Table of Contents

Learning Group

```
% Members: Meghan, Raymond

% DISCUSSION DATE: <05/13, Monday>, DURATION: <2h>

% DISCUSSION SUMMARY: We reviewed the approach for simulating T-type Ca2+
currents and post-inhibitory rebound bursting in a conductance-based neuron
model. This included parameter setup and generating applied currents for
different baseline and step values. We also examined the Connor-Stevens Type-
I neuron model, discussing its parameters and simulation process.

% CLARIFICATIONS RECEIVED: Clarifications were obtained on specific
parameters and their roles for both T-type Ca2+ currents simulation and
the Connor-Stevens neuron model, including conductance values and reversal
potentials. We also reviewed the steps for setting up the time vector and
storing simulation results.

% CONTRIBUTIONS: I contributed by clarifying the implementation details of
the simulation code, including parameter definition, time vector creation,
and result storage. Additionally, I explained the role of gating variables
and their influence on neuron model behavior during simulations.
```

1. Complete Tutorial 4.2 to simulate T-type Ca2+ currents and post-inhibitory rebound bursting in a conductance-based neuron model.

```
% Define parameters.
global leakageConductance sodiumConductance potassiumConductance
calciumConductance sodiumPotential potassiumPotential calciumPotential
leakagePotential membraneCapacitance

leakageConductance = 10e-9;
sodiumConductance = 3.6e-6;
potassiumConductance = 1.6e-6;
calciumConductance = 0.22e-6;
sodiumPotential = 55e-3;
potassiumPotential = -90e-3;
calciumPotential = 120e-3;
```

```
leakagePotential = -70e-3;
membraneCapacitance = 100e-12;

% Setup time vector.
global timeStep
timeStep = 0.01e-3;
timeVector = 0:timeStep:0.75;

% Create matrices for storing data.
spikeCount = zeros(21,21);
interspikeIntervals = zeros(21,21);

for row=1:21
    for col=1:21
        appliedCurrent = generateAppliedCurrent(-200e-12 + (col-1)*20e-12,
(row-1)*5e-12, timeVector);
        [voltage, sodiumGate, potassiumGate, calciumGate, spikeTimes] =
simulatePIR(timeVector, appliedCurrent);
        totalSpikes = sum(spikeTimes);
        spikeIndices = find(spikeTimes);
        minimumISI = 10.0;

        for idx=1:length(spikeIndices)-1
            isi = (spikeIndices(idx+1) - spikeIndices(idx)) * timeStep;
            if isi < minimumISI
                minimumISI = isi;
            end
        end

        if minimumISI == 10.0
            minimumISI = 0;
        end

        spikeCount(22-row, col) = totalSpikes;
        interspikeIntervals(22-row, col) = minimumISI;
    end
end

spikeCount
interspikeIntervals

figure1 = figure;
figure(figure1);
imagesc(spikeCount);
colorbarLabel = colorbar;
colorbarLabel.Label.String = 'Total Number of Spikes';
xlabel("Baseline Current (pA)");
ylabel("Current Step (pA)");
xticks([1:2:21]);
xticklabels({'-200','-160','-120','-80','-40','0','40','80','120','160','200'}
);
yticks([1:2:21]);
yticklabels({'100','90','80','70','60','50','40','30','20','10','0'});
title("Total Number of Spikes");
```

```
saveas(figure1, "spikeCount.png");

figure2 = figure;
figure(figure2);
imagesc(interspikeIntervals);
colorbarLabel = colorbar;
colorbarLabel.Label.String = 'Minimum ISI (seconds)';
xlabel("Baseline Current (pA)");
ylabel("Current Step (pA)");
xticks([1:2:21]);
xticklabels({'-200','-160','-120','-80','-40','0','40','80','120','160','200'}
);
yticks([1:2:21]);
yticklabels({'100','90','80','70','60','50','40','30','20','10','0'});
title("Minimum ISIs");
saveas(figure2, "interspikeIntervals.png");

% Plot qualitatively distinct behaviors.

% Plot A: baseline=-150, step=15.
appliedCurrent = generateAppliedCurrent(-150e-12, 15e-12, timeVector);
[voltage, sodiumGate, potassiumGate, calciumGate, spikeTimes] =
simulatePIR(timeVector, appliedCurrent);
totalSpikes = sum(spikeTimes);
spikeIndices = find(spikeTimes);
minimumISI = 10.0;

for idx=1:length(spikeIndices)-1
    isi = (spikeIndices(idx+1) - spikeIndices(idx)) * timeStep;
    if isi < minimumISI
        minimumISI = isi;
    end
end

if minimumISI == 10.0
    minimumISI = 0;
end

figureA = figure;
figure(figureA);
subplot(2,1,1);
plot(timeVector, appliedCurrent);
xlabel("Time (seconds)");
ylabel("Applied Current");
title("Applied Current vs. Time");
subplot(2,1,2);
plot(timeVector, voltage);
xlabel("Time (seconds)");
ylabel("Membrane Potential");
title(sprintf("Plot A. Spikes = %d, Min ISI = %f", totalSpikes, minimumISI));
saveas(figureA, "PlotA.png");

% Plot B: baseline=-80, step=90.
appliedCurrent = generateAppliedCurrent(-80e-12, 90e-12, timeVector);
```

```
[voltage, sodiumGate, potassiumGate, calciumGate, spikeTimes] =
simulatePIR(timeVector, appliedCurrent);
totalSpikes = sum(spikeTimes);
spikeIndices = find(spikeTimes);
minimumISI = 10.0;

for idx=1:length(spikeIndices)-1
    isi = (spikeIndices(idx+1) - spikeIndices(idx)) * timeStep;
    if isi < minimumISI
        minimumISI = isi;
    end
end

if minimumISI == 10.0
    minimumISI = 0;
end

figureB = figure;
figure(figureB);
subplot(2,1,1);
plot(timeVector, appliedCurrent);
xlabel("Time (seconds)");
ylabel("Applied Current");
title("Applied Current vs. Time");
subplot(2,1,2);
plot(timeVector, voltage);
xlabel("Time (seconds)");
ylabel("Membrane Potential");
title(sprintf("Plot B. Spikes = %d, Min ISI = %f", totalSpikes, minimumISI));
saveas(figureB, "PlotB.png");

% Plot C: baseline=-20, step=45.
appliedCurrent = generateAppliedCurrent(-20e-12, 45e-12, timeVector);
[voltage, sodiumGate, potassiumGate, calciumGate, spikeTimes] =
simulatePIR(timeVector, appliedCurrent);
totalSpikes = sum(spikeTimes);
spikeIndices = find(spikeTimes);
minimumISI = 10.0;

for idx=1:length(spikeIndices)-1
    isi = (spikeIndices(idx+1) - spikeIndices(idx)) * timeStep;
    if isi < minimumISI
        minimumISI = isi;
    end
end

if minimumISI == 10.0
    minimumISI = 0;
end

figureC = figure;
figure(figureC);
subplot(2,1,1);
plot(timeVector, appliedCurrent);
```

```
xlabel("Time (seconds)");
ylabel("Applied Current");
title("Applied Current vs. Time");
subplot(2,1,2);
plot(timeVector, voltage);
xlabel("Time (seconds)");
ylabel("Membrane Potential");
title(sprintf("Plot C. Spikes = %d, Min ISI = %f", totalSpikes, minimumISI));
saveas(figureC, "PlotC.png");

% Plot D: baseline=100, step=45.
appliedCurrent = generateAppliedCurrent(100e-12, 45e-12, timeVector);
[voltage, sodiumGate, potassiumGate, calciumGate, spikeTimes] =
simulatePIR(timeVector, appliedCurrent);
totalSpikes = sum(spikeTimes);
spikeIndices = find(spikeTimes);
minimumISI = 10.0;

for idx=1:length(spikeIndices)-1
    isi = (spikeIndices(idx+1) - spikeIndices(idx)) * timeStep;
    if isi < minimumISI
        minimumISI = isi;
    end
end

if minimumISI == 10.0
    minimumISI = 0;
end

figureD = figure;
figure(figureD);
subplot(2,1,1);
plot(timeVector, appliedCurrent);
xlabel("Time (seconds)");
ylabel("Applied Current");
title("Applied Current vs. Time");
subplot(2,1,2);
plot(timeVector, voltage);
xlabel("Time (seconds)");
ylabel("Membrane Potential");
title(sprintf("Plot D. Spikes = %d, Min ISI = %f", totalSpikes, minimumISI));
saveas(figureD, "PlotD.png");

% Plot E: baseline=180, step=45.
appliedCurrent = generateAppliedCurrent(180e-12, 45e-12, timeVector);
[voltage, sodiumGate, potassiumGate, calciumGate, spikeTimes] =
simulatePIR(timeVector, appliedCurrent);
totalSpikes = sum(spikeTimes);
spikeIndices = find(spikeTimes);
minimumISI = 10.0;

for idx=1:length(spikeIndices)-1
    isi = (spikeIndices(idx+1) - spikeIndices(idx)) * timeStep;
    if isi < minimumISI
```

```
        minimumISI = isi;
    end
end

if minimumISI == 10.0
    minimumISI = 0;
end

figureE = figure;
figure(figureE);
subplot(2,1,1);
plot(timeVector, appliedCurrent);
xlabel("Time (seconds)");
ylabel("Applied Current");
title("Applied Current vs. Time");
subplot(2,1,2);
plot(timeVector, voltage);
xlabel("Time (seconds)");
ylabel("Membrane Potential");
title(sprintf("Plot E. Spikes = %d, Min ISI = %f", totalSpikes, minimumISI));
saveas(figureE, "PlotE.png");

% Function Definitions:

function [voltage, sodiumGate, potassiumGate, calciumGate, spikeTimes] =
simulatePIR(timeVector, appliedCurrent, initialVoltage, initialSodiumGate,
initialPotassiumGate, initialCalciumGate)
% Simulates the thalamocortical neuron model with a T-type calcium current
given the input time vector and
% applied current.

global timeStep leakageConductance sodiumConductance potassiumConductance
calciumConductance sodiumPotential potassiumPotential calciumPotential
leakagePotential membraneCapacitance

% Default parameters if not inputted.
if (~exist('initialVoltage'))
    initialVoltage = leakagePotential;
end
if (~exist('initialSodiumGate'))
    initialSodiumGate = 0;
end
if (~exist('initialPotassiumGate'))
    initialPotassiumGate = 0;
end
if (~exist('initialCalciumGate'))
    initialCalciumGate = 0;
end

% Setup vectors.
voltage = zeros(1, length(timeVector));
sodiumGate = zeros(1, length(timeVector));
potassiumGate = zeros(1, length(timeVector));
calciumGate = zeros(1, length(timeVector));
```

```
spikeTimes = zeros(1, length(timeVector));

voltage(1) = initialVoltage;
sodiumGate(1) = initialSodiumGate;
potassiumGate(1) = initialPotassiumGate;
calciumGate(1) = initialCalciumGate;

% Spike detection parameters.
spikeDetection = 0;
spikeThreshold = 0.0;
spikeResetThreshold = -0.06;

% Simulation loop.
for idx = 1:(length(timeVector)-1)
    if voltage(idx) > spikeThreshold
        if spikeDetection == 0
            spikeTimes(idx) = 1;
            spikeDetection = 1;
        end
    end

    if voltage(idx) < spikeResetThreshold
        spikeDetection = 0;
    end

    % Update voltage.
    alpha = ((10^5)*(voltage(idx) + 0.035))/(1 - exp(-100*(voltage(idx)
+0.035)));
    beta = 4000*exp((-(voltage(idx)+0.06))/(0.018));
    sodiumActivation = alpha/(alpha+beta);
    calciumActivation = 1/(1 + exp((-(voltage(idx)+0.052))/(0.0074)));
    leakageTerm = leakageConductance*(leakagePotential-voltage(idx));
    sodiumTerm =
sodiumConductance*(sodiumActivation^3)*sodiumGate(idx)*(sodiumPotential-
voltage(idx));
    potassiumTerm =
potassiumConductance*(potassiumGate(idx)^4)*(potassiumPotential-voltage(idx));
    calciumTerm =
calciumConductance*(calciumActivation^2)*calciumGate(idx)*(calciumPotential-
voltage(idx));
    voltage(idx+1) = voltage(idx) + (timeStep/
membraneCapacitance)*(leakageTerm+sodiumTerm+potassiumTerm+calciumTerm+applied
Current(idx));

    % Update sodium gate.
    alpha = 350*exp(-50*(voltage(idx)+0.058));
    beta = 5000/(1 + exp(-100*(voltage(idx)+0.028)));
    sodiumGate(idx+1) = sodiumGate(idx) + timeStep*(alpha*(1-sodiumGate(idx))
- beta*sodiumGate(idx));

    % Update potassium gate.
    alpha = ((5*(10^4))*(voltage(idx)+0.034))/(1 - exp(-100*(voltage(idx)
+0.034)));
    beta = 625*exp(-12.5*(voltage(idx)+0.044));
```

```
    potassiumGate(idx+1) = potassiumGate(idx) + timeStep*(alpha*(1-
potassiumGate(idx)) - beta*potassiumGate(idx));

    % Update calcium gate.
    calciumSteadyState = 1/(1 + exp(500*(voltage(idx)+0.076)));
    if voltage(idx) < -0.080
        calciumTimeConstant = 0.001*exp(15*(voltage(idx)+0.467));
    else
        calciumTimeConstant = 0.028 + 0.001*exp((-(voltage(idx)+0.022))/
(0.0105));
    end
    calciumGate(idx+1) = calciumGate(idx) + timeStep*((calciumSteadyState-
calciumGate(idx))/calciumTimeConstant);
end
end

function [appliedCurrent] = generateAppliedCurrent(baseline, step, timeVector)
% Returns a vector for applied current given input baseline and step
% currents.
global timeStep
    appliedCurrent = zeros(1, length(timeVector));
    firstThird = floor(length(timeVector)/3);
    secondThird = 2*firstThird;
    appliedCurrent(1:firstThird) = baseline;
    appliedCurrent(firstThird+1:secondThird) = baseline+step;
    appliedCurrent(secondThird+1:end) = baseline;
end
```

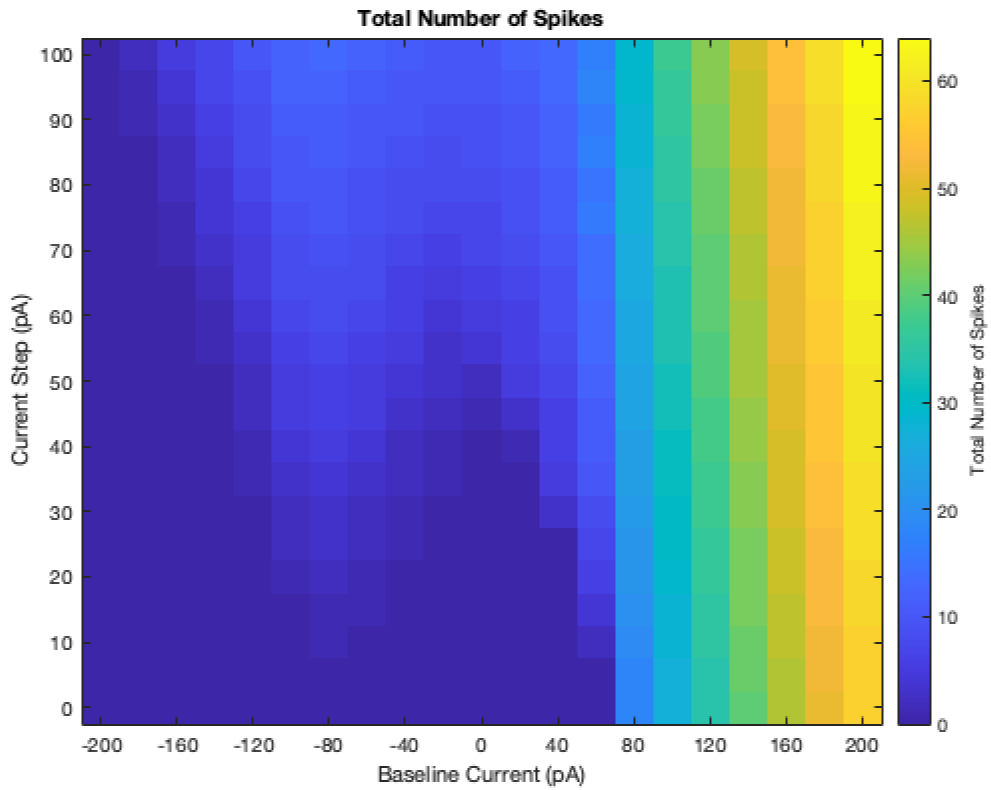
```
spikeCount =

  Columns 1 through 13

     0     2     5     7    10    12    13    12    11    10    10    12    13
     0     1     4     7     9    12    12    11    10    10    10    11    13
     0     1     3     6     8    11    11    10    10     9     9    10    12
     0     0     2     5     8    10    11    10     9     8     9    10    12
     0     0     2     4     7    10    10     9     8     8     8     9    11
     0     0     1     4     6     9    10     9     8     7     7     9    11
     0     0     1     3     5     8     9     8     7     6     7     8    10
     0     0     0     2     5     8     8     8     6     5     6     7     9
     0     0     0     1     4     7     8     7     6     4     5     6     9
     0     0     0     1     3     6     7     6     5     3     4     6     8
     0     0     0     0     2     5     6     5     4     3     2     5     7
     0     0     0     0     2     5     6     5     3     2     1     3     6
     0     0     0     0     1     4     5     4     2     1     0     1     6
     0     0     0     0     1     3     4     3     2     1     0     0     5
     0     0     0     0     0     2     3     2     1     0     0     0     3
     0     0     0     0     0     2     3     2     1     0     0     0     0
     0     0     0     0     0     1     2     1     0     0     0     0     0
     0     0     0     0     0     0     1     1     0     0     0     0     0
     0     0     0     0     0     0     1     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
```

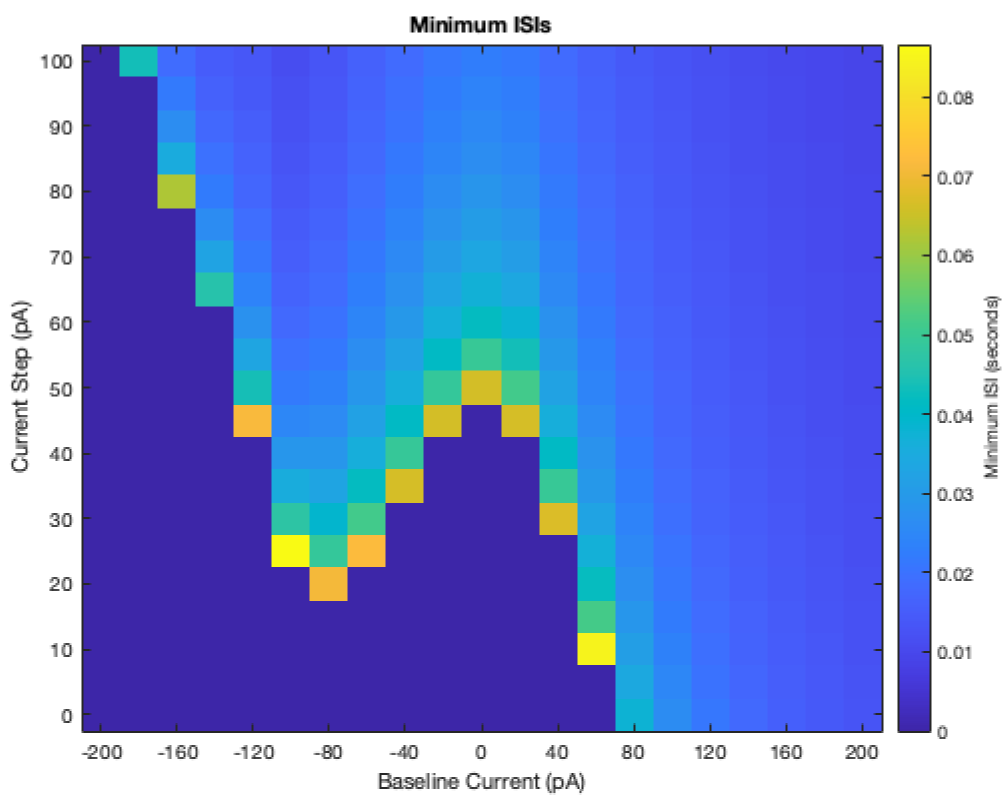
```
  Columns 14 through 21

    17    29    37    43    49    54    59    64
    18    29    36    43    48    54    59    64
    16    28    36    42    48    53    58    63
    17    28    35    42    48    53    58    63
    15    27    35    41    47    52    58    63
    16    27    34    41    47    52    57    62
    14    26    34    40    46    52    57    62
    14    26    33    40    46    51    57    62
    13    25    33    40    45    51    56    61
    13    25    33    39    45    51    56    61
    12    24    32    39    45    50    55    60
    11    24    32    38    44    50    55    60
    11    23    31    38    44    49    55    60
    10    23    31    37    43    49    54    59
     8    22    30    37    43    49    54    59
     7    21    30    36    42    48    53    59
     6    21    29    36    42    48    53    58
     4    20    28    35    42    47    53    58
     2    19    28    35    41    47    52    57
     0    18    27    34    41    46    52    57
     0    18    27    34    40    46    51    57


interspikeIntervals =

  Columns 1 through 7

         0    0.0436    0.0189    0.0149    0.0136    0.0114    0.0133
         0         0    0.0219    0.0162    0.0143    0.0119    0.0138
         0         0    0.0265    0.0177    0.0152    0.0125    0.0144
         0         0    0.0351    0.0196    0.0163    0.0131    0.0151
         0         0    0.0622    0.0223    0.0176    0.0138    0.0158
         0         0         0    0.0262    0.0192    0.0146    0.0167
         0         0         0    0.0326    0.0212    0.0155    0.0176
         0         0         0    0.0458    0.0239    0.0166    0.0188
         0         0         0         0    0.0277    0.0180    0.0201
         0         0         0         0    0.0334    0.0197    0.0216
         0         0         0         0    0.0438    0.0219    0.0235
         0         0         0         0    0.0717    0.0248    0.0259
         0         0         0         0         0    0.0289    0.0290
         0         0         0         0         0    0.0353    0.0331
         0         0         0         0         0    0.0473    0.0391
         0         0         0         0         0    0.0866    0.0490
         0         0         0         0         0         0    0.0708
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0

  Columns 8 through 14
```

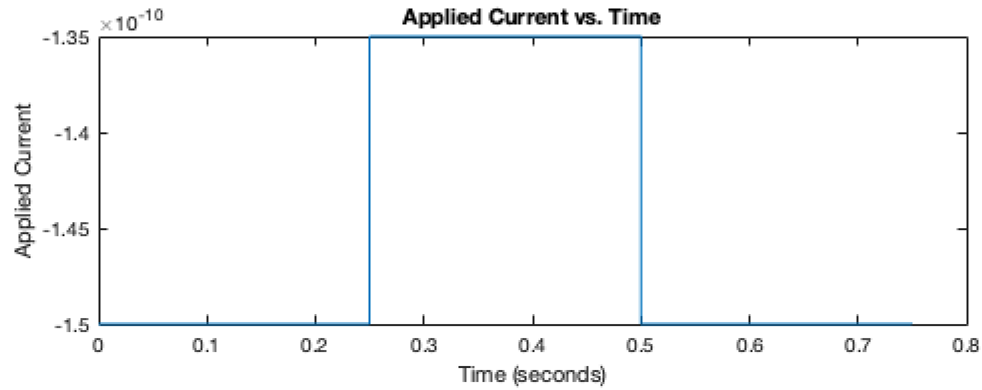
0.0160	0.0185	0.0210	0.0227	0.0214	0.0184	0.0161
0.0166	0.0194	0.0221	0.0238	0.0224	0.0191	0.0167
0.0174	0.0203	0.0232	0.0252	0.0236	0.0199	0.0172
0.0182	0.0213	0.0246	0.0267	0.0249	0.0208	0.0178
0.0191	0.0225	0.0261	0.0286	0.0266	0.0217	0.0184
0.0202	0.0239	0.0279	0.0308	0.0286	0.0229	0.0192
0.0214	0.0255	0.0301	0.0335	0.0309	0.0241	0.0199
0.0228	0.0273	0.0329	0.0370	0.0339	0.0256	0.0209
0.0245	0.0296	0.0364	0.0419	0.0380	0.0275	0.0218
0.0265	0.0324	0.0413	0.0496	0.0435	0.0295	0.0230
0.0289	0.0361	0.0489	0.0660	0.0512	0.0323	0.0242
0.0320	0.0412	0.0662	0	0.0664	0.0360	0.0258
0.0361	0.0492	0	0	0	0.0412	0.0278
0.0419	0.0665	0	0	0	0.0496	0.0298
0.0512	0	0	0	0	0.0672	0.0327
0.0723	0	0	0	0	0	0.0366
0	0	0	0	0	0	0.0422
0	0	0	0	0	0	0.0515
0	0	0	0	0	0	0.0848
0	0	0	0	0	0	0
0	0	0	0	0	0	0

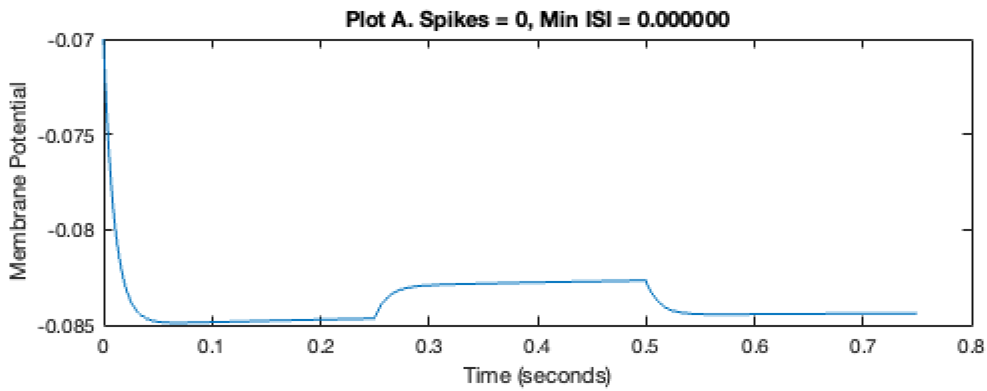
Columns 15 through 21

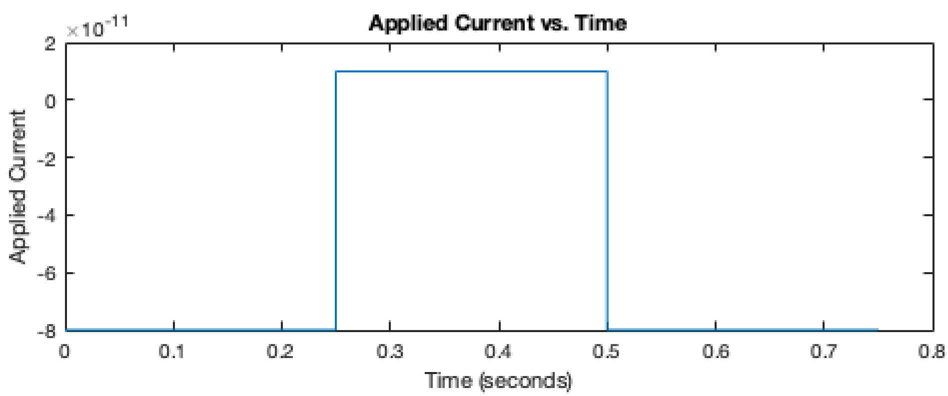
0.0143	0.0130	0.0120	0.0112	0.0105	0.0099	0.0093
0.0147	0.0133	0.0122	0.0114	0.0106	0.0100	0.0095
0.0151	0.0136	0.0125	0.0116	0.0108	0.0101	0.0096
0.0155	0.0139	0.0127	0.0118	0.0110	0.0103	0.0097
0.0160	0.0143	0.0130	0.0120	0.0112	0.0105	0.0099
0.0165	0.0147	0.0133	0.0122	0.0114	0.0106	0.0100
0.0170	0.0151	0.0136	0.0125	0.0116	0.0108	0.0101
0.0176	0.0155	0.0139	0.0127	0.0118	0.0110	0.0103
0.0182	0.0159	0.0143	0.0130	0.0120	0.0112	0.0104
0.0189	0.0164	0.0146	0.0133	0.0122	0.0113	0.0106
0.0197	0.0169	0.0150	0.0136	0.0125	0.0115	0.0108
0.0205	0.0175	0.0155	0.0139	0.0127	0.0118	0.0110
0.0215	0.0181	0.0159	0.0143	0.0130	0.0120	0.0111
0.0225	0.0188	0.0164	0.0146	0.0133	0.0122	0.0113
0.0237	0.0195	0.0169	0.0150	0.0136	0.0125	0.0115
0.0252	0.0204	0.0175	0.0154	0.0139	0.0127	0.0118
0.0268	0.0213	0.0181	0.0159	0.0143	0.0130	0.0120
0.0288	0.0223	0.0188	0.0163	0.0146	0.0133	0.0122
0.0312	0.0235	0.0195	0.0169	0.0150	0.0136	0.0125
0.0343	0.0249	0.0203	0.0174	0.0154	0.0139	0.0127
0.0379	0.0263	0.0211	0.0180	0.0158	0.0142	0.0130

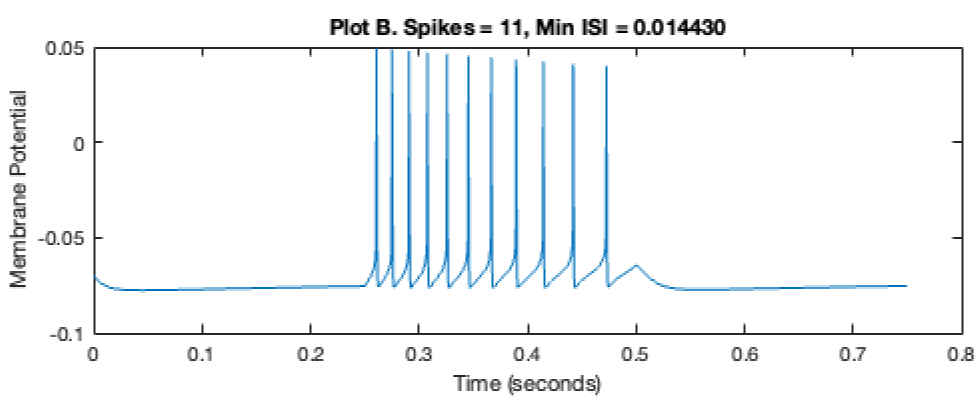

Total Number of Spikes
Current Step (pA)
Baseline Current (pA)
Total Number of Spikes
-200
-160
-120
-80
-40
0
40
80
120
160
200
0
10
20
30
40
50
60
70
80
90
100
0
10
20
30
40
50
60

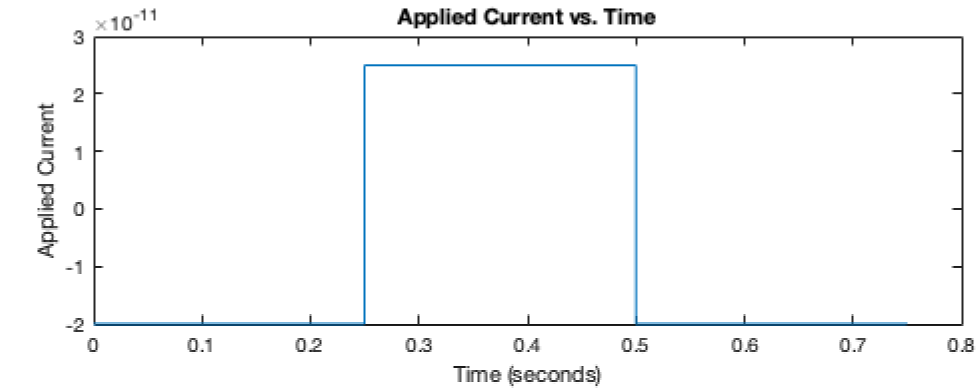

Minimum ISIs
Current Step (pA)
Baseline Current (pA)
Minimum ISI (seconds)
-200
-160
-120
-80
-40
0
40
80
120
160
200
0
10
20
30
40
50
60
70
80
90
100
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08

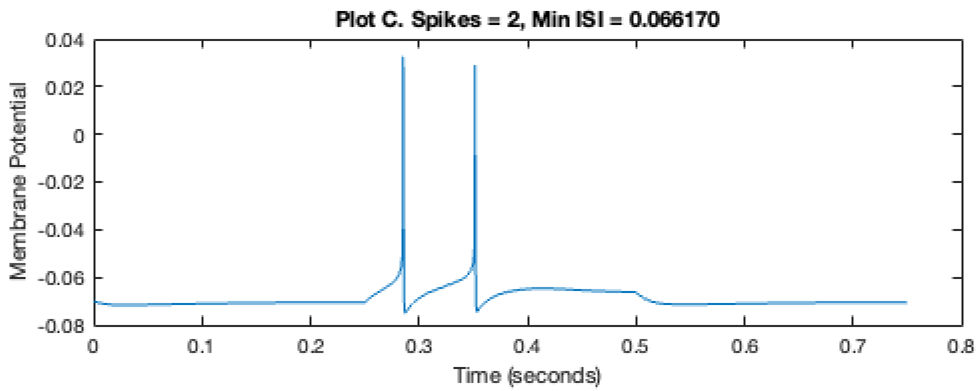

Applied Current vs. Time
×10^-10
-1.35
-1.4
-1.45
-1.5
Applied Current
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Time (seconds)

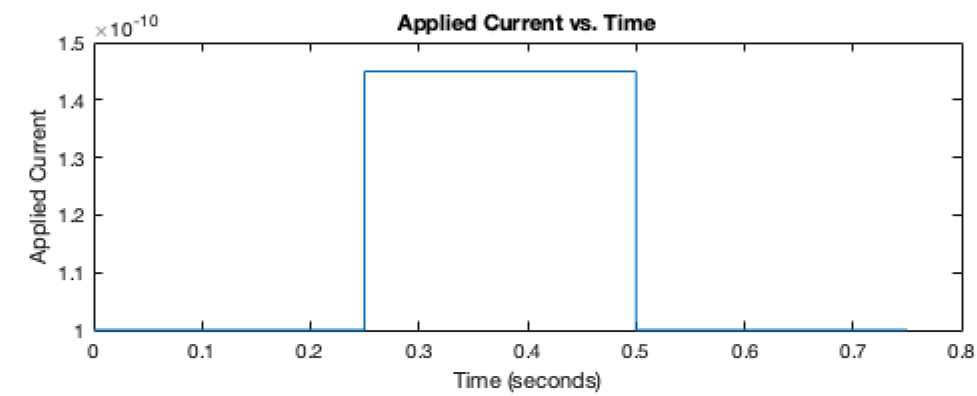

Plot A. Spikes = 0, Min ISI = 0.000000
-0.07
-0.075
-0.08
-0.085
Membrane Potential
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Time (seconds)

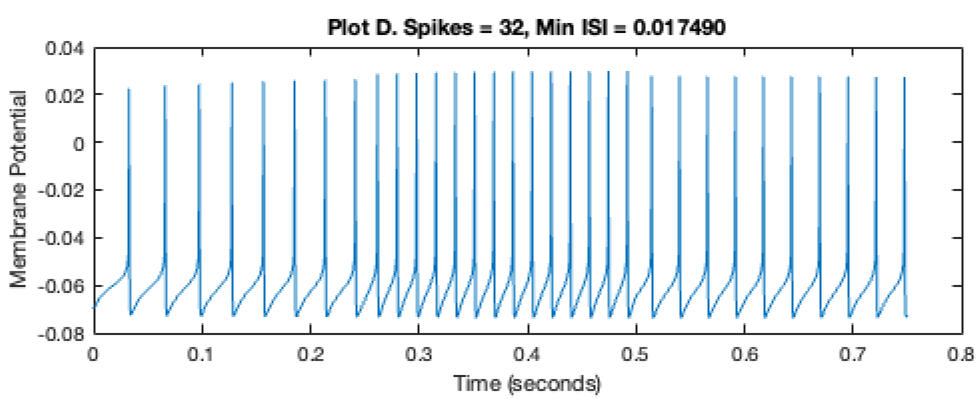

Applied Current vs. Time
×10^-11
2
0
-2
-4
-6
-8
Applied Current
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Time (seconds)

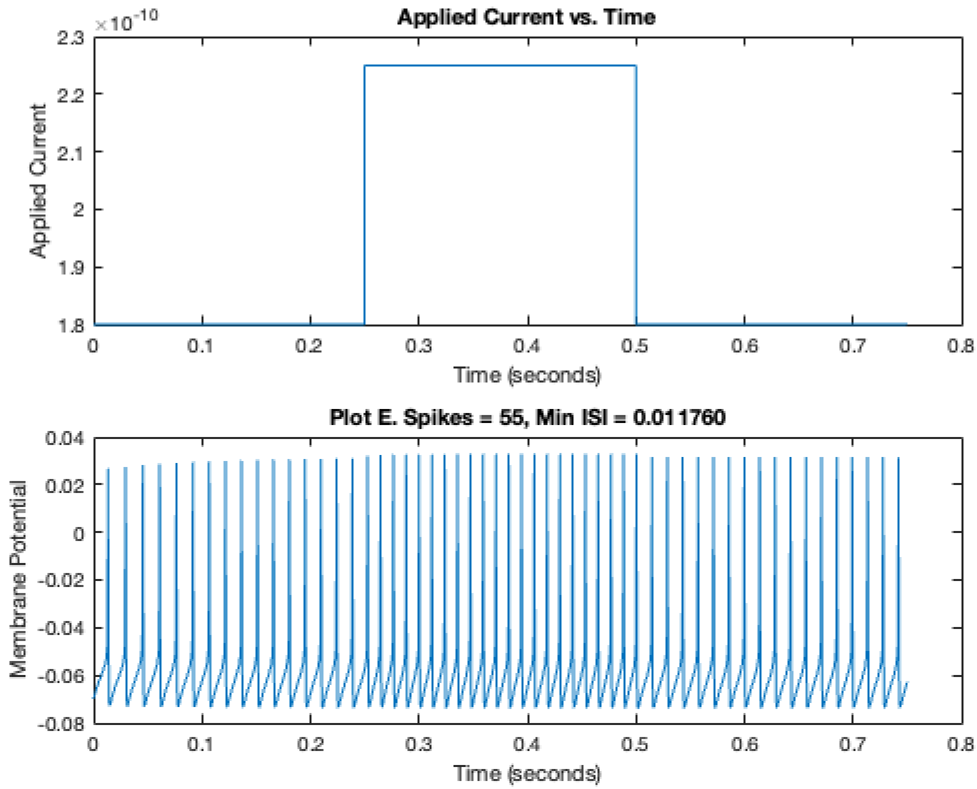

Plot B. Spikes = 11, Min ISI = 0.014430
0.05
0
-0.05
-0.1
Membrane Potential
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Time (seconds)


Applied Current vs. Time
×10^-11
Applied Current
Time (seconds)


Plot C. Spikes = 2, Min ISI = 0.066170
Membrane Potential
Time (seconds)


Applied Current vs. Time
×10^-10
Applied Current
Time (seconds)


Plot D. Spikes = 32, Min ISI = 0.017490
Membrane Potential
Time (seconds)



2. Implement the Connor-Stevens Type-I neuron model and examine its properties.

```
% Parameters for plotting
set(0,'DefaultLineLineWidth',2,...
    'DefaultLineMarkerSize',8, ...
    'DefaultAxesLineWidth',2, ...
    'DefaultAxesFontSize',14,...
    'DefaultAxesFontWeight','Bold');

% Basic cell properties for the model
leakReversalPotential = -0.017;   % leak reversal potential
sodiumReversalPotential = 0.055;  % reversal for sodium channels
potassiumReversalPotential = -0.072; % reversal for potassium channels
AReversalPotential = -0.075;  % reversal for A-type current

leakConductance = 3e-8;     % specific leak conductance
sodiumConductance = 1.2e-5;  % specific sodium conductance
potassiumConductance = 2e-6; % specific potassium conductance
AConductance = 4.77e-6;  % specific A-type potassium conductance

membraneCapacitance = 0.1e-9;    % specific membrane capacitance

timeStep = 0.000005;  % time-step for the simulation
```

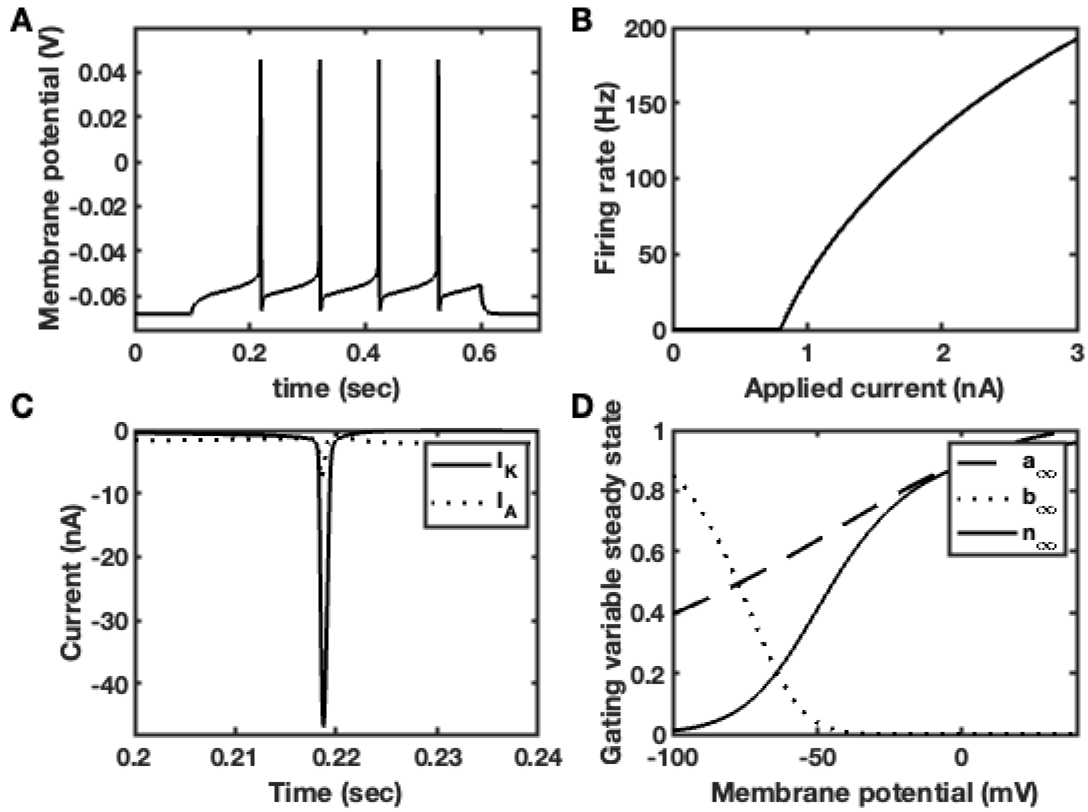
```
spikeOnsetThreshold = 0.0;  % value for spike onset detection
spikeOffsetThreshold = -0.020;  % value for spike offset detection

% Part 1 for current step (A, C in figure), Part 2 for f-I curve
for plotPart = 1:2

    if (plotPart == 1)     % current step
        timeMin = -0.1;    % start time of simulation
        timeMax = 0.7;     % end time of simulation

        appliedCurrentValues = 0.85e-9;  % applied current is fixed

        currentStart = 0.1;   % time applied current starts
        currentDuration = 0.5; % duration of applied current pulse
    else  % f-I curve
        timeMin = 0;  % start time of simulation
        timeMax = 5;  % end time of simulation
        currentStart = timeMin;  % time applied current starts
        currentDuration = timeMax - timeMin;  % duration of applied current
pulse

        timeVector = 0:timeStep:timeMax;  % time vector

        appliedCurrentValues = 0e-9:0.05e-9:3e-9; % set of applied current
values
    end

    % Indices for current onset and offset
    startIndex = floor((currentStart-timeMin)/timeStep)+1;
    stopIndex = floor((currentStart+currentDuration-timeMin)/timeStep)+1;
    baseCurrent = 0e-9;  % base current outside of step

    trialCount = length(appliedCurrentValues);  % Number of points in f-I
curve (1 in part 1)
    firingRate = zeros(1, trialCount);  % initialize for f-I curve

    timeVector = timeMin:timeStep:timeMax;  % time vector

    for trial = 1:trialCount  % loop through trials (only once in part 1)

        appliedCurrent = appliedCurrentValues(trial);  % applied current
value for this trial

        currentVector = baseCurrent*ones(size(timeVector));  % Applied current
        currentVector(startIndex:stopIndex) = appliedCurrent;  % make non-
zero for duration of current pulse

        spikeVector = zeros(size(timeVector));  % vector to store spike times
        inSpike = 0;  % flag to indicate if in a spike

        membranePotential = zeros(size(timeVector));  % voltage vector
        membranePotential(1) = leakReversalPotential;  % set the initial
value of voltage
```

```
        potassiumActivation = zeros(size(timeVector));  % n: potassium
activation gating variable
        potassiumActivation(1) = 0.0;  % start off at zero
        sodiumActivation = zeros(size(timeVector));  % m: sodium activation
gating variable
        sodiumActivation(1) = 0.0;  % start off at zero
        sodiumInactivation = zeros(size(timeVector));  % h: sodium
inactivation gating variable
        sodiumInactivation(1) = 0.0;  % start off at zero

        ACurrentActivation = zeros(size(timeVector));  % A-current activation
gating variable
        ACurrentActivation(1) = 0.0;  % start off at zero
        ACurrentInactivation = zeros(size(timeVector));  % A-current
inactivation gating variable
        ACurrentInactivation(1) = 0.0;  % start off at zero

        totalCurrent = zeros(size(timeVector));  % total current
        sodiumCurrent = zeros(size(timeVector));  % sodium current
        potassiumCurrent = zeros(size(timeVector));  % potassium current
        ACurrent = zeros(size(timeVector));  % A-type current
        leakCurrent = zeros(size(timeVector));  % leak current

        for idx = 2:length(timeVector)  % simulation loop

            membraneVoltage = membranePotential(idx-1);  % convert voltage to
mV

            % find steady state and time constant of all gating variables
            % given the membrane potential
            [sodiumActivationInf, sodiumActivationTau, sodiumInactivationInf,
sodiumInactivationTau, potassiumActivationInf, potassiumActivationTau,
ACurrentActivationInf, ACurrentActivationTau, ...
                ACurrentInactivationInf, ACurrentInactivationTau] =
gating(membraneVoltage);

            sodiumActivation(idx) = sodiumActivation(idx-1) +
(sodiumActivationInf-sodiumActivation(idx-1))*timeStep/sodiumActivationTau;
% Update sodium activation
            sodiumInactivation(idx) = sodiumInactivation(idx-1) +
(sodiumInactivationInf-sodiumInactivation(idx-1))*timeStep/
sodiumInactivationTau;  % Update sodium inactivation
            potassiumActivation(idx) = potassiumActivation(idx-1) +
(potassiumActivationInf-potassiumActivation(idx-1))*timeStep/
potassiumActivationTau;  % Update potassium activation

            ACurrentActivation(idx) = ACurrentActivation(idx-1) +
(ACurrentActivationInf-ACurrentActivation(idx-1))*timeStep/
ACurrentActivationTau;  % Update A-current activation
            ACurrentInactivation(idx) = ACurrentInactivation(idx-1) +
(ACurrentInactivationInf-ACurrentInactivation(idx-1))*timeStep/
ACurrentInactivationTau;  % Update A-current inactivation
```

```
            leakCurrent(idx) = leakConductance*(leakReversalPotential-
membranePotential(idx-1));  % leak current

            sodiumCurrent(idx) =
sodiumConductance*sodiumActivation(idx)^3*sodiumInactivation(idx)*(sodiumRever
salPotential-membranePotential(idx-1));  % sodium current

            potassiumCurrent(idx) =
potassiumConductance*potassiumActivation(idx)^4*(potassiumReversalPotential-
membranePotential(idx-1));  % potassium current

            ACurrent(idx) =
AConductance*ACurrentActivation(idx)^3*ACurrentInactivation(idx)*(AReversalPot
ential-membranePotential(idx-1));  % A-type current

            totalCurrent(idx) = leakCurrent(idx) + sodiumCurrent(idx) +
potassiumCurrent(idx) + ACurrent(idx) + currentVector(idx);  % total current

            membranePotential(idx) = membranePotential(idx-1) +
totalCurrent(idx)*timeStep/membraneCapacitance;  % Update membrane potential

            % Record spike time on the upswing.
            if (inSpike == 0) && (membranePotential(idx) >
spikeOnsetThreshold)
                inSpike = 1;  % Now "in a spike"
                spikeVector(idx) = 1;  % Record spike time
            elseif (inSpike == 1) && (membranePotential(idx) <
spikeOffsetThreshold)
                inSpike = 0;  % Reset for next spike detection
            end

        end

        % Find spike times in seconds
        spikeTimes = timeStep * find(spikeVector);
        if length(spikeTimes) > 1  % if more than 1 spike
            isiValues = diff(spikeTimes);  % find inter-spike intervals
            firingRate(trial) = 1 / isiValues(end);  % final rate is 1/ISI
        end

    end  % end of trial loop

    % Plot results
    if (plotPart == 1)  % single trial with current step
        figure(1)
        clf
        % Plot V vs time for current step
        subplot('Position', [0.12 0.6 0.36 0.36])
        plot(timeVector, membranePotential, 'k');
        xlabel('time (sec)')
        ylabel('Membrane potential (V)')
        axis([0 0.7 -0.075 0.06])

        % Zoom in on potassium currents around a spike
```

```
        subplot('Position', [0.12 0.12 0.36 0.36])
        plot(timeVector, potassiumCurrent*1e9, 'k-');
        hold on
        plot(timeVector, ACurrent*1e9, 'k:')
        axis([0.2 0.24 -48 0])

        xlabel('Time (sec)')
        ylabel('Current (nA)')
        legend('I_{K}', 'I_{A}')
    else  % Many trials with different Iapp
        subplot('Position', [0.6 0.6 0.36 0.36])
        plot(appliedCurrentValues*1e9, firingRate, 'k');
        xlabel('Applied current (nA)')
        ylabel('Firing rate (Hz)')
    end

end  % end of plot loop

% Show V-dependence of I_A gating variables
voltageRange = -0.100:0.001:0.040;  % Range of values for V
[sodiumActivationInf, sodiumActivationTau, sodiumInactivationInf,
sodiumInactivationTau, potassiumActivationInf, potassiumActivationTau,
ACurrentActivationInf, ACurrentActivationTau, ...
    ACurrentInactivationInf, ACurrentInactivationTau] = gating(voltageRange);

% Plot gating variables versus V
subplot('Position', [0.6 0.12 0.36 0.36])
plot(voltageRange*1000, ACurrentActivationInf, 'k--');
xlabel('Membrane potential (mV)')
ylabel('Gating variable steady state')
hold on
plot(voltageRange*1000, ACurrentInactivationInf, 'k:');
plot(voltageRange*1000, potassiumActivationInf, 'k');
axis([-100 40 0 1])
legend('a_{\infty} ', 'b_{\infty} ', 'n_{\infty} ')

annotation('textbox', [0 0.98 0.02 0.02], 'LineStyle', 'none', 'FontSize',
16, 'FontWeight', 'Bold', 'String', 'A')
annotation('textbox', [0.5 0.98 0.02 0.02], 'LineStyle', 'none', 'FontSize',
16, 'FontWeight', 'Bold', 'String', 'B')
annotation('textbox', [0.0 0.52 0.02 0.02], 'LineStyle', 'none', 'FontSize',
16, 'FontWeight', 'Bold', 'String', 'C')
annotation('textbox', [0.5 0.52 0.02 0.02], 'LineStyle', 'none', 'FontSize',
16, 'FontWeight', 'Bold', 'String', 'D')

function[sodiumActivationInf, sodiumActivationTau, sodiumInactivationInf,
sodiumInactivationTau, potassiumActivationInf, potassiumActivationTau,
ACurrentActivationInf, ACurrentActivationTau, ...
    ACurrentInactivationInf, ACurrentInactivationTau] = gating(voltage)

% Returns the steady states and time constants of the gating variables for
the Connor-Stevens model.
% Input: membrane potential (voltage) in Volts.
```

```
% Sodium and potassium gating variables are defined by the voltage-dependent
transition rates between states (alpha and beta).
alpha_sodiumActivation = 3.80e5 * (voltage + 0.0297) ./ (1 - exp(-100 *
(voltage + 0.0297)));
beta_sodiumActivation = 1.52e4 * exp(-55.6 * (voltage + 0.0547));

alpha_sodiumInactivation = 266 * exp(-50 * (voltage + 0.048));
beta_sodiumInactivation = 3800 ./ (1 + exp(-100 * (voltage + 0.018)));

alpha_potassiumActivation = 2e4 * (voltage + 0.0457) ./ (1 - exp(-100 *
(voltage + 0.0457)));
beta_potassiumActivation = 250 * exp(-12.5 * (voltage + 0.0557));

% Steady state values and time constants for m, h, and n gating variables.
sodiumActivationTau = 1 ./ (alpha_sodiumActivation + beta_sodiumActivation);
% time constant (s)
sodiumActivationInf = alpha_sodiumActivation ./ (alpha_sodiumActivation +
beta_sodiumActivation);

sodiumInactivationTau = 1 ./ (alpha_sodiumInactivation +
beta_sodiumInactivation);  % time constant (s)
sodiumInactivationInf = alpha_sodiumInactivation ./ (alpha_sodiumInactivation
+ beta_sodiumInactivation);

potassiumActivationTau = 1 ./ (alpha_potassiumActivation +
beta_potassiumActivation);  % time constant (s)
potassiumActivationInf = alpha_potassiumActivation ./
(alpha_potassiumActivation + beta_potassiumActivation);

% A-type current gating variables: steady-state values and time constants
found empirically.
ACurrentActivationInf = (0.0761 * exp(31.4 * (voltage + 0.09422)) ./ (1 +
exp(34.6 * (voltage + 0.00117)))).^(1/3);
ACurrentActivationTau = 0.3632e-3 + 1.158e-3 ./ (1 + exp(49.7 * (voltage +
0.05596)));

ACurrentInactivationInf = (1 ./ (1 + exp(68.8 * (voltage + 0.0533)))).^4;
ACurrentInactivationTau = 1.24e-3 + 2.678e-3 ./ (1 + exp(62.4 * (voltage +
0.050)));

end
```



Published with MATLAB® R2024a