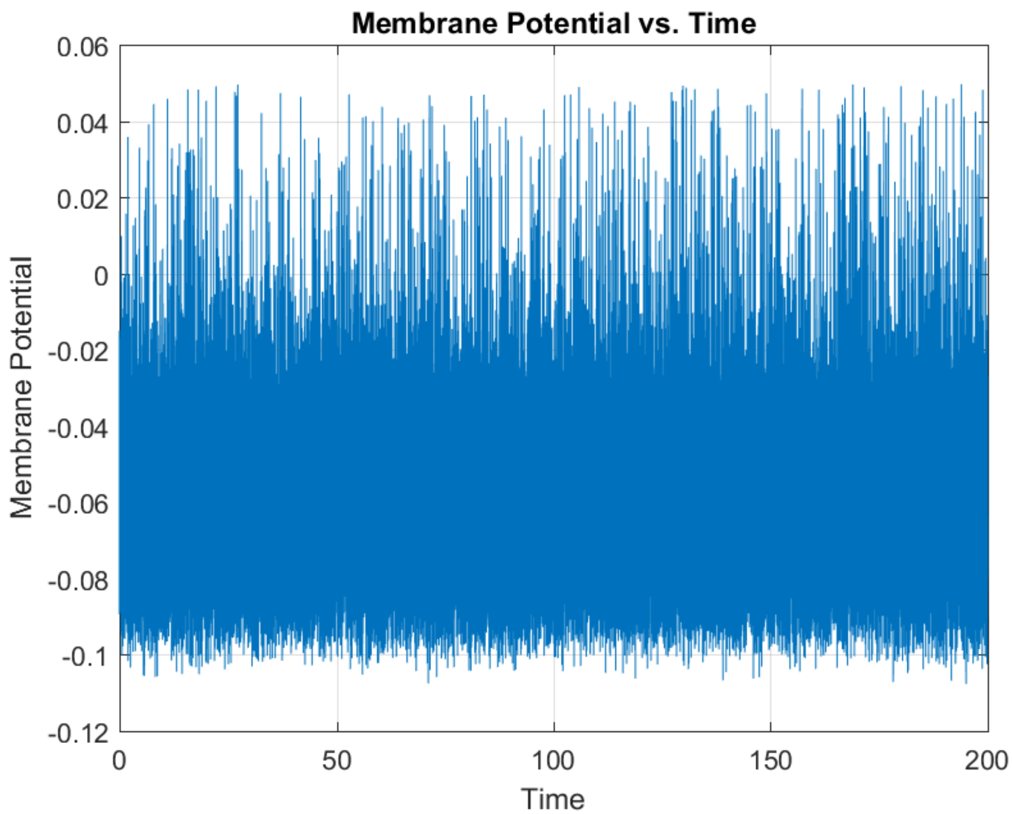
A.

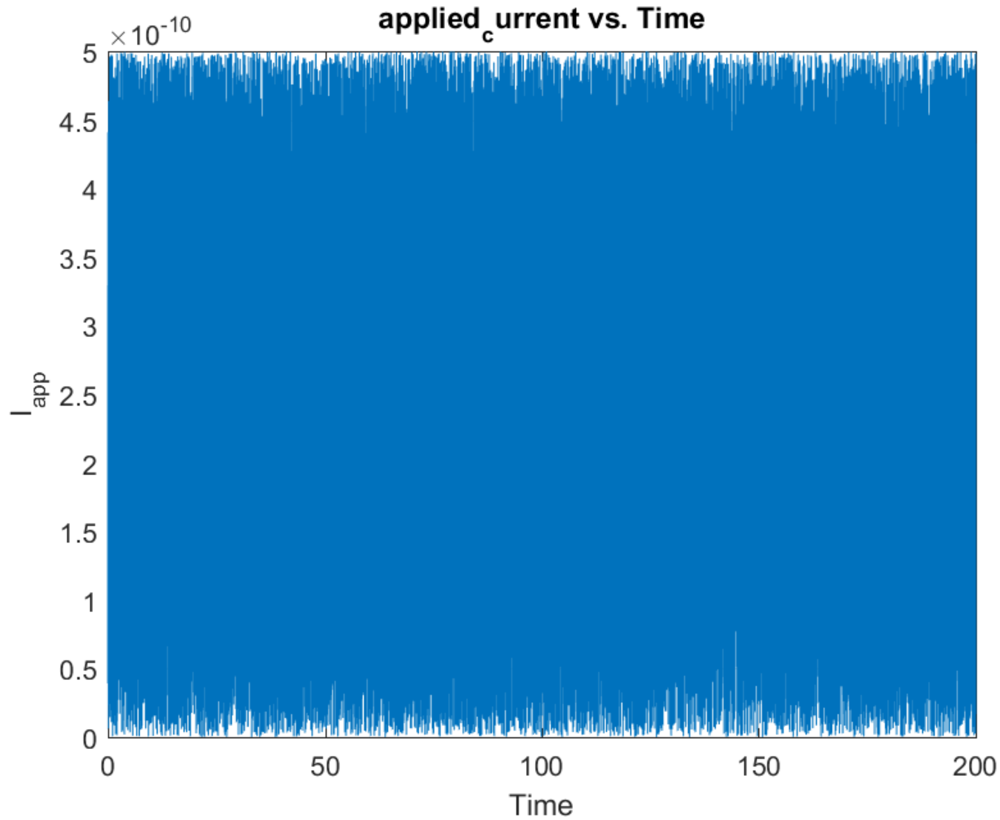
```
close all
clear
clc

leak_potential = -60e-3;                % Leak potential (V)
V_threshold = -50e-3;          % Threshold potential (V)
reset_potential = -80e-3;           % Reset potential (V)
delta_th = 2e-3;             % Threshold shift factor (V)
G_Leak = 8e-9;               % Leak conductance (S)
R=1/8e9;                   % resistance (ohm)
C = 100e-12;               % Capacitance (F)
a = 10e-9;                 % adaptation recovery (S)
b = 0.5e-9;               % adaptation strength (A)
tau_SRA = 50e-3;             % Adaptation time constant (s)

V_max = 50e-3;             % level of voltage to detect a spike
tau_m = R*C;

current_val = 1e-9 * (rand(1, 40000) -0.5);% Scale to nA and shift to range
[-0.5, 0.5]

total_time = 200000e-3;
dt=0.02e-3;
time_vector = 0:dt:total_time;

applied_current = zeros(size(time_vector));
for j=1:(length(current_val))
for i=1+250*(j-1):250*j
    applied_current(i)=current_val(j);
end
end
v = zeros(size(time_vector));       % initialize voltage
v(1) = leak_potential;
I_sra = zeros(size(time_vector));       % initialize adaptation variable
spikes = zeros(size(time_vector));

for j = 1:length(time_vector)-1

    if ( v(j) > V_max )          % if there is a spike
        v(j) = reset_potential;        % reset the voltage
        I_sra(j) = I_sra(j) + b;        % increase the adaptation variable by
b
        spikes(j) = 1;          % Record the spike as a "1" in the time-bin
    end

    % next line integrates the voltage over time, first part is like LIF
    % second part is an exponential spiking term
    % third part includes adaptation
    v(j+1) = v(j) + dt*( G_Leak*(leak_potential-v(j) + delta_th*exp((v(j)-
V_threshold)/delta_th) ) ...
       - I_sra(j) + applied_current(j))/C;

   % next line decys the adaptation toward a steady state in between spikes
```

```
    I_sra(j+1) = I_sra(j) + dt*( a*(v(j)-leak_potential) - I_sra(j)
)/tau_SRA;

end
f1 = figure;
figure(f1);
plot(time_vector, v)
grid on
title('Membrane Potential vs. Time');
xlabel('Time');
ylabel('Membrane Potential');
saveas(f1, sprintf('membrane_potential.png'));
```

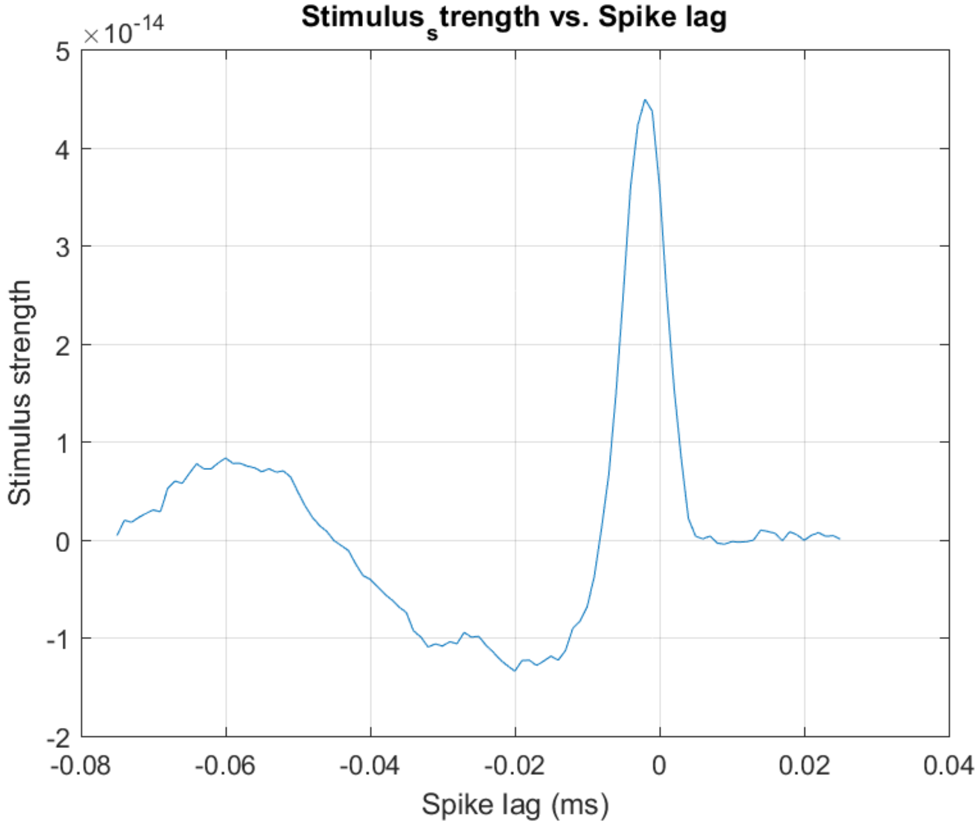



```
% Now downsample the stimulus and response to 1ms bins using the online
% function expandbin
new_dt = 0.001;                          % new time-bin of 1ms
spikes = expandbin(spikes,dt,new_dt);
spikes(find(spikes)) = 1;    % replace fractions with a "1" for a spike
applied_current = expandbin(applied_current,dt,new_dt);   % The function
expandbin does the downsampling
new_time_vector = 0:new_dt:total_time;

f2 = figure;
figure(f2);
plot(new_time_vector(1:200000), applied_current(1:200000))
```

```
grid on
title('applied_current vs. Time');
xlabel('Time');
ylabel('I_{app}');
saveas(f2, sprintf('applied_current.png'));
```




```
[sta, tcorr] = STA(applied_current, spikes, new_dt);
f3 = figure;
plot(tcorr,sta)
xlabel('Spike lag (ms)')
ylabel('Stimulus strength')
grid on
title('Stimulus_strength vs. Spike lag');
saveas(f3, sprintf('Stimulus_strength.png'));
```




```
function [new_vector] = expandbin(old_vector, old_dt, new_dt)
length_old = length(old_vector);
scale_ratio = round(new_dt/old_dt);
length_new = round(length_old/scale_ratio);
new_vector = zeros(1,length_new);
tsteps = 50;
for k = 1:length(new_vector)
new_vector(k) = mean(old_vector((k-1)*tsteps+1:k*tsteps));
end

function [sta, tcorr] = STA(Iapp, spikes, dt, tminus, tplus)
% Computes the spike-triggered average.

if (~exist('tminus'))
    tminus = 75e-3;
end
if (~exist('tplus'))
    tplus = 25e-3;
end
nminus = ceil(tminus/dt); % Number of time points before zero
nplus = ceil(tplus/dt);   % Number of time points after zero
nt = length(Iapp);      % length of original data set
sum_I = zeros(1,nminus+nplus+1);  % STA will accumulate here
tcorr = -nminus*dt:dt:nplus*dt;   % Vector of time points for STA
Iapp = Iapp - mean(Iapp); % Removes mean applied current
spikeposition = find(spikes);  % Time bins for each spike
totalspikes = length(spikeposition)     % Total number of spikes
for spike = 1:totalspikes
```

```
    ispike = spikeposition(spike);      % ispike is the bin containing a spike
    imin = max(1,ispike-nminus);        % Bin to start measuring stimulus
    imax = min(nt,ispike+nplus);        % Bin to finish measuring
    % The following lines put the stimulus, Iapp, into bins shifted
    % by the spike time (ispike)
    for i = imin:imax
        sum_I(i-ispike+nminus+1) = sum_I(i-ispike+nminus+1) ...
            + Iapp(i)/totalspikes;
    end
end

% Finally normalize by the number of contributing spikes to obtain the
% average (the mean)
sta = sum_I/totalspikes;
```

The spike-triggered average (STA) is a measure of the average synaptic input that precedes a spike. It is calculated by aligning the synaptic input to all spikes in a neuron's response and then averaging them. The STA can be influenced by a number of AELIF parameters, including:

Leak conductance: This parameter determines how quickly the membrane potential decays towards its resting potential. A higher leak conductance will lead to a faster decay of the synaptic input, and thus a narrower STA.

Capacitance: This parameter determines how much current is needed to change the membrane potential. A higher capacitance will lead to a slower change in membrane potential, and thus a wider STA.

Time constants: The AELIF model has two time constants, one for the excitatory and one for the inhibitory synaptic inputs. These time constants determine how quickly the synaptic input rises and falls. A shorter time constant will lead to a faster rise and fall of the synaptic input, and thus a narrower STA.

Threshold potential: This parameter determines the membrane potential at which the neuron fires an action potential. A higher threshold potential will make it more difficult for the neuron to fire, and thus will lead to a smaller STA.

. Leak Parameters:

Leak potential (leak_potential): A more negative leak potential will shift the STA downward, as it makes it more difficult for the membrane potential to reach the threshold.

Leak conductance (G_Leak): A higher leak conductance will lead to a faster decay of the STA, making it narrower. This is because a higher leak conductance means the membrane potential will decay more quickly towards the leak potential.

2. Threshold Parameters:

Threshold potential (V_threshold): A higher threshold potential will reduce the amplitude of the STA because it will be more difficult for the synaptic input to trigger a spike.

Threshold shift factor (delta_th): A larger threshold shift factor will lead to a more pronounced adaptation effect in the STA, as it will cause a greater increase in the threshold potential after each spike.

3. Capacitance (C):

A higher capacitance will lead to a wider STA, as it will take more charge (and therefore more time) to change the membrane potential.

4. Adaptation Parameters:

Adaptation recovery (a): A higher adaptation recovery rate will lead to a faster decay of the adaptation current, and thus a shorter-lasting adaptation effect in the STA.

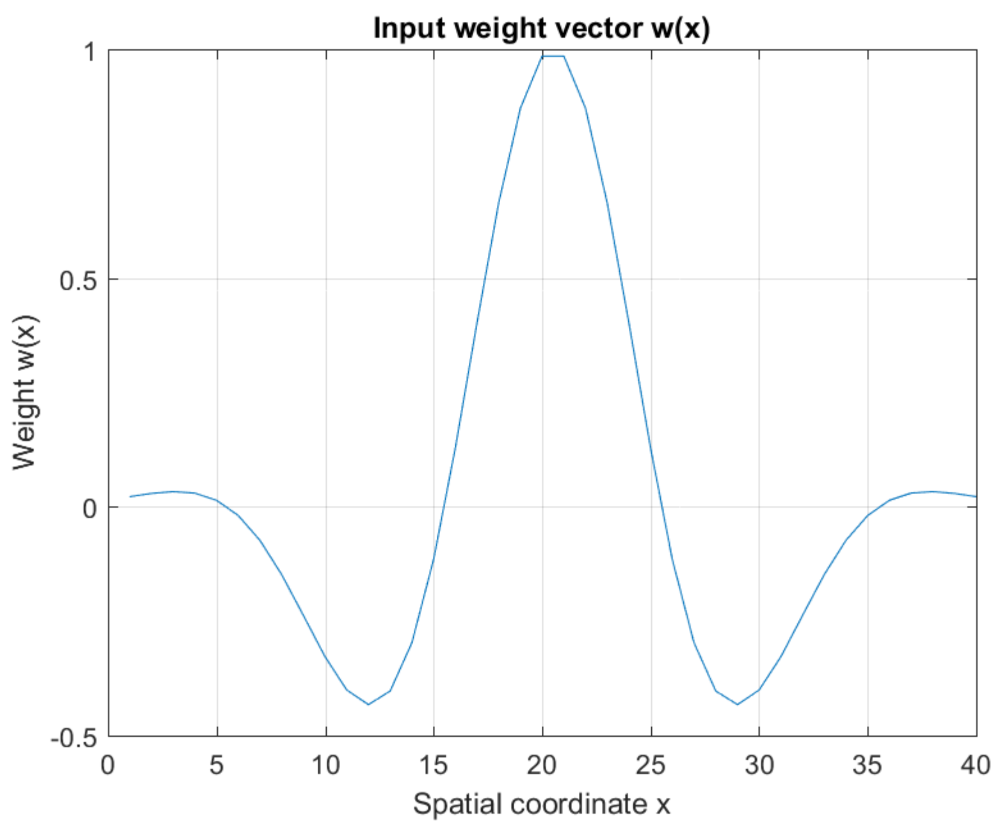
Adaptation strength (b): A larger adaptation strength will lead to a more pronounced adaptation effect in the STA, as it will cause a greater increase in the threshold potential after each spike.

Adaptation time constant (tau_SRA): A longer adaptation time constant will lead to a longer-lasting adaptation effect in the STA.

B.

```
%%%%%%%%%%%

%B

Nsteps = 40000;           % Number of time-steps of distinct applied currents

Nspatial = 40;            % Number of spatially distinct values per time-step

s = (rand(Nspatial, Nsteps) - 0.5)* 1e-9;

x_max = 40;

x_0 = 20.5;


% Generate the spatial coordinates

x = 1:x_max;


% Calculate the weight vector w(x)
```

```
w = cos(4*pi*(x - x_0)/x_max) .* exp(-16*((x - x_0)/x_max).^2);

I_app = w * s;

% Plot the weight vector

f4=figure;

plot(x, w);

xlabel('Spatial coordinate x');

ylabel('Weight w(x)');

title('Input weight vector w(x)');

grid on

saveas(f4, sprintf('Spatial coordinate vs Input weight vector_w.png'));
```




```
a = 40e-9;

b = 1e-9;

total_time = 200000e-3;

dt = 0.02e-3;
```

```
time_vector = 0:dt:total_time;
I_app_B = zeros(size(time_vector));
for j=1:(length(I_app))
for i=1+250*(j-1):250*j
   I_app_B(i)=I_app(j);
end
end
% Initialize variables
v = zeros(size(time_vector));
v(1) = leak_potential;
I_sra = zeros(size(time_vector));
spikes_B = zeros(size(time_vector));

% Simulation loop
for j = 1:length(time_vector)-1

   if (v(j) > V_max)
      v(j) = reset_potential;
      I_sra(j) = I_sra(j) + b;
      spikes_B(j) = 1;
   end

   v(j+1) = v(j) + dt*(G_Leak*(leak_potential-v(j) + delta_th*exp((v(j)-V_threshold)/delta_th)) ...
         - I_sra(j) + I_app_B(j))/C;

   I_sra(j+1) = I_sra(j) + dt*(a*(v(j)-leak_potential) - I_sra(j))/tau_SRA;

end
```

% Plot membrane potential

f5=figure;

plot(time_vector, v)

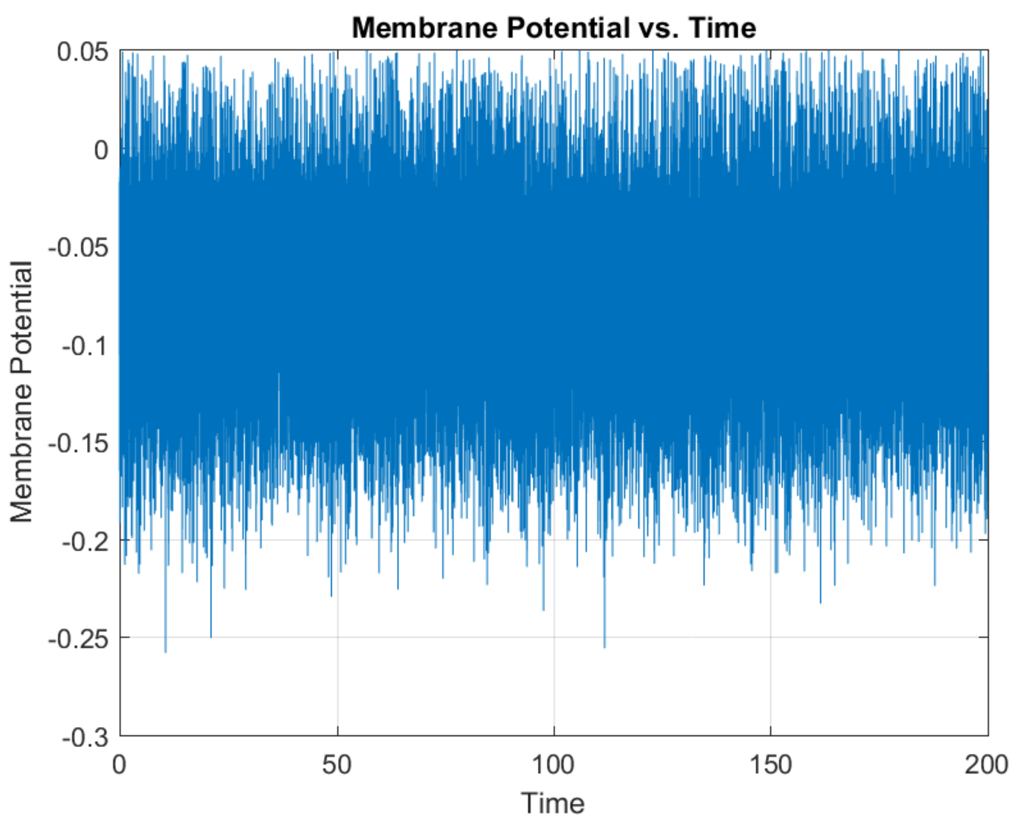
grid on

title('Membrane Potential vs. Time');

xlabel('Time');

ylabel('Membrane Potential');

saveas(f5, sprintf('Membrane Potential vs. Time_B.png'));




```
new_dt = 0.001;                                % new time-bin of 1ms
spikes_B = expandbin(spikes_B,dt,new_dt);
newlen = length(spikes);                       % New length of spike vector
newIapp = zeros(Nspatial,newlen);              % Define a new input vector
step_length = 0.001;
newnsteplength = round(step_length/new_dt); % New number of 1ms bins per step


for step = 1:Nsteps;                           % Loop through steps of constant
current
    istart = (step-1)*newnsteplength+1;  % first time point with 1ms steps
    istop = step*newnsteplength;           % last time point with 1ms steps
```

```
    % generate the input vector as before, but with bins of 1ms, not of dt
    applied_current_B(:,istart:istop) = s(:,step)*ones(1,newnsteplength);
end
I_app_B=mean(applied_current_B, 2)
[sta, tcorr] = STA_spatial(I_app_B, spikes_B, new_dt);

figure()
imagesc(fliplr(sta));        % reverses time-axis to plot STA
colormap(gray)               % grayscale
set(gca,'XTick',[1, 26, 51, 76, 101])
set(gca,'XTickLabel',{'-25' '0', '25', '50', '75'})
xlabel('Spike lag (ms)')
ylabel('Stimulus coordinate')
set(gca,'YTick',[ ])
imagesc(fliplr(sta(12, :));
imagesc(fliplr(sta(20, :));
imagesc(fliplr(sta(28, :));
imagesc(fliplr(sta(:, 25));
imagesc(fliplr(sta(:, 50));
imagesc(fliplr(sta(:, 75));

function [sta, tcorr] = STA_spatial(stim_array, spikes, dt, tminus, tplus)
% Computes the spatiotemporal spike-triggered average.

if (~exist('tminus'))
  tminus = 75e-3;
end
if (~exist('tplus'))
  tplus = 25e-3;
end

[Nspace, Nt] = size(stim_array); % Get spatial and temporal dimensions
nminus = ceil(tminus/dt); % Number of time points before zero
nplus = ceil(tplus/dt);  % Number of time points after zero

sta = zeros(Nspace, nminus+nplus+1); % Initialize STA
tcorr = -nminus*dt:dt:nplus*dt; % Vector of time points for STA

stim_array = stim_array - mean(stim_array, 2); % Remove mean from each
spatial bin
spikeposition = find(spikes); % Time bins for each spike
totalspikes = length(spikeposition); % Total number of spikes

for spike = 1:totalspikes
  ispike = spikeposition(spike); % Time bin containing the spike
  imin = max(1, ispike-nminus); % Start of stimulus window
  imax = min(Nt, ispike+nplus); % End of stimulus window

  % Accumulate stimulus values for each spatial bin, accounting for time
shift
  for i = imin:imax
    for j = 1:Nspace
      sta(j, i-ispike+nminus+1) = sta(j, i-ispike+nminus+1) + stim_array(j,
i)/totalspikes;
```

```
    end
  end
end

% Normalize by the number of contributing spikes
sta = sta / totalspikes;
```