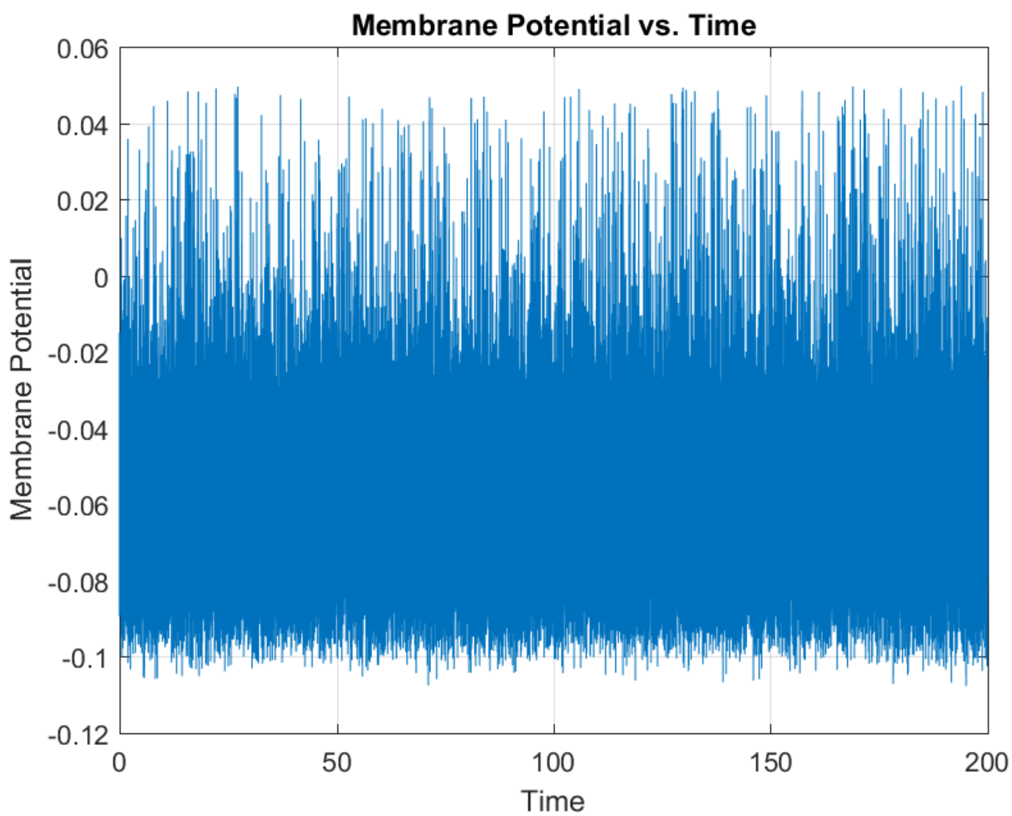
A.

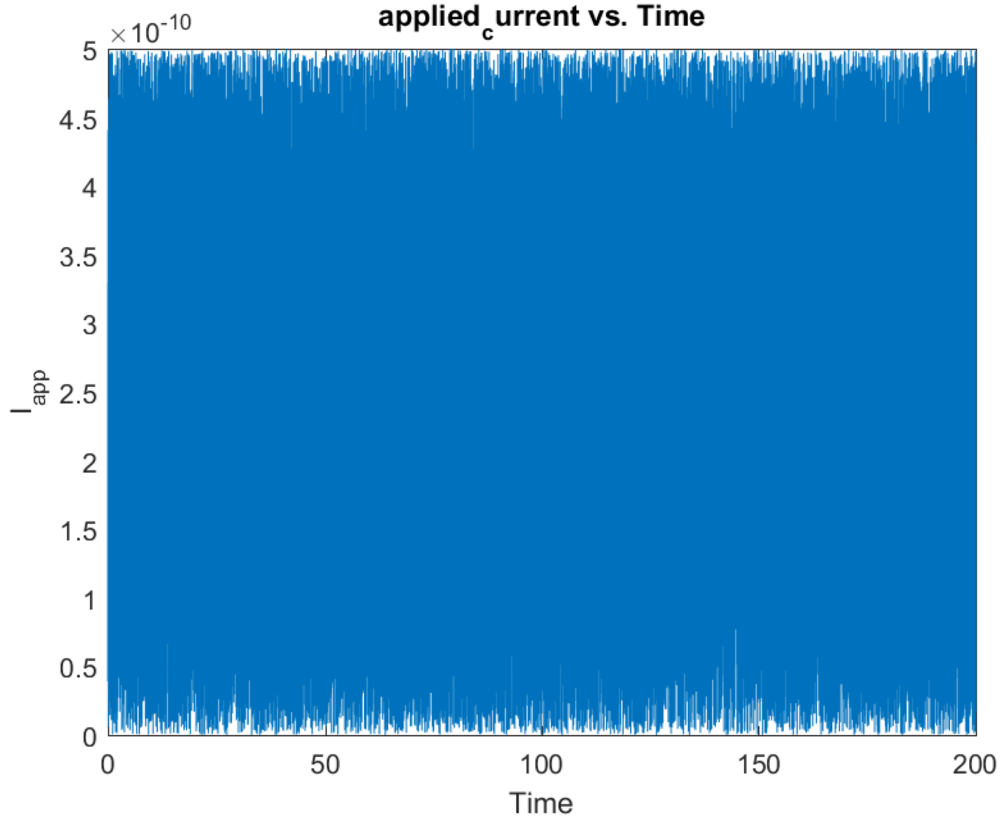
```
close all
clear
clc

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 )
        v(j) = reset_potential;
        I_sra(j) = I_sra(j) + b;
        spikes(j) = 1;
    end

    % this line shows the voltage over time, first part shows LIF
    % second part shows an exponential spiking term
    % third part shows 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;

   % the adaptation toward a steady state
    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'));
```

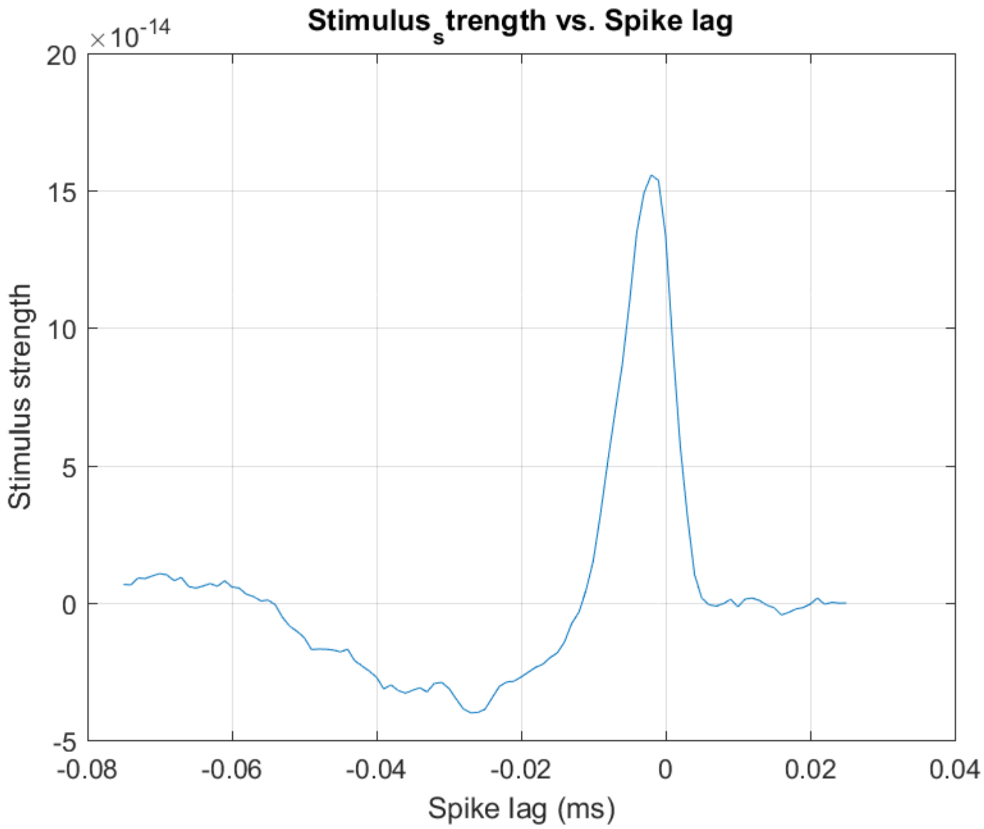



```
% Now downsample the stimulus and response to 1ms bins using the online
% function expandbin
new_dt = 0.001;
spikes = expandbin(spikes,dt,new_dt);
spikes(find(spikes)) = 1;
applied_current = expandbin(applied_current,dt,new_dt);
new_time_vector = 0:new_dt:total_time;

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_vect] = expandbin(old_vect, old_dt, new_dt)
length_old = length(old_vect);
scale_ratio = round(new_dt/old_dt);
length_new = round(length_old/scale_ratio);
new_vect = zeros(1,length_new);
tsteps = 50;
for k = 1:length(new_vect)
new_vect (k) = mean(old_vect ((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); % time points before zero
nplus = ceil(tplus/dt);    % time points after zero
nt = length(Iapp);      % original data set length
sum_I = zeros(1,nminus+nplus+1);  % STA
tcorr = -nminus*dt:dt:nplus*dt;   %time for STA
Iapp = Iapp - mean(Iapp); % Removes mean of applied current
spikeposition = find(spikes);  % Time bins for each spike
totalspikes = length(spikeposition)     % Total number of spikes
for spike = 1:totalspikes
```

```
    ispike = spikeposition(spike);      % the bin containing a spike
    imin = max(1,ispike-nminus);        % Bin to start measuring stimulus
    imax = min(nt,ispike+nplus);        % Bin to finish measuring

    for i = imin:imax
        sum_I(i-ispike+nminus+1) = sum_I(i-ispike+nminus+1) ...
            + Iapp(i)/totalspikes;
    end
end

% % average (the mean)STA
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 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.

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.

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

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.

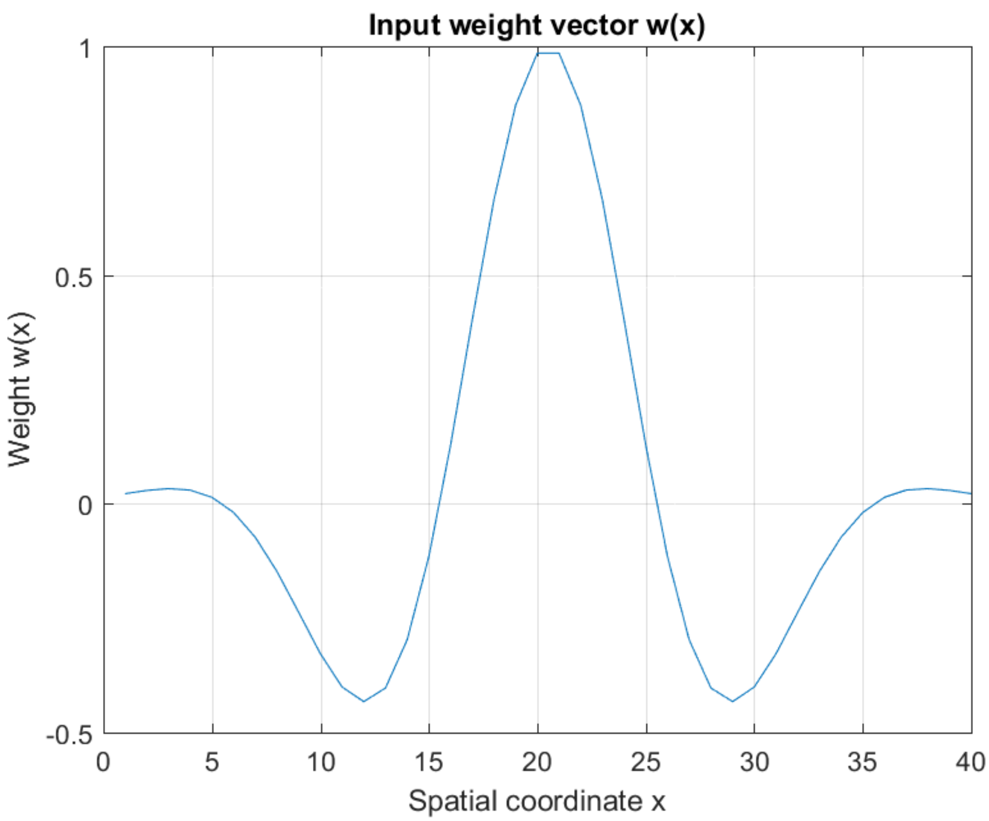
Adaptation strength (b): A larger adaptation strength will lead to a more pronounced adaptation effect in the STA, as it will cause a greater increase in the threshold potential after each spike.

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

B.

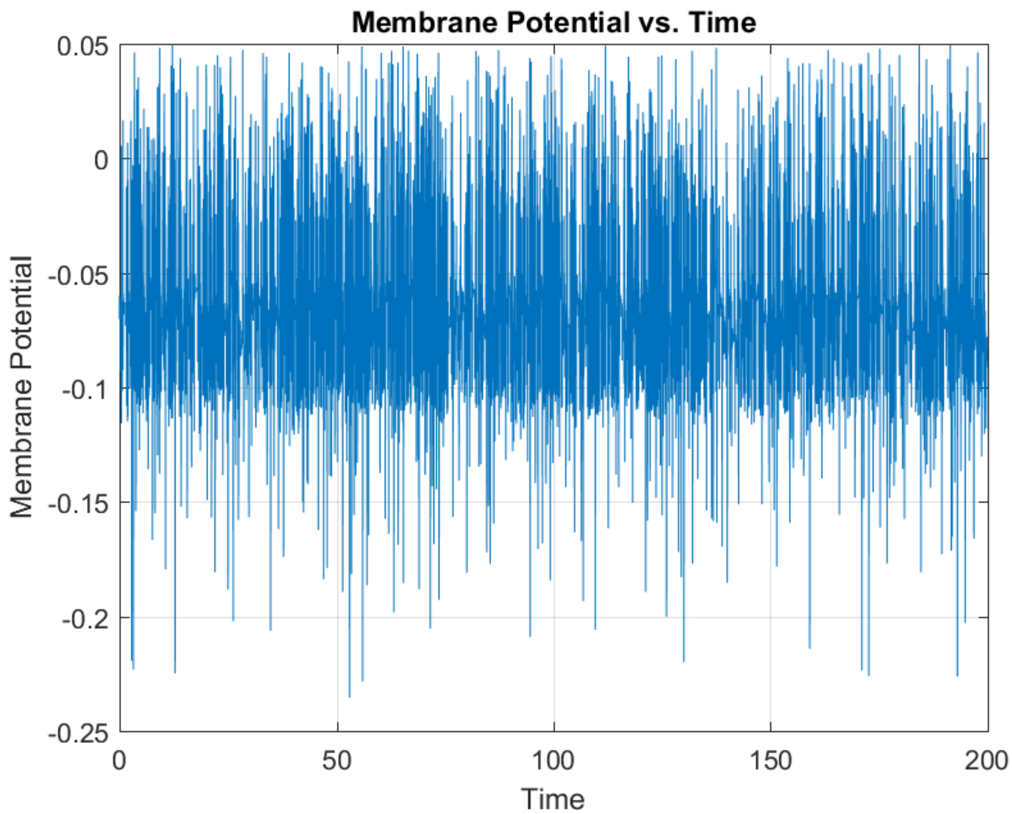
```
%B
Nsteps = 40000;              % Number of time-steps of distinct applied
currents
Nspatial = 40;               % Number of spatially distinct values per time-
step
```

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

x_max = 40; % Assuming x_max value
x_0 = 20.5;     % Assuming x_0 value
I_app = zeros(1, Nsteps);
for t = 1:length(I_app)-1
    sum_val = 0;
    for x = 1:x_max
        sum_val = sum_val + cos(4*pi*(x - x_0)/x_max) * exp(-16*((x -
x_0)/x_max)^2) * s(x, t); % Assuming s(x, t) is a function defined elsewhere
    end
    I_app(t) = sum_val;
end
x = 1:x_max;
w=cos(4*pi*(x - x_0)/x_max) .* exp(-16*((x - x_0)/x_max).^2);
% 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'));
```



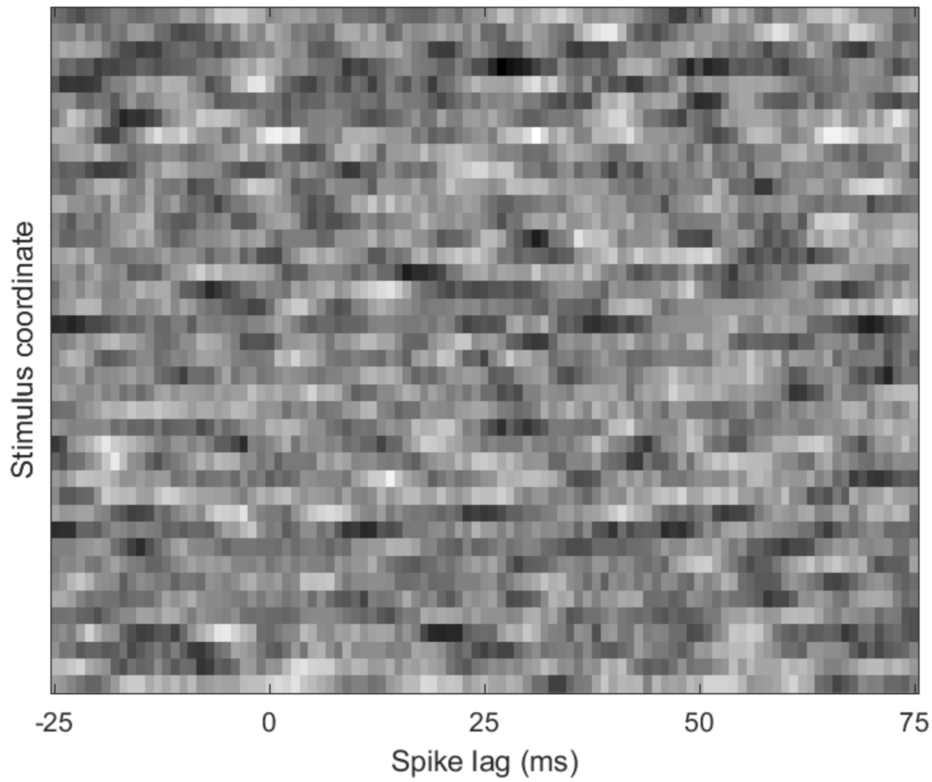

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%5
a = 40e-9;
b = 1e-9;
total_time = 200000e-3;
```

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

I_app_B = zeros(Nspatial,length(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 part_B.png'));
%%%%%%%%%%%%%%%%%%%%%%%%%%
new_dt = 0.001;                         % new time-bin of 1ms
spikes_B = expandbin(spikes_B,dt,new_dt);
spikes_B(find(spikes_B)) = 1;
```

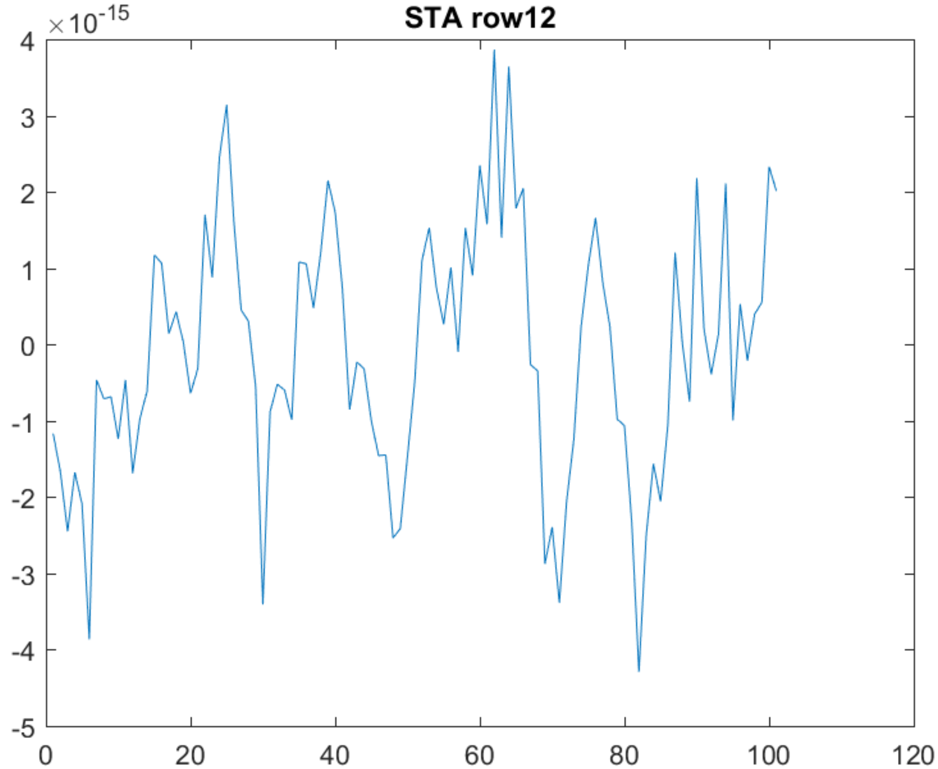



```
%%%%%%%%%%%%%%%
steplength = 0.005;
new_Nt = length(spikes_B);                    % New length of spike vector
newIapp = zeros(Nspatial,new_Nt);          % Define a new input vector
new_nstep_length = round(steplength/new_dt); % New number of 1ms bins per
step

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

    % generate the input vector as before, but with bins of 1ms, not of dt
    new_Iapp(:,istart:istop) = s(:,step)*ones(1,new_nstep_length);
end
%%%%%%%%%%


[sta, tcorr] = STA_spatial(new_Iapp, spikes_B, new_dt,new_nstep_length);

f6=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',[ ])
```

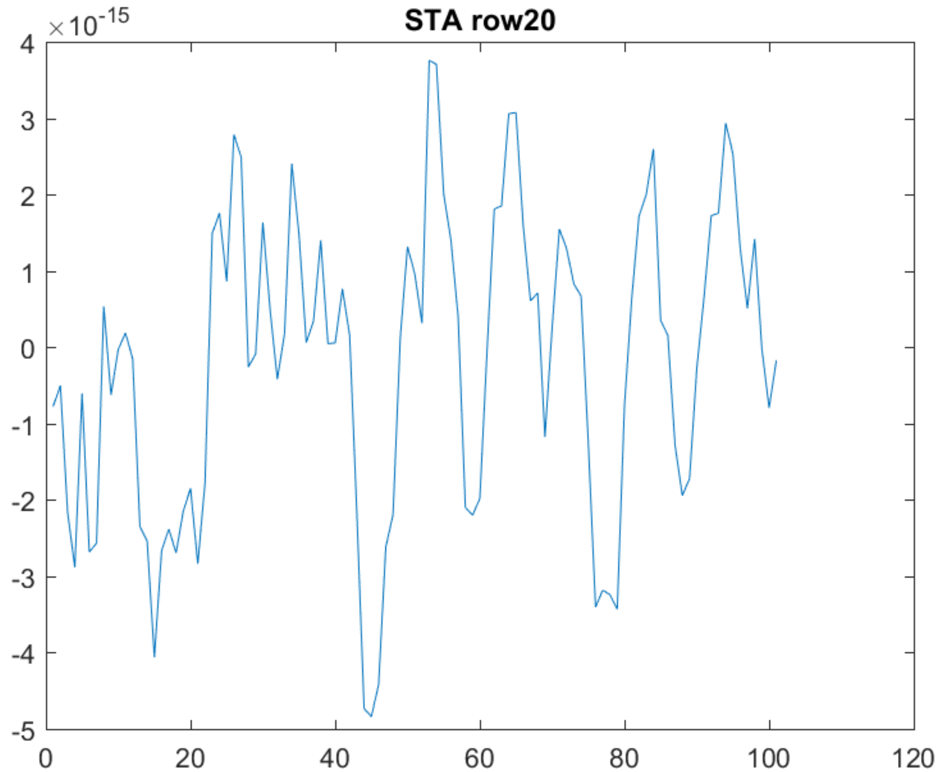
```
saveas(f6, sprintf('imagsec.png'));
```



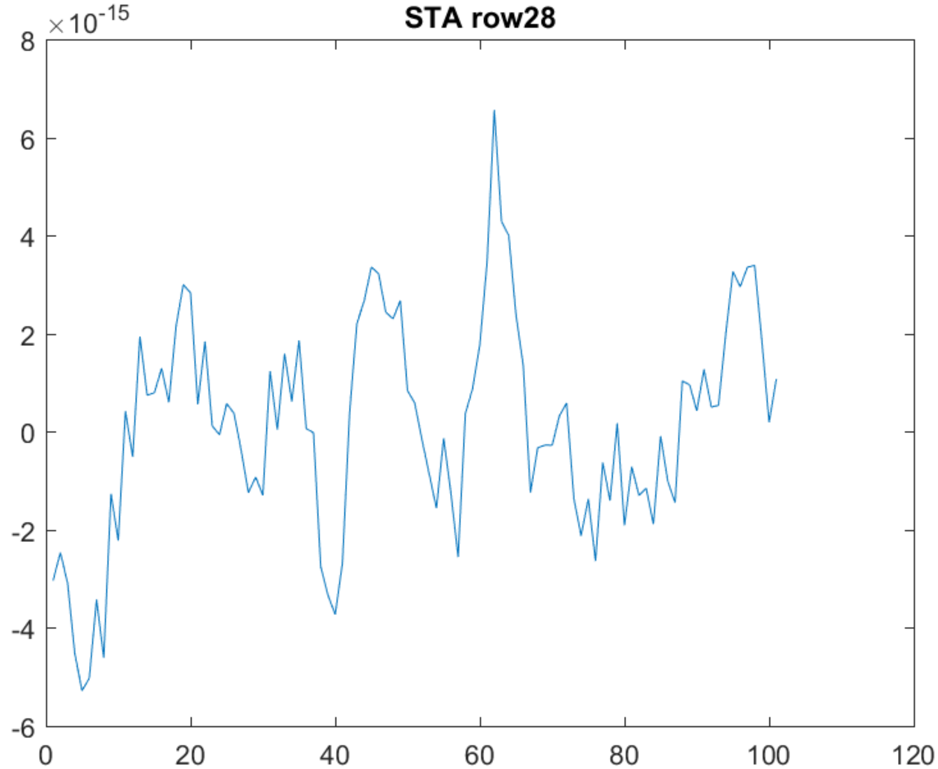

```
f7=figure();
plot(sta(12, :));saveas(f7, sprintf('row12.png'));
```



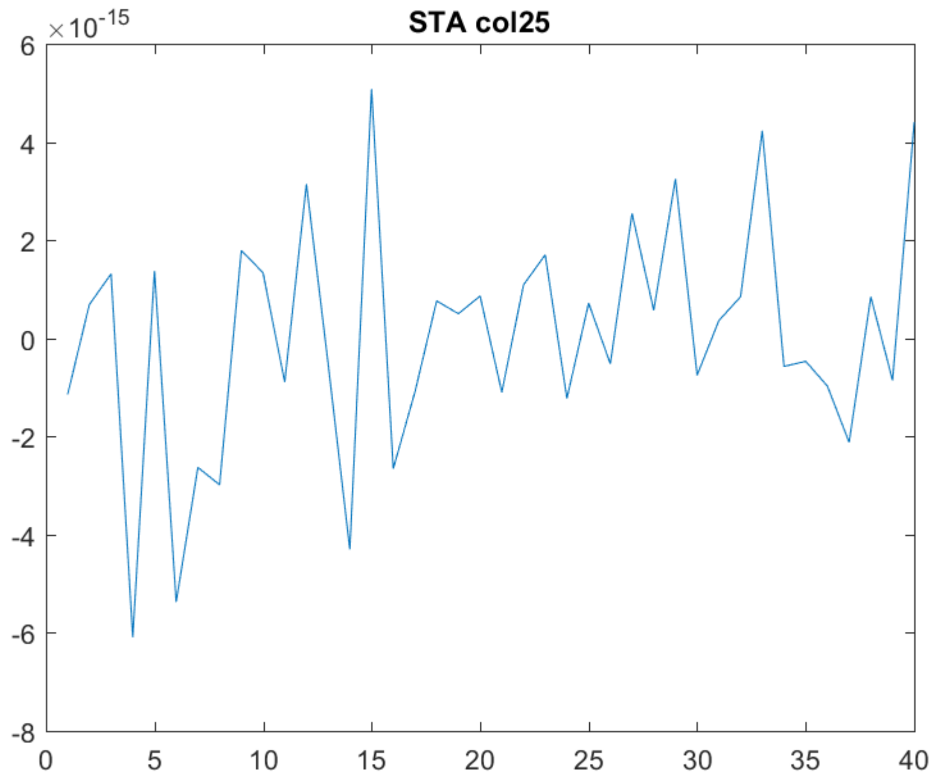

```
f8=figure();
plot(sta(20, :));saveas(f8, sprintf('row20.png'));
```



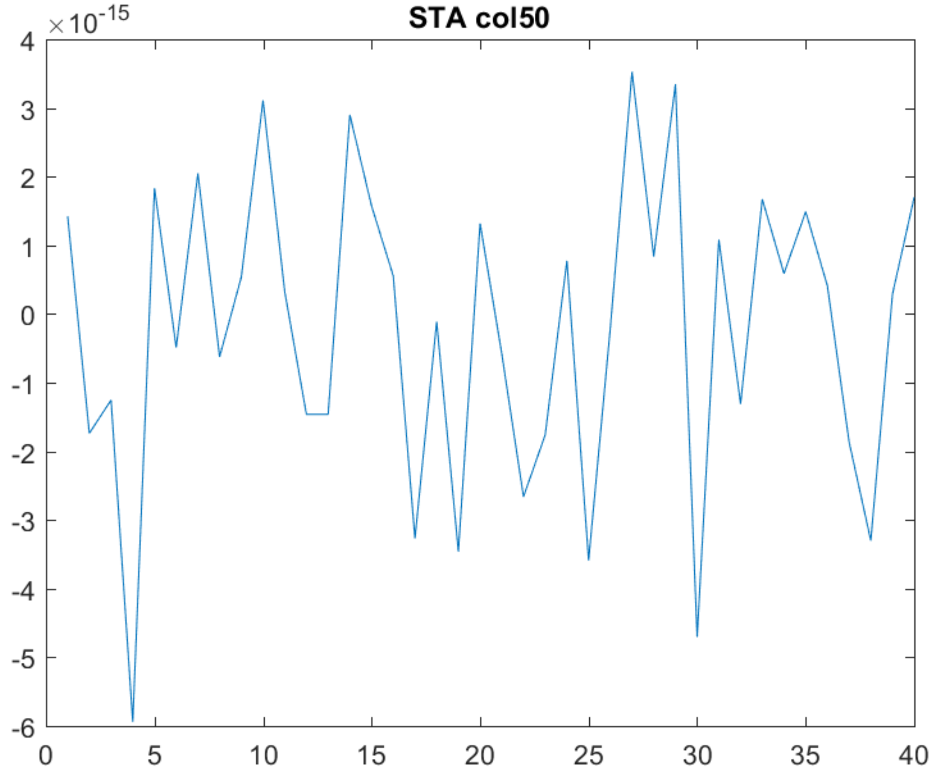

```
f9=figure();
plot(sta(28, :));saveas(f9, sprintf('row28.png'));
```



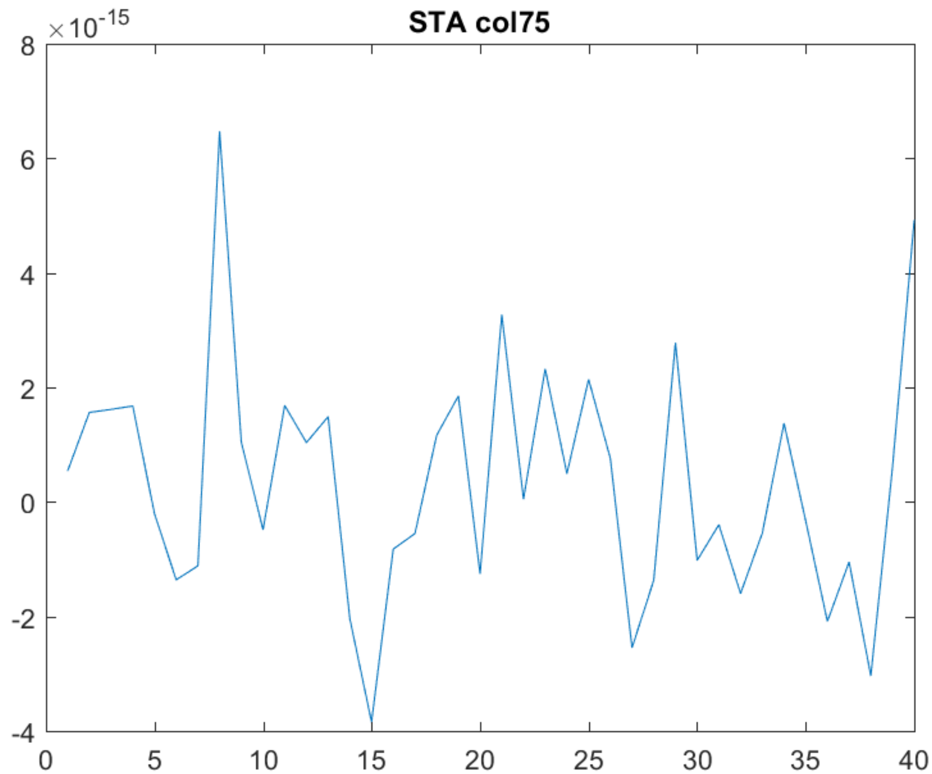

```
f10=figure();
plot(sta(:, 25));saveas(f10, sprintf('col25.png'));
```




```
f11=figure();
plot(sta(:, 50));saveas(f11, sprintf('col50.png'));
```




```
f12=figure();
plot(sta(:, 75));saveas(f12, sprintf('col75.png'));
```




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

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

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

  % Initialize STA and time vector
  sta = zeros(Nspace, nminus+nplus+1);
  tcorr = -nminus*dt:dt:nplus*dt;

  % Remove mean from each spatial bin
  for step = 1:Nt/new_nstep_length;                       % Loop through steps
of constant current
      step
    istart = (step-1)*new_nstep_length+1; % first time point with 1ms steps
    istop = step*new_nstep_length;        % last time point with 1ms steps
```

```
    istart
    istop
    % generate the input vector as before, but with bins of 1ms, not of dt
    stim_array(:,istart:istop) =  repmat((stim_array(:,istart)-
mean(stim_array, 2)),1,new_nstep_length);
end
%   for i=1:Nt
%    stim_array(:,i) = stim_array(:,i) - mean(stim_array, 2);
%   end
  % Find spike positions and total number of spikes
  spikeposition = find(spikes);
  totalspikes = length(spikeposition);

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

    % Accumulate stimulus values for each spatial bin
    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;

  % Return STA and time vector
  return;
end
```