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)

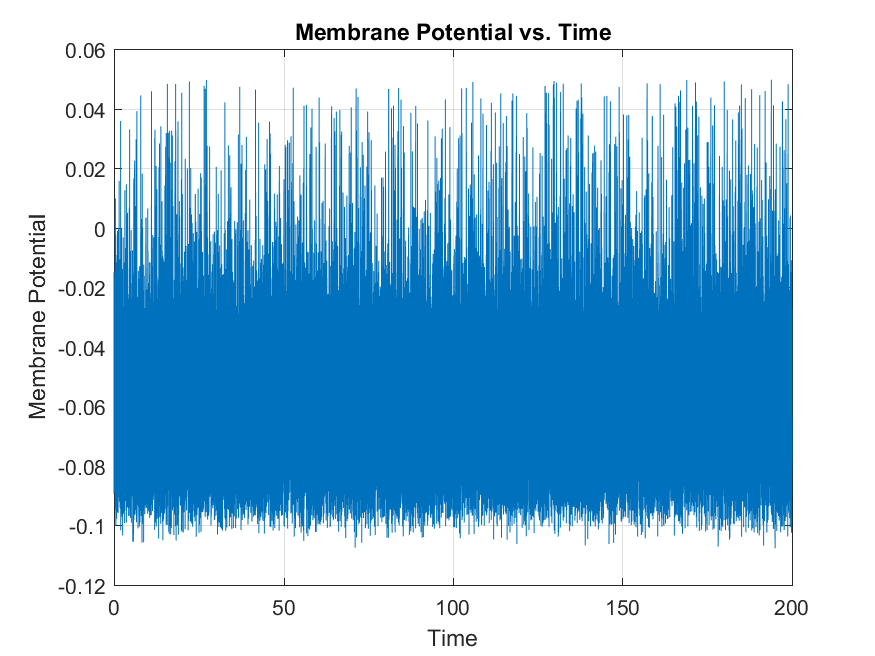
grid on

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))

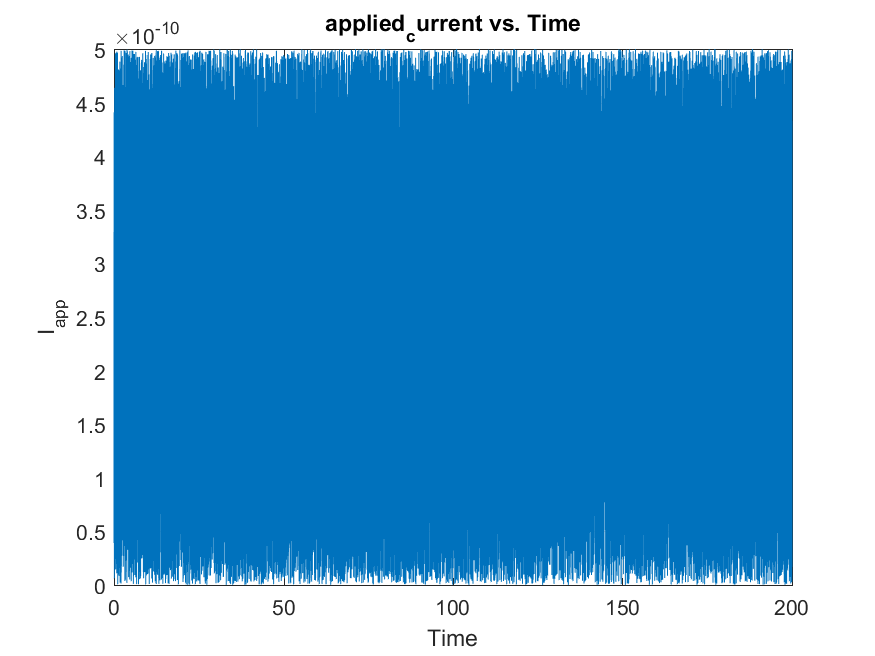
grid on

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)

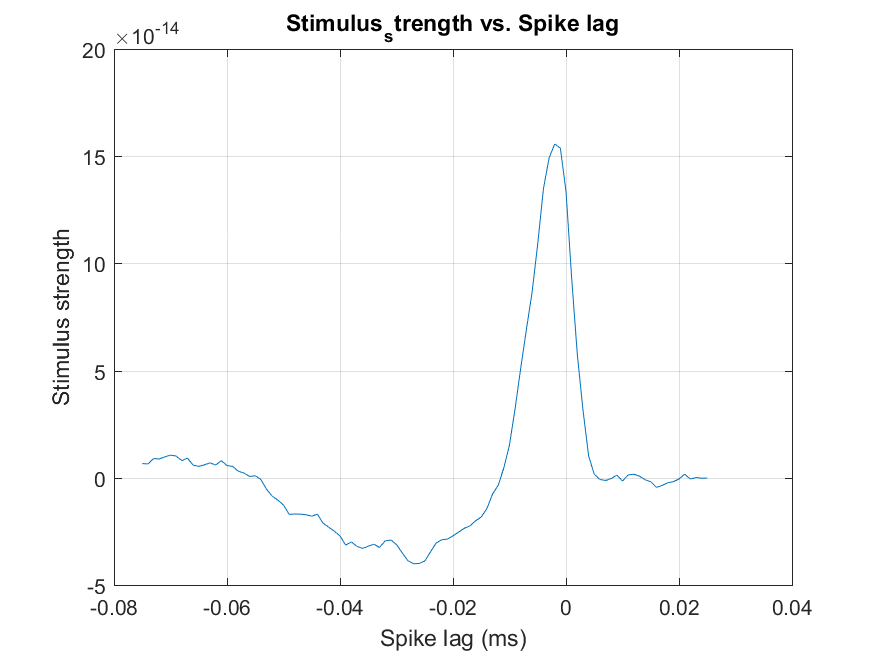
xlabel('Spike lag (ms)')

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);

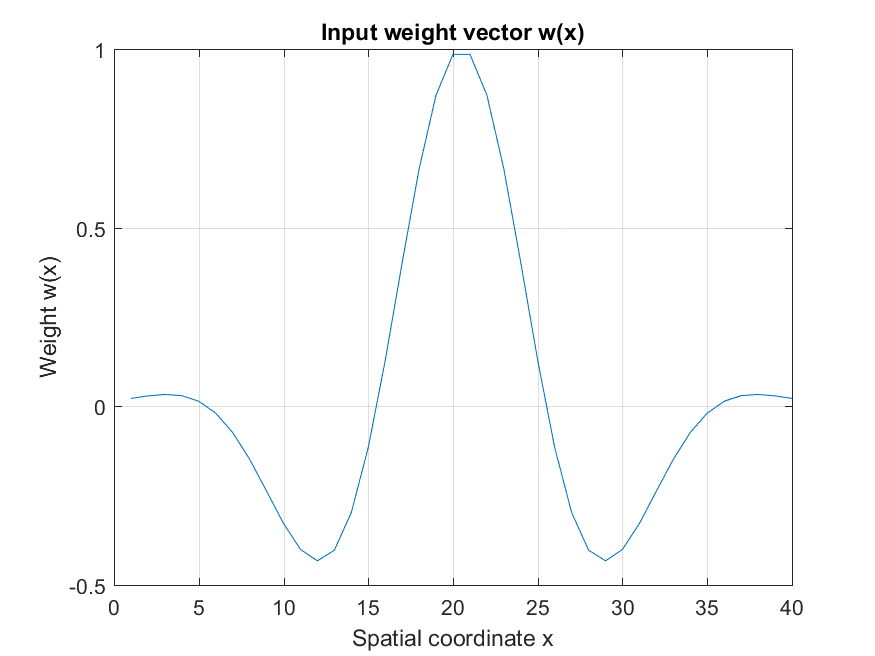
xlabel('Spatial coordinate x');

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');

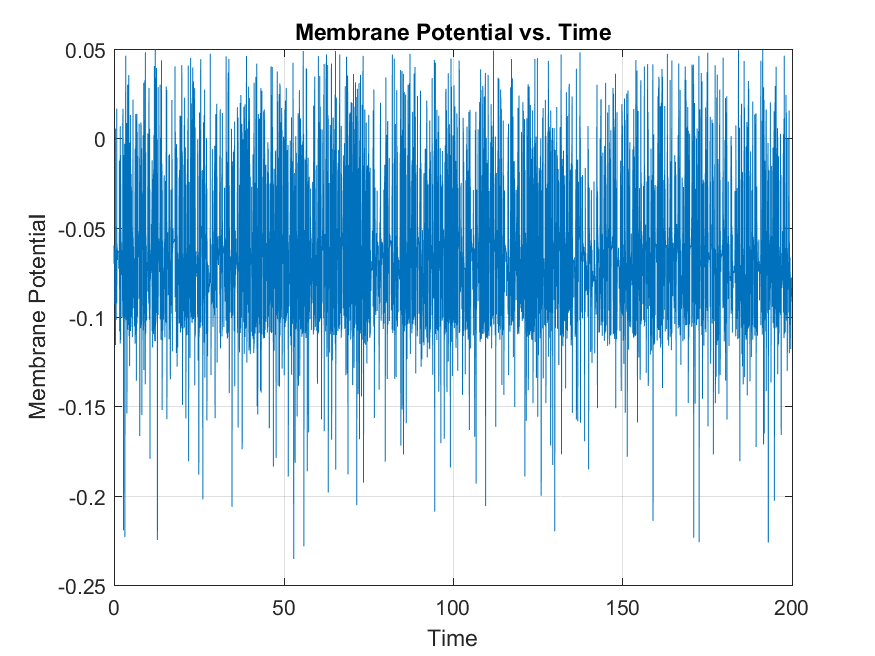
saveas(f5, sprintf('Membrane Potential vs. Time part\_B.png'));

%%%%%%%%%%%%%%%%%%%%%%%%%%%

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])

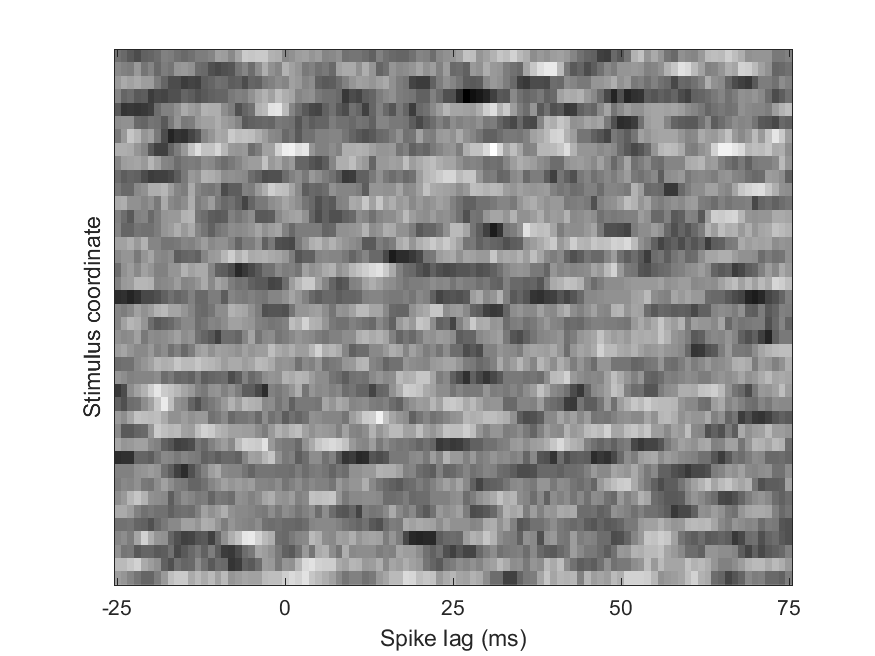
set(gca,'XTickLabel',{'-25' '0', '25', '50', '75'})

xlabel('Spike lag (ms)')

ylabel('Stimulus coordinate')

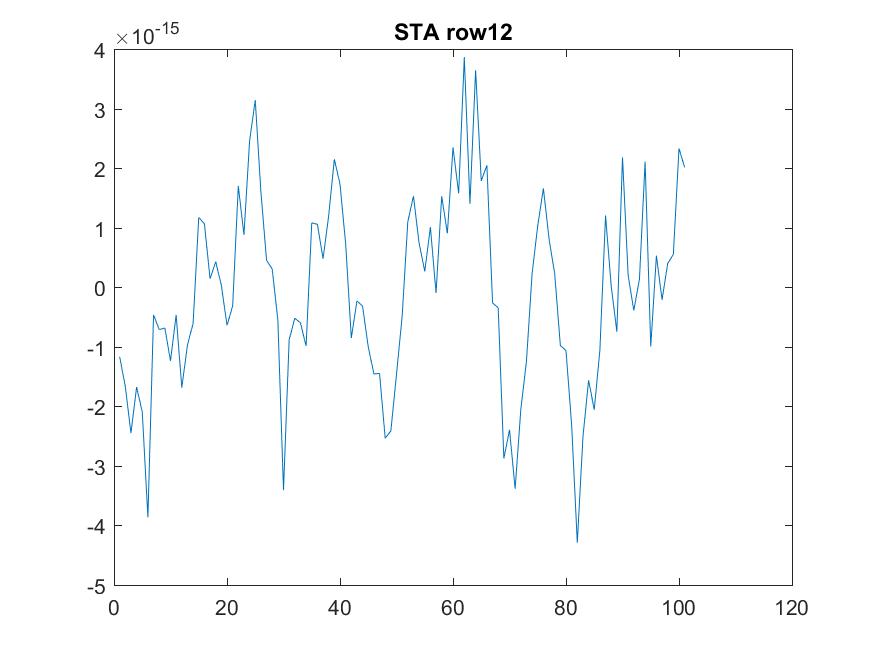
set(gca,'YTick',[ ])

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



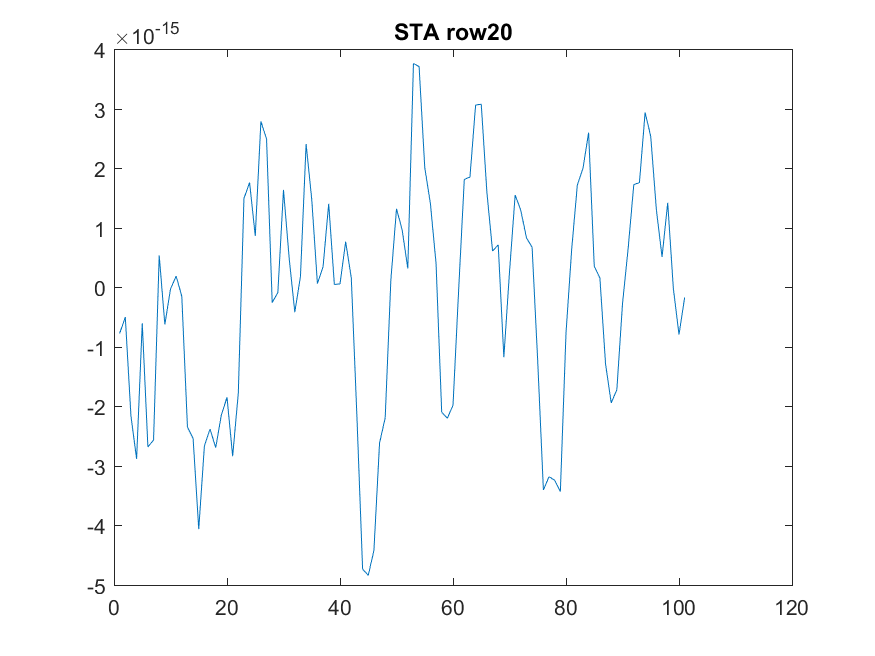
f7=figure();

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



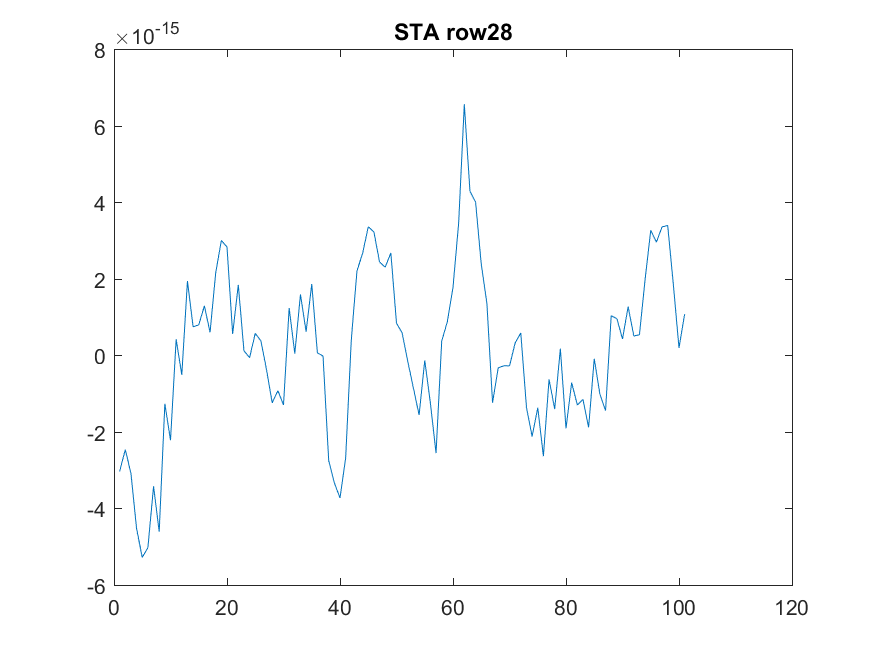
f8=figure();

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



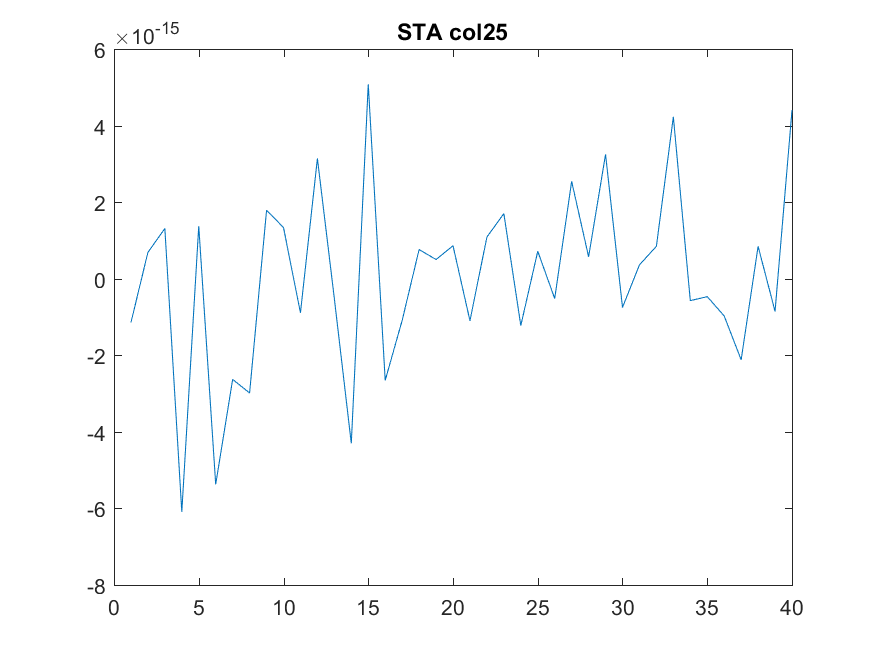
f9=figure();

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



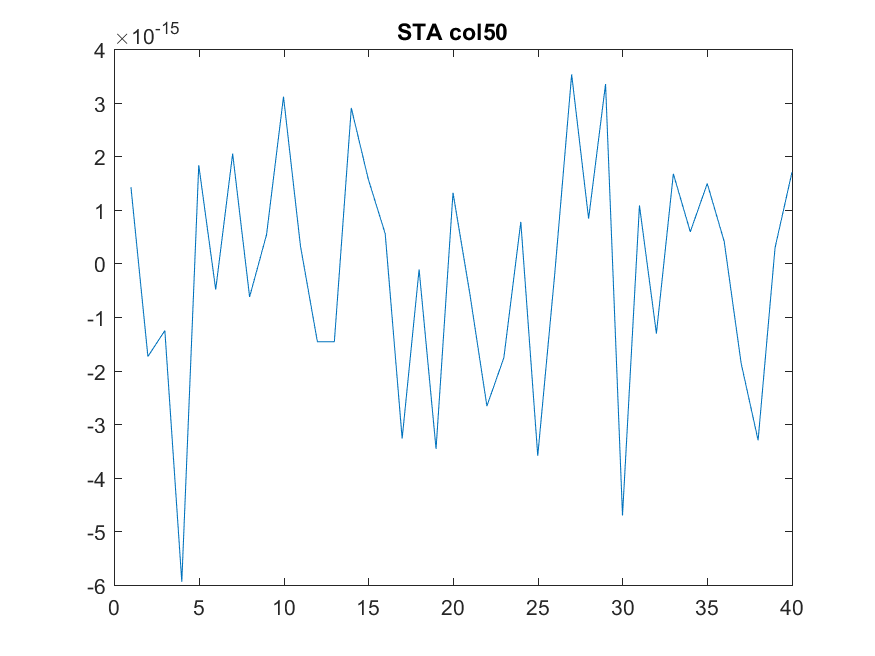
f10=figure();

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



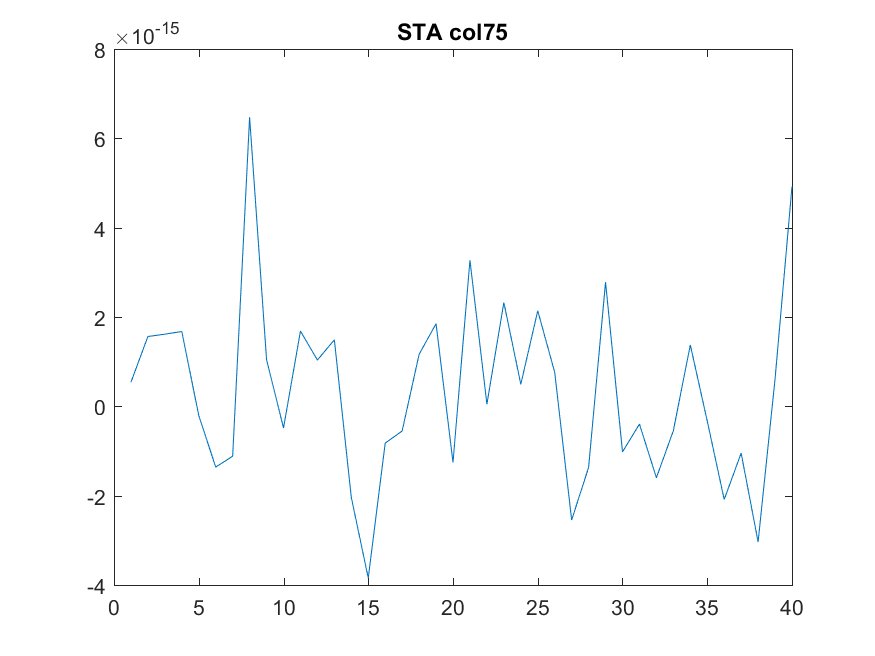
f11=figure();

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