else
            Sin(n,k+1) = 3; % S(k+1) = S3 (O)
        end
    end
    % Get states of outward channels
    for m = 1:Nout
        rd=rand;
        if rd < Aout(1,Sout(m,k))
            Sout(m,k+1) = 1; % S(k+1) = S1 (C1)
        elseif rd < Aout(1,Sout(m,k))+Aout(2,Sout(m,k))
            Sout(m,k+1) = 2; % S(k+1) = S2 (C2)
        else
            Sout(m,k+1) = 3; % S(k+1) = S3 (O)
        end
    end
end;

% Calculate dominant eigenvector of Ain
[Vin,Din] = eig(Ain);
lamdas_in = diag(Din);
jmax_in=find(abs(lamdas_in) == max(abs(lamdas_in))) ;
dom_eigenvec_in=Vin(:,jmax_in);
rescaled_dom_eigenvec_in = dom_eigenvec_in/sum(dom_eigenvec_in);
num_of_channels_S_in = Nin*rescaled_dom_eigenvec_in
% Output of num_of_channels_S_in:
%     5.0000
%     1.0000
%     4.0000

% double check with num of channels in each state with outputs:
N1in = length(find(Sin==1))/Nstep  % 4.9968
N2in = length(find(Sin==2))/Nstep  % 1.0014
N3in = length(find(Sin==3))/Nstep  % 40.017

% Calculate dominant eigenvector of Aout
[Vout,Dout] = eig(Aout);
lamdas_out = diag(Dout);
jmax_out=find(abs(lamdas_out) == max(abs(lamdas_out))) ;
dom_eigenvec_out=Vout(:,jmax_out);
rescaled_dom_eigenvec_out = dom_eigenvec_out/sum(dom_eigenvec_out);
num_of_channels_S_out = Nout*rescaled_dom_eigenvec_out
% Output of num_of_channels_S_out:
%     1.0000
%     1.0000
%     3.0000

% double check with num of channels in each state with outputs:
```

```
    N1out = length(find(Sout==1))/Nstep  % 0.9992
    N2out = length(find(Sout==2))/Nstep  % 0.9989
    N3out = length(find(Sout==3))/Nstep  % 3.0019

    % The equilibrium of the net current at t
    Snet(t) = N3in-N3out
    % Output of Snet:
    %     1.0000

    % Thus, it shows that the equilibrium would have 1 unit flows

end

% plot histograms
fig6 = figure

[Nt,xt]=hist(Snet,10);
bar(xt,Nt/Trial)
% axis([0 max(xt)/4 -Inf Inf])
xlabel('T')
ylabel('Probability')
title('histogram of probability on T for Nin = 10, Nout = 5')
```




```
%% For Nin = 1000, Nout = 500

Trial = 100;
Nin = 1000;
Nout = 500;

Snet = zeros(1, Trial);

% Markov Matrices
Ain = [0.98 0.1 0;
    0.02 0.7 0.05;
    0 0.2 0.95 ];
Aout = [0.9 0.1 0;
    0.1 0.6 0.1;
    0 0.3 0.9 ];
```

```
% across the entire time frame
Nstep = 10^6;

for t = 1:Trial

    Sin = zeros(Nin,Nstep);
    Sout = zeros(Nout,Nstep);

    Sin(:,1) = 1;
    Sout(:,1) = 1;

    % simulation of Markov Chains
    for k=1 : Nstep-1
        % Get states of inward channels
        for n = 1:Nin
            rd=rand;
            if rd < Ain(1,Sin(n,k))
                Sin(n,k+1) = 1; % S(k+1) = S1 (C1)
            elseif rd < Ain(1,Sin(n,k))+Ain(2,Sin(n,k))
                Sin(n,k+1) = 2; % S(k+1) = S2 (C2)
            else
                Sin(n,k+1) = 3; % S(k+1) = S3 (O)
            end
        end
        % Get states of outward channels
        for m = 1:Nout
            rd=rand;
            if rd < Aout(1,Sout(m,k))
                Sout(m,k+1) = 1; % S(k+1) = S1 (C1)
            elseif rd < Aout(1,Sout(m,k))+Aout(2,Sout(m,k))
                Sout(m,k+1) = 2; % S(k+1) = S2 (C2)
            else
                Sout(m,k+1) = 3; % S(k+1) = S3 (O)
            end
        end
    end;

    % Calculate dominant eigenvector of Ain
    [Vin,Din] = eig(Ain);
    lamdas_in = diag(Din);
    jmax_in=find(abs(lamdas_in) == max(abs(lamdas_in))) ;
    dom_eigenvec_in=Vin(:,jmax_in);
    rescaled_dom_eigenvec_in = dom_eigenvec_in/sum(dom_eigenvec_in);
    num_of_channels_S_in = Nin*rescaled_dom_eigenvec_in
    % Output of num_of_channels_S_in:
    %     500
    %     100
    %     400

    % double check with num of channels in each state with outputs:
    N1in = length(find(Sin==1))/Nstep  % 500
    N2in = length(find(Sin==2))/Nstep  % 100
    N3in = length(find(Sin==3))/Nstep  % 400

    % Calculate dominant eigenvector of Aout
    [Vout,Dout] = eig(Aout);
```

```
    lamdas_out = diag(Dout);
    jmax_out=find(abs(lamdas_out) == max(abs(lamdas_out))) ;
    dom_eigenvec_out=Vout(:,jmax_out);
    rescaled_dom_eigenvec_out = dom_eigenvec_out/sum(dom_eigenvec_out);
    num_of_channels_S_out = Nout*rescaled_dom_eigenvec_out
    % Output of num_of_channels_S_out:
    %    100
    %    100
    %    300

    % double check with num of channels in each state with outputs:
    N1out = length(find(Sout==1))/Nstep  % 100
    N2out = length(find(Sout==2))/Nstep  % 100
    N3out = length(find(Sout==3))/Nstep  % 300

    % The equilibrium of the net current at t
    Snet(t) = N3in-N3out
    % Output of Snet:
    %    100

    % Thus, it shows that the equilibrium would have 1 unit flows

end

% plot histograms
fig7 = figure

[Nt,xt]=hist(Snet,10);
bar(xt,Nt/Trial)
% axis([0 max(xt)/4 -Inf Inf])
xlabel('T')
ylabel('Probability')
title('histogram of probability on T for Nin = 1000, Nout = 500')
```