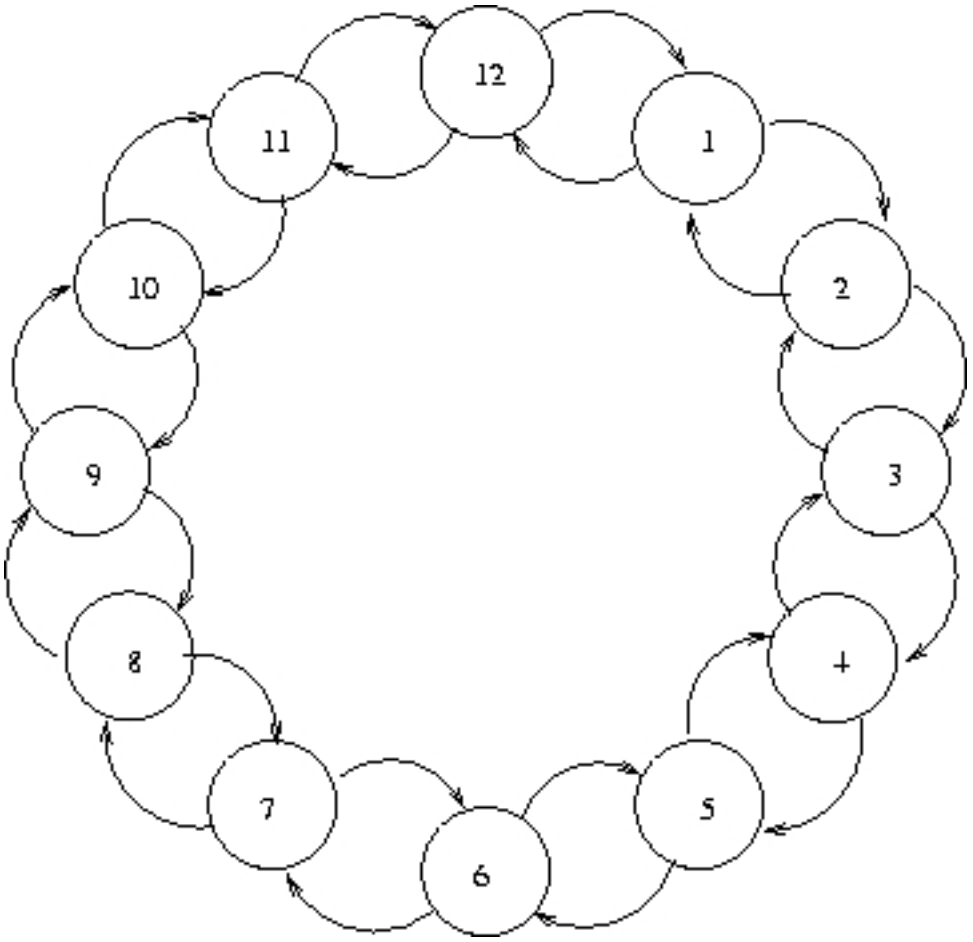
Backward Shift of Hippocampal Place Fields

Catherine Johnston, caj2@rice.edu



Backward Shift of Hippocampal Place Fields: Catherine Johnston, caj2@rice.edu

Published 29-Jul-2008 13:02:15

Abstract

This code simulates the backward shift of a simple ring network. This backward shift has been shown experimentally by Mehta et. al. in their paper Experience-dependent, asymmetric expansion of hippocampal place fields. If a pre-synaptic cell spikes before a post-synaptic cell the weight between the two cells should increase. If the post-synaptic cell spikes before the pre-synaptic cell, the weight between the cells should decrease. This causes a backward shift in the firing of cells, so that each cell starts to fire earlier.

Table of Contents

Chapter 1. Code:

```
function backward_shift_new(dt)
close all

cells = 12;
ee = ones(cells-1, 1);
connect = diag(ee, 1) + diag(ee, -1);
connect(1, cells) = 1;
connect(cells, 1) = 1;
figure(1)
spy(connect)
title('CA3 to CA3 Connectivity')
ylabel('Post: CA3')
xlabel('Pre: CA3')
drawnow

laptime = 120; %laptime is ms
laps = 20;

simtime = laps*laptime;


Ne_M = cells;  %number of MEC excitatory cells
Ne_C = cells;

%parameters
tauM_E = 20;    %exc. membrane voltage decay constant (ms)
Vrest_E = -70;  %exc. resting potential (ms)
Vth_E = -54;    %exc. threshold potential (mV)
Vreset_E = -60; %exc. reset potential (mV)

tau_gE = 5;

wMax_net = 1;
Wnet = .1*wMax_net*connect;
Winp = 2*diag(ones(cells,1));

%Initiate vars
v_E = Vrest_E*ones(cells,1);  %voltage for CA3 E cells

gE_E = zeros(Ne_C,1);   %excitatory conductance of CA3 E cells

j=0;

%Parameters for plasticity from CA3 E to CA3 E cells
tauPlus_net = 10;
tauMin_net = 10;
APlus_net = wMax_net/10;
AMin_net = APlus_net*1.05;

%Variables for plasticity
M_net = zeros(cells,1);  %related to timings of post spikes (CA3-CA3)
P_net = zeros(1,cells);    %related to timings of pre spikes (CA3-CA3)

%constants to be used in while loop
gEbot = tau_gE + dt;
quot_E = tauM_E/dt;

Pbot_net = tauPlus_net + dt;
Mbot_net = tauMin_net + dt;
```

```
%Assume all CA3 cells are initially at rest
spikedEx_C = [];

trefe = 5;  % refractory period (ms)
refracCells = zeros(cells,1);   % initialize refractory cell vector
Wstore = spalloc(cells*cells, simtime, 24*simtime);
vstore = zeros(cells, simtime/dt);
gstore = vstore;
mstore = vstore;
pstore = vstore;

NumSpikedCA3 = 0;
figure(2)
fprintf('entering loop\n')
delta_t = laptime/cells;
% outputStep = ceil(simtime/100);
lap = 0;
while j*dt < simtime
    % Display the percentage complete
%     if mod(j*dt,outputStep)==0
%         fprintf('%g%% complete\n', (j*dt/simtime)*100)
%     end

    %Find MEC spikes

    spiked_inj = [];
    tdiff = mod(j*dt, laptime);
    if tdiff == 0
        lap = lap + 1;
    end
    if mod(j*dt, delta_t)==0
        spiked_inj = tdiff/delta_t;
        if spiked_inj==0
            spiked_inj=cells;
        end
    end

    %conductances decay
    gEtop = tau_gE*gE_E;
    gE_E = gEtop/gEbot;

    mstore(:, j+1) = M_net;
    pstore(:, j+1) = P_net;

    %P_net and M_net decay
    Ptop_net = tauPlus_net*P_net;
    P_net = Ptop_net/Pbot_net;

    Mtop_net = tauMin_net*M_net;
    M_net = Mtop_net/Mbot_net;

    %Add mec input
    gE_E = gE_E + sum(Winp(:,spiked_inj),2);

    %Add CA3 input
    gE_E = gE_E + sum(Wnet(:,spikedEx_C),2);

    vstore(:, j+1) = v_E;
    gstore(:, j+1) = gE_E;

    %Update voltages
    vtop_E = quot_E*v_E + Vrest_E; % +gEx*Eex
    vbot_E = quot_E + 1 + gE_E;
```

```
    refracIndex = find(refracCells > j*dt); % get cells that are still refractory
    vbot_E(refracIndex) = quot_E + 1;   % hold refractory cells at rest

    v_E = vtop_E./vbot_E;

    %find CA3 spiked cells
    spikedEx_C = find(v_E >= Vth_E);

    if ~isempty(spikedEx_C)
        v_E(spikedEx_C) = Vreset_E;
        refracCells(spikedEx_C) = j*dt + trefe;    % set new refractory times
        NumSpikedCA3 = NumSpikedCA3 + length(spikedEx_C);

        %CA3 - CA3 plasticity
        M_net(spikedEx_C) = M_net(spikedEx_C) - AMin_net;
        Wnet = stdp_net(spikedEx_C,Wnet,M_net,P_net,wMax_net);
        P_net(spikedEx_C) = P_net(spikedEx_C) + APlus_net;
    end

    %Set up next step
    j = j+1;

    %Plotting storage
        Wlong = reshape(Wnet', cells*cells, 1);
        Wstore(:, ceil(j*dt)) = Wlong;
%         if ~isempty(spikedEx_C)
%             figure(2)
%             ff = ones(length(spikedEx_C), 1);
%             plot(ceil(j*dt*ff), spikedEx_C, '.r')
%             hold on
%             drawnow
%         end


    if ~isempty(spikedEx_C)
        pos = (j*dt)/laptime;
        if j*dt <= 4*laptime + delta_t
            polar(pos*2*pi, 1,'.r')
            hold on

        elseif j*dt <= 12*laptime + delta_t
            polar(pos*2*pi, 1,'.b')

        else
            polar(pos*2*pi, 1,'.g')
        end
        drawnow
    end
end

hold off

% %plots
% figure(4)
% for k=1:cells
%     subplot(3, 4,k)
%     plot(dt*(1:(simtime)),vstore(k,:),'b')
%     title(['v cell ' num2str(k)])
% end
% figure(5)
% for k=1:cells
%     subplot(3, 4,k)
%     plot(dt*(1:(simtime)),gstore(k,:),'b')
%     title(['gE cell ' num2str(k)])
```

```
%
% end
% figure(6)
%
% for k=1:cells
%     subplot(3,4,k)
%     plot(dt*(1:simtime), mstore(k,:), 'b')
%     title(['M_net ' num2str(k)])
% end
% figure(7)
% for k=1:cells
%     subplot(3,4,k)
%     plot(dt*(1:simtime), pstore(k,:), 'b')
%     title(['P_net ' num2str(k)])
% end

[row, col, a] = find(connect == 1);

n = length(a);

if ceil(sqrt(n))*floor(sqrt(n)) >= n
    subdimx = floor(sqrt(n));
    subdimy = ceil(sqrt(n));
else
    subdimx = ceil(sqrt(n));
    subdimy = ceil(sqrt(n));
end

figure(3)
for k=1:n

    subplot(subdimx,subdimy,k)
    plot(dt*(1:simtime),Wstore((row(k)-1)*12+col(k),:))
    title(['W CA3  ' num2str(col(k)) ' to W CA3 ' num2str(row(k))])
end

return

function W = stdp_net(spks, W, M, P, wMax)

%plasticity portion code used to compare different stdp rules for a network

%Inputs:
%spks: indeces of spiked CA3 cells
%W: CA3 E - CA3 E weight matrix to be changed
%M: vector containing info about how long ago postsynaptic cells spiked
%P: vector containing info about how long ago presynaptic cells spiked
%wMax: maximum weight allowed

%Outputs:
%W: modified weight matrix

%plasticity implemented according to Song's paper

persistent W0
if isempty(W0)
    W0 = spones(W);
    disp('set WEtoE0')
end

%LTD (spikes are presynaptic)
Mmat = repmat(M, 1, length(spks));
deltaW = Mmat * wMax;
W(:, spks) = max(W(:, spks) + deltaW, 0);
```

```
%LTP (spikes are postsynaptic)
Pmat = repmat(P, length(spks), 1);
deltaW = Pmat .* W0(spks, :) * wMax;
W(spks, :) = min(W(spks, :) + deltaW, wMax);

return
```

Chapter 2. Command Window Output

The first figure shows the connectivity of the simple ring network. The second figure shows the spiking of the cells. The red dots represent spiking of the cells during laps 1 through 4, the blue dots represent spiking of the cells during laps 5 through 12, and the green dots represent the spiking of the cells during laps 13 through 20. This simulation was run with a timestep of 1 for 20 laps, with a laptime of 120 ms. The third figure graphically shows the changes in weights between the connected synapses. This shows that the weights for the pre-to-post synapses increase, and the weights for the post-to-pre synapses decrease.

```
entering loop
```

Chapter 3. Figure(s) Created

Table of Contents

figure 1

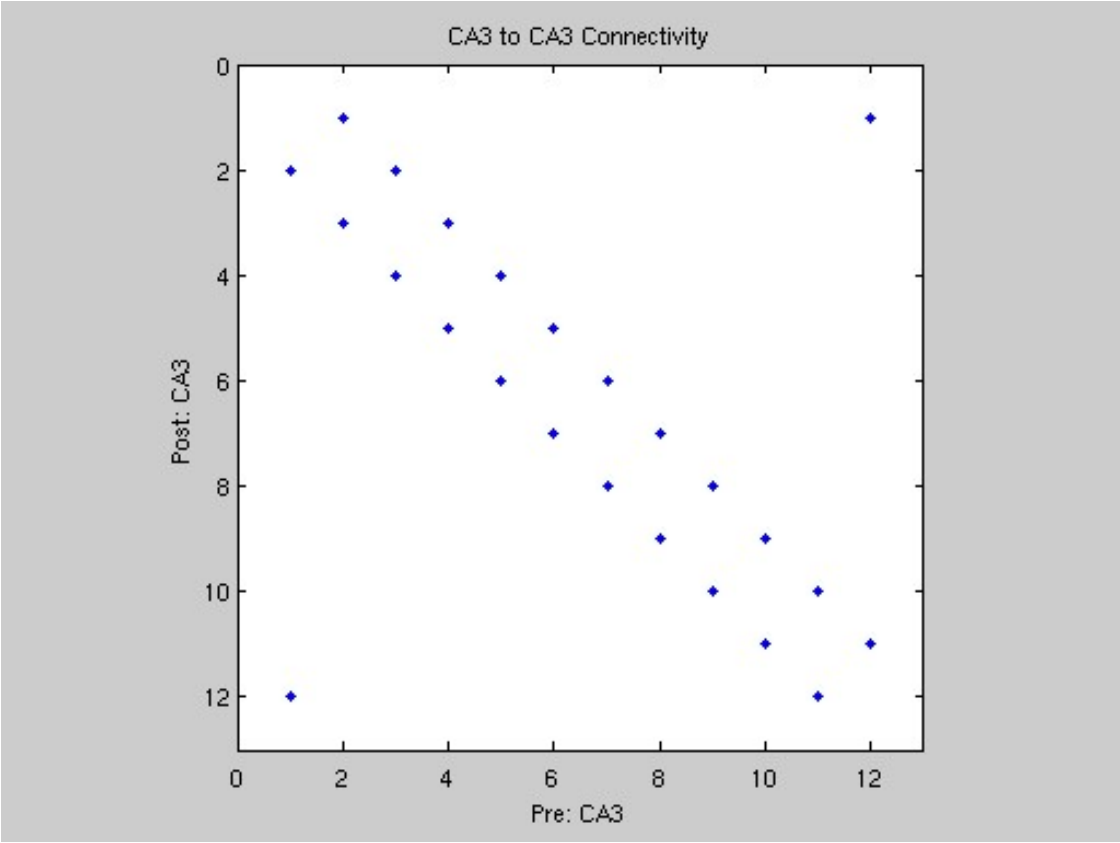


figure 2

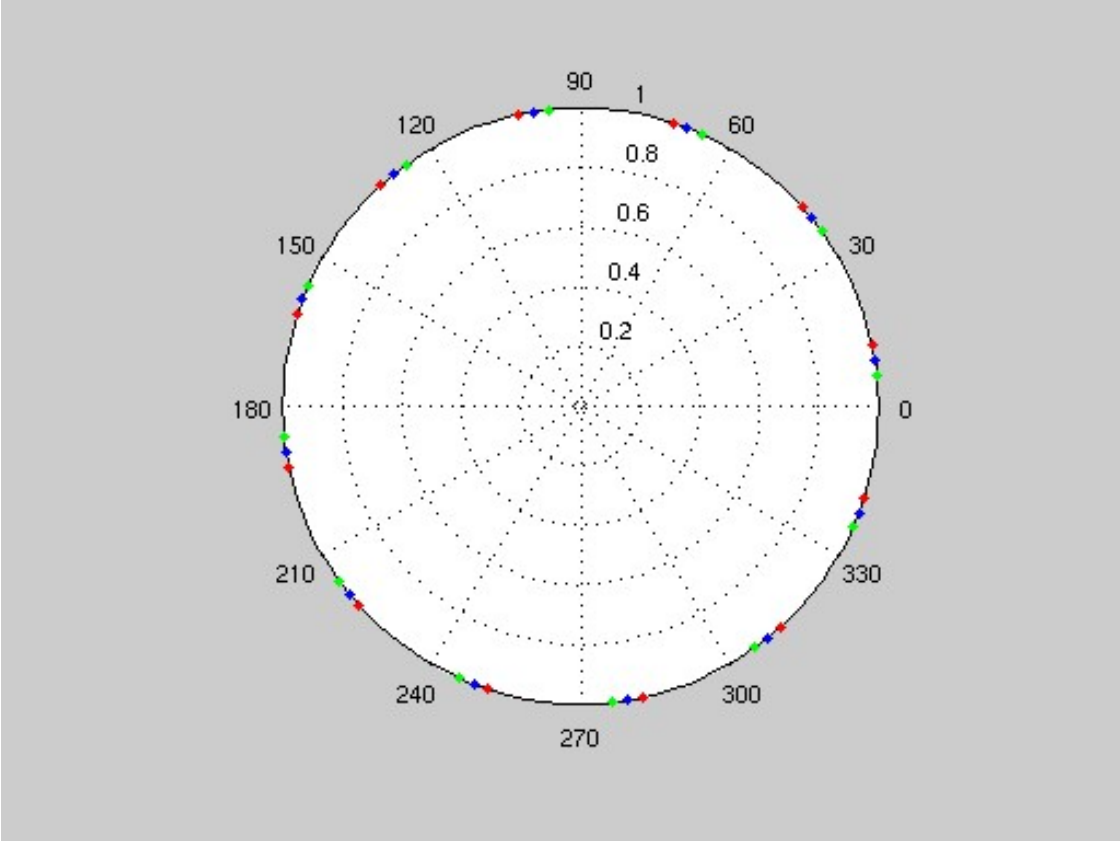
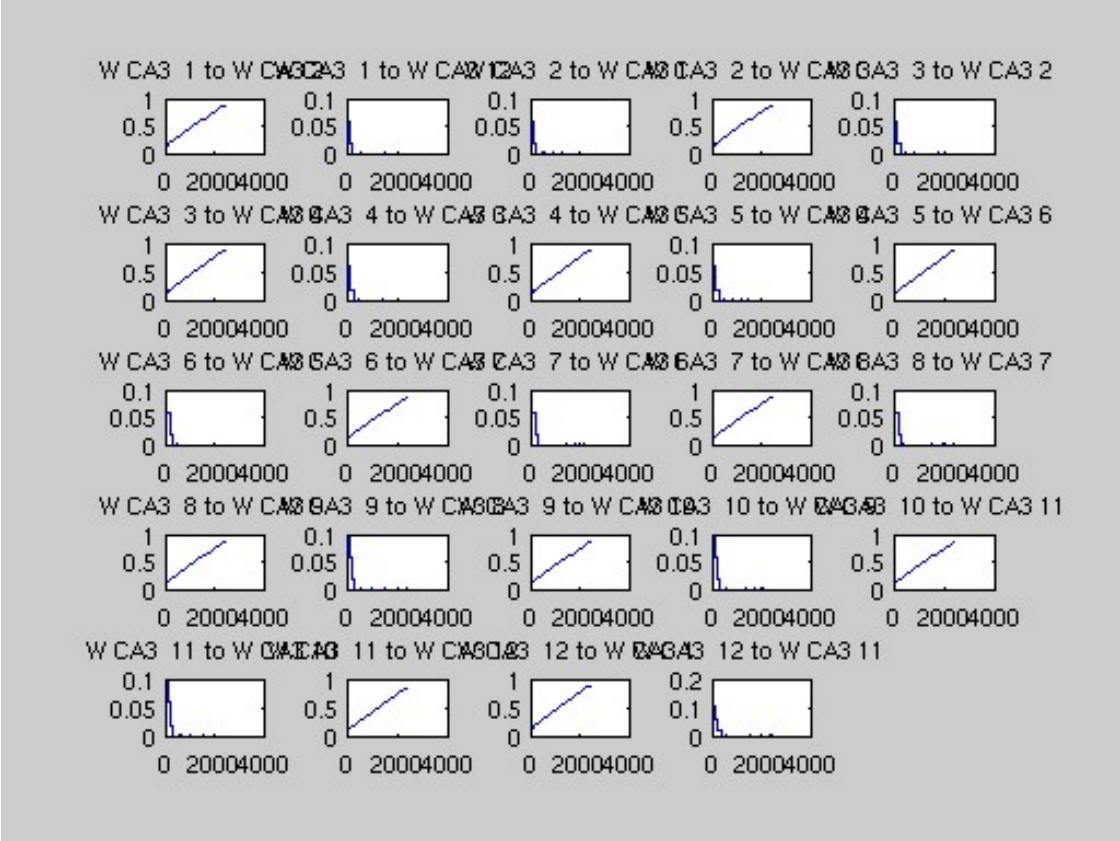


figure 3


W CA3 1 to W CA3 2
W CA3 2 to W CA3 3
W CA3 3 to W CA3 4
W CA3 4 to W CA3 5
W CA3 5 to W CA3 6
W CA3 6 to W CA3 7
W CA3 7 to W CA3 8
W CA3 8 to W CA3 9
W CA3 9 to W CA3 10
W CA3 10 to W CA3 11
W CA3 11 to W CA3 12
W CA3 12 to W CA3 11
0 20004000