Table of Contents


```
clear
close all
clc

% Project Authors: Sarah Fatkin, Emily Gu, Rehan Chinoy

% Introduction:
% The inward rectifying potassium channel (Kir) is an essential part
% of the muscle fiber membrane.
% By producing a strong inward potassium current after a cell spikes,
% Kir channels stabilize resting
% membrane potential and act as shunts. We are using the Morris-Lecar
% model to examine how Kir channels
% perturb cell spiking and bifurcation behavior.

% Conclusion:
% We were able to successfully implement the Kir channel into the
% Morris-Lecar model using experimental
% parameters. Kir channel addition produced no difference in spiking
% frequency and a slight shift of the
% bifurcation diagram. Our results were more subtle than initially
% anticipated, but they make sense in
% light of the differences in magnitude between external input and Kir.

%%%%%%
% Please note that this is the code for our main analysis. The bifurcation
% diagram code can be found on our Github in the script ad_ex2_bifurcation.m,
 linked here:

% https://github.com/rehanbchinoy/morris-lecar
```

code font settings

```
%%%% Set "Arial" as the Default font
set(0,'defaultAxesFontSize',16);
set(0,'defaultAxesFontName','Arial');
set(0,'defaultTextFontSize',16);
```

```
set(0,'defaultTextFontName','Arial');

set(0,'defaultUipanelFontName','Arial');
set(0,'defaultUicontrolFontName','Arial');
```

Simulation parameters

```
Nt     = 50000;  % Num. of sample
dt     = 0.01;   % time step for numerical integration; unit : msec
time   = linspace(0, Nt-1, Nt) * dt; % time vector; unit : msec
```

typical parameter setting for Type I mode

```
C    =  5; %5;  % (1e-10 Farad)
gL   =  2; % (1e-9)
gK   =  8; %
gCa  =  4;
VL   = -60; % mV
VK   = -86.9; % -80 was default
VCa  =  120;
V1   = -1.2;
V2   =  18;
V3   =  12;
V4   =  17.4;
Iext =  40; % 39.8; (1e-12)
phi  =  1/15;
```

typical parameter setting for Type II mode

C = 5; gL = 2; gK = 8; gCa = 4.4; VL = -60; VK = -80; VCa = 120; V1 = -1.2; V2 = 18; V3 = 2; V4 = 30; Iext = 100.5; phi = 1/25; %unit: 1/msec

```
% KIR ADDITION
with_kir = true;


X0     = [0, 0]; % initial value of state variables
                 % X0(1): membrane potential, v
                 % X0(2): recovery variable,  w
%%%%% parameter settings
```

Solve differential equation - with KIR

```
X      = zeros(Nt, length(X0));
X(1,:) = X0;
kir(1) = 0;
for i = 2:Nt
    X_now  = X(i-1,:);
    %%%%% Numerical integral scheme with 4th order Runge Kutta method
    [X(i,:), kir(i)] = runge_kutta(X_now, dt, @MorrisLecar, ...
```

```
                                          C, gL, gK, gCa,...
                                             VL, VK, VCa,...
                                             V1, V2, V3, V4,...
                                             Iext, phi, with_kir);
end
```

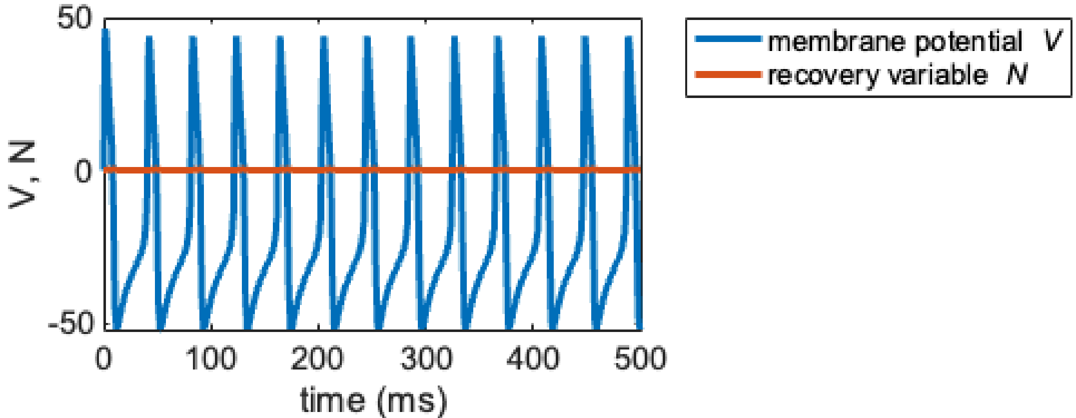
plot

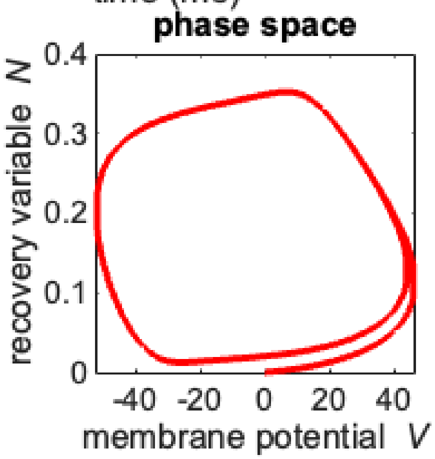
```
fig = figure(1);
% figure_setting(60, 40, fig);
sfh1 = subplot(2,1,1,'parent', fig);
plot(time, X(:,1), 'LineWidth', 3);
hold on
plot(time, X(:,2), 'LineWidth', 3);
hold off

xlabel('time (ms)')
ylabel('V, N')
lgnd = legend({'membrane potential \it V', 'recovery variable \it
 N'}, 'location', 'northeastoutside');

%%%%%%%
sfh2 = subplot(2,1,2,'parent', fig);
plot(X(:,1), X(:,2), 'r', 'LineWidth', 3);
xlabel('membrane potential \it V')
ylabel('recovery variable \it N')
title('phase space')
axis square
sfh2.Position = sfh2.Position - [0.1, 0, 0, 0];

%%%%%
% sfh2 = subplot(2,2,3,'parent', fig);
% plot(time, kir, 'LineWidth', 3);
% fname = [filepath, filesep, 'figures', filesep, 'ex1', filesep, 'result'];
% figure_save(fig, fname)
```





compare with and without KIR

solve - without KIR X2 = zeros(Nt, length(X0)); X2(1,:) = X0; kir(1) = 0; with_kir = false; for i = 2:Nt X_now = X(i-1,:); %%%%% Numerical integral scheme with 4th order Runge Kutta method [X2(i,:),kir2(i)] = runge_kutta(X_now, dt, @MorrisLecar, ... C, gL, gK, gCa,... VL, VK, VCa,... V1, V2, V3, V4,... Iext, phi, with_kir); end % % plot with and without KIR figure subplot(2,2,1) plot(time, X(:,1), 'LineWidth', 3); ylabel('membrane potential w/ Kir') subplot(2,2,2) plot(time, kir, 'LineWidth', 3); ylabel('Kir') subplot(2,2,3) plot(time, X2(:,1), 'LineWidth', 3); ylabel('membrane potential w/o Kir') subplot(2,2,4) plot(time, kir2, 'LineWidth', 3);

functions

```
function [dXdt, I_kir] = MorrisLecar(X, varargin)
    V    = X(1);
    N    = X(2);

    if length(varargin)==1
        par  = varargin{1};
    else
        par  = varargin;
    end

    C    = par{1};
```

```
    gL   = par{2};
    gK   = par{3};
    gCa  = par{4};
    VL   = par{5};
    VK   = par{6};
    VCa  = par{7};
    V1   = par{8};
    V2   = par{9};
    V3   = par{10};
    V4   = par{11};
    Iext = par{12};
    phi  = par{13};
    with_kir = par{14}; % true or false

    Minf = Sigm(V, V1, V2);
    Ninf = Sigm(V, V3, V4);

    % KIR calculation
    if with_kir
        f_kir = 0.12979 * (V - VK)/(1+exp(0.093633 * (V+72))); % experimental
 parameters from Van Putten, 2015
        P_3 = Sigmoid(V); % parameters for Sigmoid are in function below
        I_kir = 10 * C * P_3 * f_kir;
    else
        I_kir = 0;
    end

    % Model equations
    dVdt = 1/C * (- gL   * (V - VL) ...
                  - gCa * Minf * (V - VCa) ...
                  - gK   * N   * (V - VK) + Iext + I_kir);
    dNdt =  Lambda(V, V3, V4, phi) * (Ninf - N);

    dXdt = [dVdt, dNdt];

end

function val = Sigm(V, V1, V2)
    %%%% Sigmoid function for Minf and Ninf
    val =  1 / (1 + exp(-2 * (V - V1)/V2));
    % This function can be also expressed as: val = 0.5 * (1 + tanh((V - V1)/
V2));
end

function lambda = Lambda(V, V1, V2, phi)
    lambda = phi * cosh((V-V1)/(2*V2));
end

function [X_next,Ikir] = runge_kutta(X_now, dt, func, varargin)
    [k1,Ikir]     = func(X_now, varargin);

    X_k2   = X_now + (dt/2) * k1;
    k2     = func(X_k2, varargin);
```

```
    X_k3   = X_now + (dt/2) * k2;
    k3     = func(X_k3, varargin);

    X_k4   = X_now + dt * k3;
    k4     = func(X_k4, varargin);

    X_next = X_now + (dt/6)*(k1 + 2*k2 + 2*k3 + k4);
end

function S = Sigmoid(V)
    % Sigmoid function for P3
 b1 = -110;
    b2_first = 20;
    b2_second = 10;

    if V < -110
        b2 = b2_first;
    else
        b2 = b2_second;
    end

 S = 1 ./ (1 + exp((b1 - V)/b2));
end
```

Published with MATLAB® R2022a