Morris-Lecar Function

```
function dy = Morris_Lecar(t,y)
%Iapp
Iapp = 100; % in nA
%output Variables (Anthony)
Vhopf=y(1);
nhopf=y(2);
VSNLC=y(3);
nSNLC=y(4);
%define Global Variables (Anthony)
Cm = 20 ; %microfarad/cm^2
ECa=120; %millivolts
gK=8;% millisiemens/ cm^2
EK=-84; %millivolts
gL=2;% millisiemens/ cm^2
EL=-60;%millivolts
v1=-1.2; %millivolts
v2= 18 ; %millivolts
%____________________Hopf____________________
%Additional Parameters for Hopf Bifurcation Class 2 Neurons (Anthony)
v3_hopf= 2 ; %millivolts
v4_hopf= 30; %millivoltshb
phi_hopf=0.04; % per millisecond
gCa_hopf=4.4; % millisiemens/ cm^2
%Other supporting functions in functions of voltage.
m_inf_hopf = (0.5*(1+tanh((Vhopf-v1)/v2))); %(Anthony)
n_inf_hopf = (0.5*(1+tanh((Vhopf-v3_hopf)/v4_hopf))); %(Anthony)
tau_n_hopf = (1/(cosh((Vhopf-v3_hopf)/(2*v4_hopf)))); % (Eric)
%differential equations for hopf
% dn/dt (Anthony)
dnhopf_over_dt = phi_hopf*(n_inf_hopf - nhopf) / tau_n_hopf;
% dV/dt (Eric)
P1 = gL*(Vhopf - EL); %leak component
P2 = gK * nhopf * (Vhopf - EK); % pottasium component
P3 = gCa_hopf * m_inf_hopf * (Vhopf-ECa); %Calcium component
dVhopf_over_dt = (Iapp - P1 - P2 - P3) / Cm; %combine
%___________________________________________________________
%_______________________SNLC functions_______________________
%additional Parameters for (SNLC) Bifurcation Class 1 Neurons (Eric)
v3_SNLC = 12; % in mV
v4_SNLC = 17.4; % in mV
phi_SNLC = 0.067; % in ms^-1
gCa_SNLC=4.4; % millisiemens/ cm^2
%Other supporting functions in functions of voltage.
m_inf_SNLC = (0.5*(1+tanh((VSNLC-v1)/v2))); %(Anthony)
n_inf_SNLC = (0.5*(1+tanh((VSNLC-v3_SNLC)/v4_SNLC))); %(Anthony)
tau_n_SNLC = (1/(cosh((VSNLC-v3_SNLC)/(2*v4_SNLC)))); % (Eric)
%differential equations for hopf
% dn/dt(Anthony)
dnSNLC_over_dt = phi_SNLC*(n_inf_SNLC - nSNLC) / tau_n_SNLC;
% dV/dt(Eric)
```

```
P1_SNLC = gL*(VSNLC - EL); %leak component
P2_SNLC = gK * nSNLC * (VSNLC - EK); % pottasium component
P3_SNLC = gCa_SNLC * m_inf_SNLC * (VSNLC-ECa); %Calcium component
dVSNLC_over_dt = (Iapp - P1_SNLC - P2_SNLC - P3_SNLC) / Cm; %combine
%_____________________________________________________________
%Wrapper function (Eric)
dy = [dVhopf_over_dt; dnhopf_over_dt; dVSNLC_over_dt;dnSNLC_over_dt];
end
```

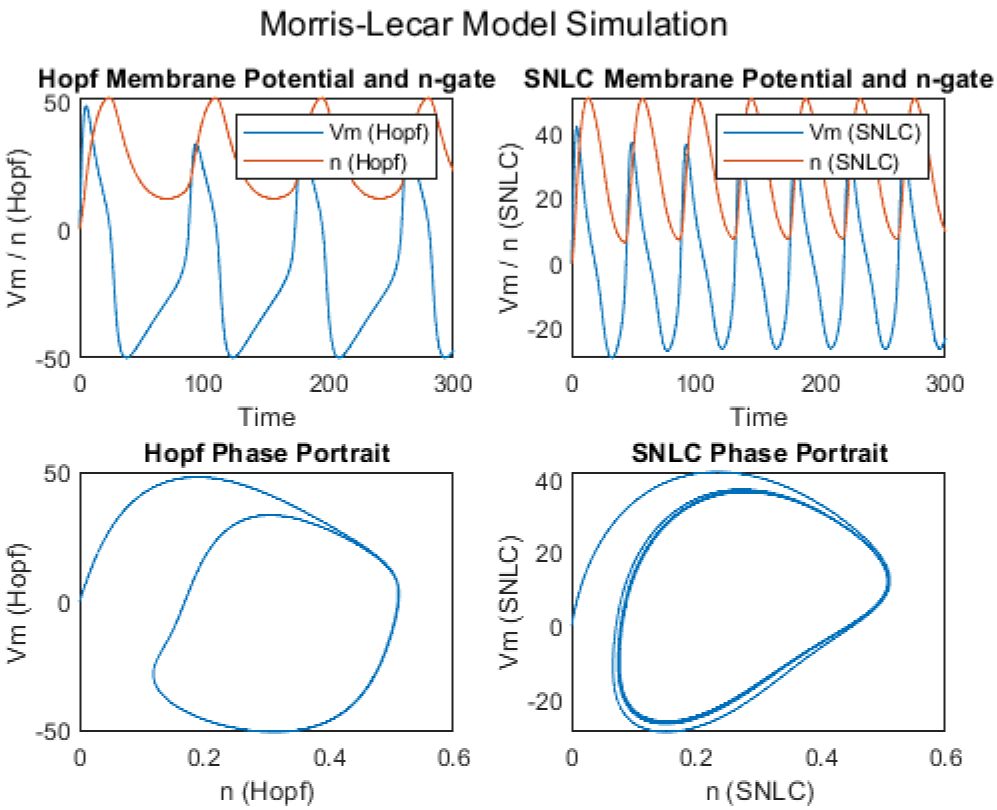
```
Not enough input arguments.

Error in Morris_Lecar (line 7)
Vhopf=y(1);
```

Don't worry about this, this is just the .m file that defines functions to be called in another .m file.

Published with MATLAB® R2022b

Lab 7: Morris-Lecar Simulation

```
%Initial Values
V0 = 0;
n0 = 0;
y0 = [V0;n0;V0;n0];
tspan = [0:2*10^-3:300]; %time span
[t, y] = ode45(@(t,y)Morris_Lecar(t,y),tspan, y0);
%Output Variables
Vm_hopf = y(:,1);
n_hopf = y(:,2);
Vm_SNLC = y(:,3);
n_SNLC = y(:,4);

figure;

subplot(2, 2, 1);
plot(t, Vm_hopf)
hold on
plot(t, n_hopf*100) % Upscale n by 100 for visualization
hold off
xlabel('Time')
ylabel('Vm / n (Hopf)')
title('Hopf Membrane Potential and n-gate')
legend('Vm (Hopf)', 'n (Hopf)')

subplot(2, 2, 2);
plot(t, Vm_SNLC)
hold on
plot(t, n_SNLC*100) % Upscale n by 100 for visualization
hold off
xlabel('Time')
ylabel('Vm / n (SNLC)')
title('SNLC Membrane Potential and n-gate')
legend('Vm (SNLC)', 'n (SNLC)')

subplot(2, 2, 3);
plot(n_hopf, Vm_hopf)
xlabel('n (Hopf)')
ylabel('Vm (Hopf)')
title('Hopf Phase Portrait')

subplot(2, 2, 4);
plot(n_SNLC, Vm_SNLC)
xlabel('n (SNLC)')
ylabel('Vm (SNLC)')
title('SNLC Phase Portrait')

sgtitle('Morris-Lecar Model Simulation')
```



Explanations (Mauricio)

```
% Keep in mind the Morris-Lecar neuron model has the capacity to exhibit
% Hopf Birfurcation and also Saddle Node bifurcation, it just depends on
% the parameters and initial conditions passed into the function.

% As you can see in the Hopf diagrams, we see that we have oscillatory
% behavior, where if you start outside the oscillarory attractor you will
% be attracted to it as it is a stable limit cycle.

% Similarly, you can see in the SNLC Phase portrait that the trajectory is
% also drawn to a stable limit cycle attrractor.

% You can also observe that when the membrane is near the threshold
% potential, the activation and inactivation dynamics of the ion channels
% can lead to sustained oscillations

% This model also exhibits bistability where depending on the state of the
% system, it can can either remain at a low voltage resting state, or to a
% high voltage active state. This is a result of input current creating
% sustained firing.

% This model is also highly sensitive to input current.
```

Published with MATLAB® R2022b