Problem Set 3
EPS 528, Science of Complex Systems

Jonas Katona

November 16, 2022

Problem 1.

Solution. The ODE system we desire to analyze is the $I_{Na,p} + I_K$ model:

$$\begin{aligned} C\dot{V} &= I - g_L\left(V - E_L\right) - g_{Na}m_\infty(V)\left(V - E_{Na}\right) - g_K n\left(V - E_K\right) := Cf_V\left(V, n\right), \\ \dot{n} &= \frac{n_\infty(V) - n}{\tau(V)} := f_n\left(V, n\right), \end{aligned} \tag{1}$$

where

$$m_\infty(V) = \frac{1}{1 + \exp\left(\frac{V_{1/2,m} - V}{k_m}\right)}, \quad n_\infty(V) = \frac{1}{1 + \exp\left(\frac{V_{1/2,n} - V}{k_n}\right)},$$
$$\tau\left(V\right) = C_{\text{base}} + C_{\text{amp}} \exp\left(\frac{-\left(V_{\max} - V\right)^2}{\sigma^2}\right).$$

Of course, we can choose constants C_{base}, C_{amp}, $V_{\max}$, and σ such that $\tau\left(V\right) = \text{const.}$, which we will do for our analysis. The V- and n-nullclines can be found by setting both equations in (1) to zero separately, which (hopefully) will give us two equations for each of the nullcline curves. This gives us the following two equations:

$$\begin{aligned} &I - g_L\left(V - E_L\right) - g_{Na}m_\infty(V)\left(V - E_{Na}\right) - g_K n\left(V - E_K\right) = 0 \\ &\Rightarrow \boxed{n = \frac{I - g_L\left(V - E_L\right) - g_{Na}m_\infty(V)\left(V - E_{Na}\right)}{g_K\left(V - E_K\right)}}, \end{aligned} \tag{2}$$

$$\frac{n_\infty(V) - n}{\tau(V)} = 0 \Rightarrow n_\infty(V) - n = 0 \Rightarrow \boxed{n = n_\infty(V)}. \tag{3}$$

(It turns out that we will only use (2) and (3) when plotting the nullclines; they are not of much use for finding the fixed points.)

Note that when (2) and (3) intersect, then $\dot{V} = \dot{n} = 0$, in which case we have a fixed point. Unfortunately, these fixed points cannot be expressed analytically, which means that we must find these numerically, e.g., using Newton's method, the secant method, bisection method, etc. However, since I wrote my code in MATLAB, we could make use of MATLAB's (presumably) more robust built-in root finder, `fsolve`, to find the zeros of (1) directly, which is what I do. Of course, most root-finding algorithms only work for starting values close enough to a fixed point, so to find all fixed points, we populate our domain with equally-spaced starting values and run `fsolve` on each starting value. If the algorithm converges,

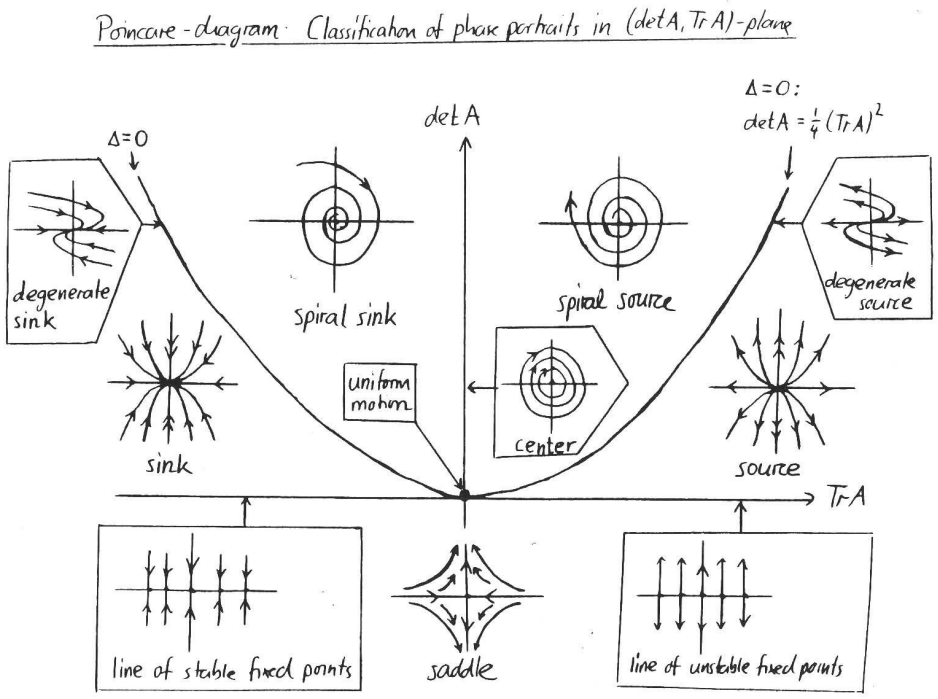

Figure 1: A trace-determinant diagram showing the fixed point classification I used for a given combination of $\det J$ and $\mathrm{tr}J$. This diagram was taken from ttps://tex.stackexchange.com/questions/347201/drawing-the-trace-determinant-diagram-on-latex/347401.

we save the fixed point it converged too, and at the end, we remove duplicate values and display where the final fixed points are.

From here, once we find these fixed points numerically, we need to plug these into the Jacobian of (1). We could, in theory, calculate the Jacobian explicitly as a function of the parameters and the location (V, n) in the phase plane, since derivatives can be calculated explicitly for all of the functions used in (1). However, there is no need when we can calculate the Jacobian numerically as well. To do so, we use second-order centered differences:

$$\begin{aligned} J(V,n) &= \begin{pmatrix} \partial_V f_V(V,n) & \partial_n f_V(V,n) \\ \partial_V f_n(V,n) & \partial_n f_n(V,n) \end{pmatrix} \\ &\approx \frac{1}{2h} \begin{pmatrix} f_V(V+h,n) - f_V(V-h,n) & f_V(V,n+h) - f_V(V,n-h) \\ f_n(V+h,n) - f_n(V-h,n) & f_n(V+h,n) - f_n(V-h,n) \end{pmatrix}, \end{aligned} \tag{4}$$

where $h \ll 1$. In fact, for simplicity, we use the same value $\texttt{tol} \ll 1$ for h, the function and optimality tolerances in `fsolve`, and the relative error tolerance in `ode45`, which we use to plot the phase space trajectories. In particular, for the simulations in this paper, we let $\texttt{tol} = 10^{-10}$.

Finally, the code I wrote also classifies all fixed points using the determinant and trace of (4). Since our Jacobian is an approximation to the true Jacobian and the fixed points we plugged in are also approximate, we only classify these fixed points with `sqrt(tol)` tolerance. In other words, if our fixed point is within `sqrt(tol)` of some degenerate state, i.e., a degenerate sink or source, a line of fixed points, a center, or uniform motion, then we will label the fixed point as possibly being one of these things, but otherwise, we will take the numerical values as they are; see Figure 1 for more details.

At the very end, my code returns the classification of each fixed point, where these classifications are ordered in the same way as the locations of each fixed point were displayed earlier. However, before ending, my code returns a phase portrait with the following:

1. V-nullclines and n-nullclines, as given in (2) and (3). `Vq` gives the domain grid values (i.e., in V) at which to evaluate (2) and (3) and generate two curves.

2. Seven sample trajectories. The vectors of starting coordinates at $t = t_i$ for V and n must be inputted as `V0` and `n0`, respectively. The system is integrated up to time $t = t_f$, such that `tspan`$=(t_i, t_f)$. This numerical integration is done via `ode45` with relative error tolerance `tol`. To add arrows in my plot which show the direction of trajectories with time, I used a utility function called `arrowh` which I found in the MathWorks File Exchange website: https://www.mathworks.com/matlabcentral/fileexchange/4538-arrowhead.

3. Fixed points, along with their stabilities. Stable fixed points are blue, unstable fixed points are red, and all other fixed points with (possibly) degenerate stabilities are green, e.g., centers and points which are stable or unstable in a nonlinear sense.

Anyways, here is my code, `simulation`. Test it out if you would like, but do not forget that your phase space trajectories might look strange if you use weirdly-spaced initial conditions. Also, my code can handle *up to* seven trajectories (yes to less than seven, no to more than seven).[1]

```
function [ fpclassify ] = simulation(tol, V0, n0, tspan, Vq,...
    I, C, gL, EL, gNa, ENa, Vm, km, gK, EK, Vn, kn, tauV)

lesstol = sqrt(tol);
minf = @(V) 1 ./ (1 + exp((Vm - V) / km));
ninf = @(V) 1 ./ (1 + exp((Vn - V) / kn));
Vfunc = @(V, n) (I - gL * (V - EL) - gNa *...
    minf(V) .* (V - ENa) - gK * n .* (V - EK)) / C;
nfunc = @(V, n) (ninf(V) - n) / tauV;
fun = @(vec) [Vfunc(vec(1), vec(2)); nfunc(vec(1), vec(2))];
Vnullcline = @(V) (I - gL * (V - EL) - gNa *...
    minf(V) .* (V - ENa)) ./ (gK * (V - EK));

% find fixed points by brute force
Vinit = -100 : 1 : 100;
ninit = 0 : 0.1 : 1;
fps = NaN(2, length(Vinit) * length(ninit));
options = optimoptions('fsolve', 'FunctionTolerance', tol,...
    'OptimalityTolerance', tol, 'MaxFunctionEvaluations',...
    1e3, 'Display', 'off');
for i = 1 : length(Vinit)
    for j = 1 : length(ninit)
        % solve for steady states (fixed points)
        trial = fsolve(fun, [Vinit(i) ninit(j)], options);
        % check if actually converged to fixed point
        if norm(fun(trial)) < lesstol
```

[1] I did this because I liked MATLAB's default colors for plots.

```
            fps(:, i + length(Vinit) * (j - 1)) = trial;
        end
    end
end
% remove redundant fixed point values
fps = unique(round(fps, round(-log10(lesstol))).', 'rows').';
fps = fps(:, ~(isnan(fps(1, :)) | isnan(fps(2, :))));
disp(fps) % display fixed points located

Jacobian = zeros(2, 2);
% first row is determinant
% second row is trace
fpdata = zeros(2, length(fps(1, :)));
% returns type of fixed point
fpclassify = struct([]);
for i = 1 : length(fps(1, :))
    Vi = fps(1, i);
    ni = fps(2, i);
    % second-order central difference approximation
    % to the first-derivatives
    Jacobian(:, 1) = (fun([Vi + tol ni])...
        - fun([Vi - tol ni])) / (2 * tol);
    Jacobian(:, 2) = (fun([Vi ni + tol])...
        - fun([Vi ni - tol])) / (2 * tol);
    fpdata(1, i) = det(Jacobian);
    fpdata(2, i) = trace(Jacobian);
    % classify fixed point via Jacobian and combinations
    % of determinant and trace values
    if abs(fpdata(1, i)) < lesstol
        if fpdata(2, i) > lesstol
            fpclassify{i} = 'near unstable line';
        elseif fpdata(2, i) < -lesstol
            fpclassify{i} = 'near stable line';
        else
            fpclassify{i} = 'near uniform motion';
        end
    elseif (abs(fpdata(2, i)) < lesstol) &&...
            (fpdata(1, i) > lesstol)
        fpclassify{i} = 'near center';
    else
        if fpdata(1, i) < -lesstol
            fpclassify{i} = 'saddle';
        else
            if abs(fpdata(1, i) - 0.25 *...
                    (fpdata(2, i)) ^ 2) < lesstol
                if fpdata(2, i) > lesstol
                    fpclassify{i} = 'near degenerate source';
                else
                    fpclassify{i} = 'near degenerate sink';
```

```
                end
            else
                if fpdata(2, i) > lesstol
                    if fpdata(1, i) < 0.25 * (fpdata(2, i)) ^ 2
                        fpclassify{i} = 'source';
                    else
                        fpclassify{i} = 'spiral source';
                    end
                else
                    if fpdata(1, i) < 0.25 * (fpdata(2, i)) ^ 2
                        fpclassify{i} = 'sink';
                    else
                        fpclassify{i} = 'spiral sink';
                    end
                end
            end
        end
    end
end

% identify stable, unstable, and other points
stable = (fpdata(1, :) >= 0) & (fpdata(2, :) < -lesstol);
degenerate = (fpdata(1, :) >= 0) & (abs(fpdata(2, :)) < lesstol);
unstable = setdiff(1 : length(fps(1, :)), stable | degenerate);

figure
% plot nullclines
h = zeros(1, 2);
h(1) = plot(Vq, Vnullcline(Vq), 'c--',...
    'DisplayName', 'V-nullcline');
hold on
grid on
h(2) = plot(Vq, ninf(Vq), 'm--',...
    'DisplayName', 'n-nullcline');
title(['Phase portrait for neuron model, I=', num2str(I)],...
    'fontsize', 16)
xlabel('$V(t)$', 'interpreter', 'latex',...
    'fontsize', 16)
ylabel('$n(t)$', 'interpreter', 'latex',...
    'fontsize', 16)
xlim([Vq(1) Vq(end)])
ylim([0 1])
% plot trajectories
opts = odeset('RelTol', tol); % set error tolerance
func = @(t, vec) fun(vec);
colors = {[0 0.4470 0.7410], [0.8500 0.3250 0.0980],...
    [0.9290 0.6940 0.1250], [0.4940 0.1840 0.5560], ...
    [0.4660 0.6740 0.1880], [0.3010 0.7450 0.9330],...
    [0.6350 0.0780 0.1840]};
```

```
for i = 1 : length(V0)
    u0 = [V0(i) n0(i)]; % set initial values for x_{1} and x_{2}
    sol = ode45(func, tspan, u0, opts); % use ode45 to solve ODE
    u = sol.y;
    plot(u(1, :), u(2, :), 'Color', colors{i})
    arrowh(u(1, :), u(2, :), colors{i}, [],...
        [0.5 : 0.5 : 2.5 3 : 9] .^ 2);
end
% plot fixed points
% stable is blue, unstable is red, and all others are green
scatter(fps(1, stable), fps(2, stable), 'filled', 'b')
scatter(fps(1, degenerate), fps(2, degenerate), 'filled', 'g')
scatter(fps(1, unstable), fps(2, unstable), 'filled', 'r')
hold off
legend(h) % add legend only for nullclines
```

□

Problem 2.

Solution. Using initial conditions $(V(0), n(0)) = (-50, 0.1)$, $(50, 0.1)$, $(-40, 0.6)$, $(50, 0.5)$, $(50, 0.9)$, $(-30, 0.8)$, and $(-50, 0.35)$ for the phase space trajectories, we generate Figures 2 and 3 by varying I in integers from 43 to 51. The color corresponding to each of the seven trajectories can be recovered easily by tracing each trajectory back to its starting point. Furthermore, we only integrate these trajectories from $t = 0$ to $t = 50$. We do this using `simulation` from Problem 1 and the following script:

```
% initial conditions
tol = 1e-10;
t0 = 0; % initial time
tf = 50; % final time
tspan = [t0 tf];

Iq = 43 : 51;
V0 = [-50 50 -40 50 50 -30 -50];
n0 = [0.1 0.1 0.6 0.5 0.9 0.8 0.35];
Vq = -75 : 0.5 : 50; % domain values for nullclines

C = 1;
gL = 1;
EL = -78;
gNa = 4;
ENa = 60;
Vm = -30;
km = 7;
gK = 4;
EK = -90;
Vn = -45;
kn = 5;
tauV = 1;
```

```
for I = Iq
    simulation(tol, V0, n0, tspan, Vq,...
    I, C, gL, EL, gNa, ENa, Vm, km, gK, EK, Vn, kn, tauV)
end
```

As I increases from 48 to 49, the fixed point at the center of the limit cycle changes from stable to unstable, which signals a subcritical Hopf bifurcation as the inner limit cycle (unstable) vanishes and we are left with a spiral source, from which trajectories spiral into the outer limit cycle (stable) which exists for all of the values of I we tested in this problem. The unsteady limit cycle appears to lie between the light blue and gold trajectories when $I = 43$, since the gold trajectory (which starts closer to the fixed point) spirals into the fixed point, while the light blue trajectory (which starts further from the fixed point) enters the outer stable limit cycle. In particular, since there are no fixed points in this annular region, the Poincaré-Bendixson theorem tells us that an unstable limit cycle must exist here. Then, when $I = 44, \ldots, 47$, the unstable limit cycle must lie between the gold and maroon trajectories for the same reasons. Finally, when $I = 48$, even the maroon trajectory spirals outwards away from the fixed point,[2] which indicates that the unstable limit cycle lies *inside* the maroon trajectory. Hence, as we increase I, the area outlined by the unstable limit cycle gets smaller and smaller until I reaches some critical bifurcation value $I^* \in (48, 49)$, at which the unstable limit cycle vanishes entirely and a subcritical Hopf bifurcation occurs.

□

Problem 3.

Solution. Similar to Problem 2, using initial conditions $(V(0), n(0)) = (20, 0)$, $(20, 0.5)$, $(20, 0.9)$, $(-10, 0)$, $(-30, 0)$, $(-40, 1)$, and $(-60, 1)$ for the phase space trajectories, we generate Figures 4 and 5 by varying I in intervals of length $1/2$ from 3 to 6. Furthermore, we integrate these trajectories from $t = 0$ to $t = 50$, and do this all using `simulation` from Problem 1 and the following script:

```
% initial conditions
tol = 1e-10;
t0 = 0; % initial time
tf = 50; % final time
tspan = [t0 tf];

Iq = 3 : 0.5 : 6;
V0 = [20 * ones(1, 3) -10 -30 -40 -60];
n0 = [0 0.5 0.9 0 0 1 1] ;
Vq = -75 : 0.5 : 20; % domain values for nullclines

C = 1;
gL = 8;
EL = -80;
gNa = 20;
ENa = 60;
Vm = -20;
km = 15;
```

[2] Although, the maroon trajectory spirals outwards very slowly. This suggests that the starting point for the maroon trajectory, $(-50, 0.35)$, is very close to lying on the unstable limit cycle.

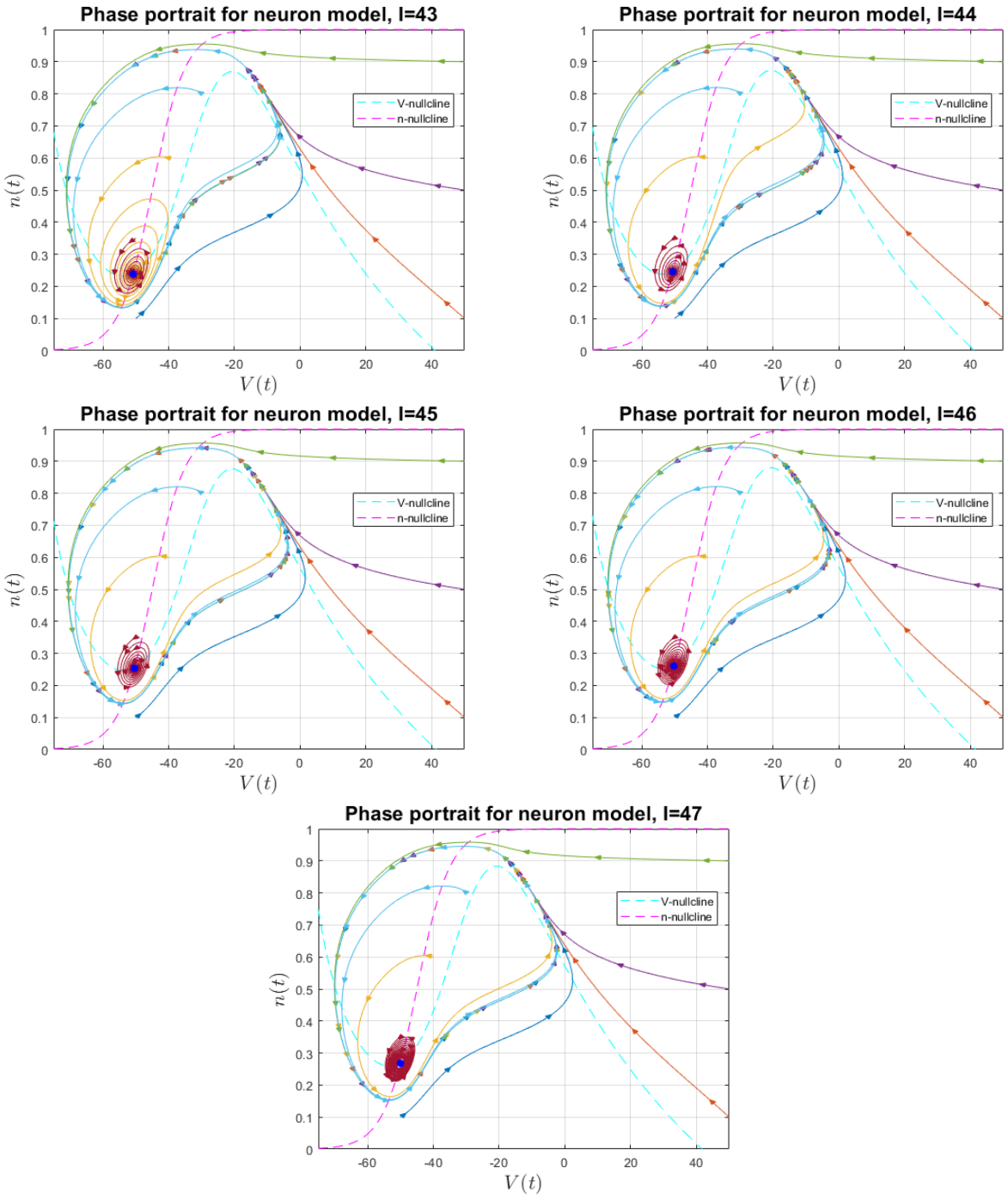

Figure 2: The first five phase portraits ($I = 43, \ldots, 47$) for (1) with parameters $C = 1$, $g_L = 1$, $E_L = -78$, $g_{Na} = 4$, $E_{Na} = 60$, $V_{1/2,m} = -30$, $k_m = 7$, $g_K = 4$, $E_K = -90$, $V_{1/2,n} = -45$, $k_n = 5$, and $\tau(V) = 1$ as I varies from 43 to 51.

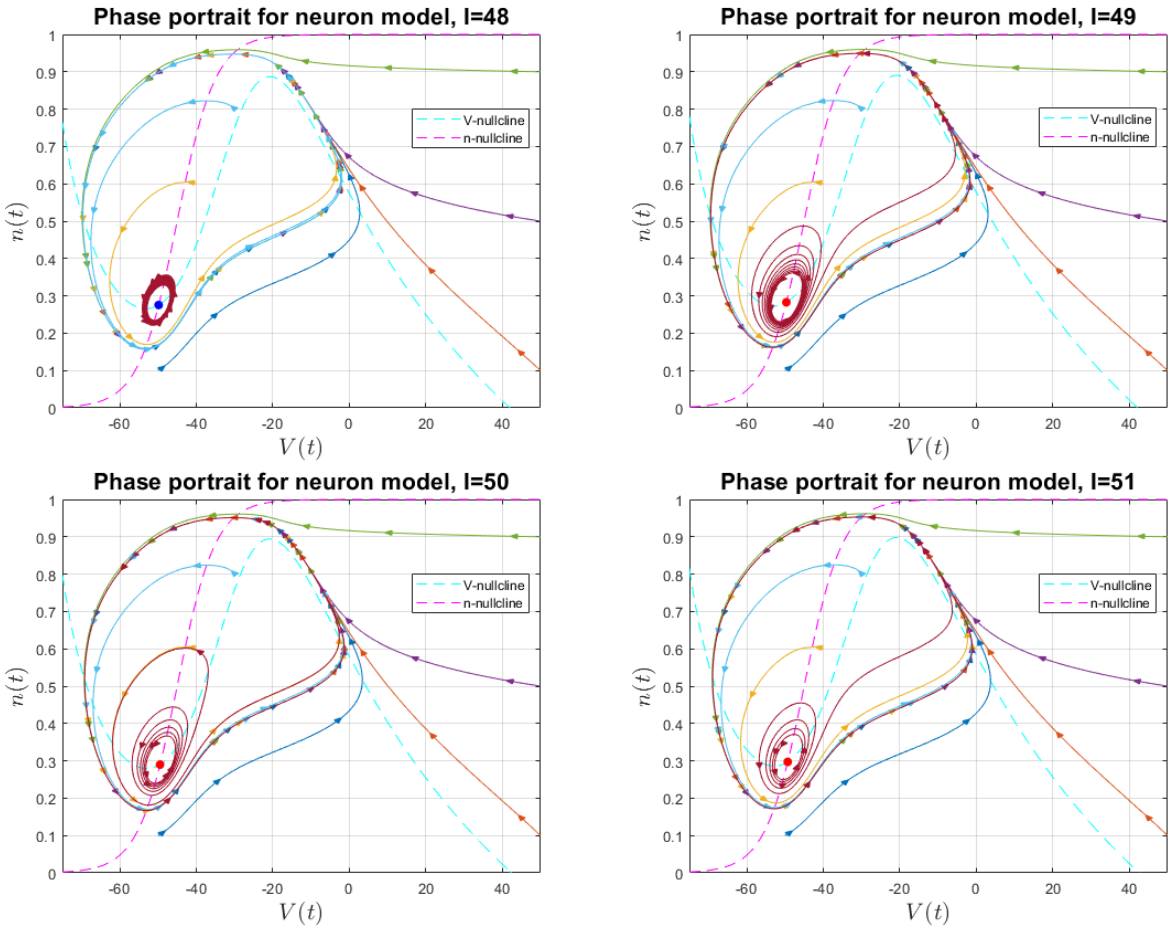

Figure 3: The last four phase portraits ($I = 48, \ldots, 51$) for (1) with parameters $C = 1$, $g_L = 1$, $E_L = -78$, $g_{Na} = 4$, $E_{Na} = 60$, $V_{1/2,m} = -30$, $k_m = 7$, $g_K = 4$, $E_K = -90$, $V_{1/2,n} = -45$, $k_n = 5$, and $\tau(V) = 1$ as I varies from 43 to 51.

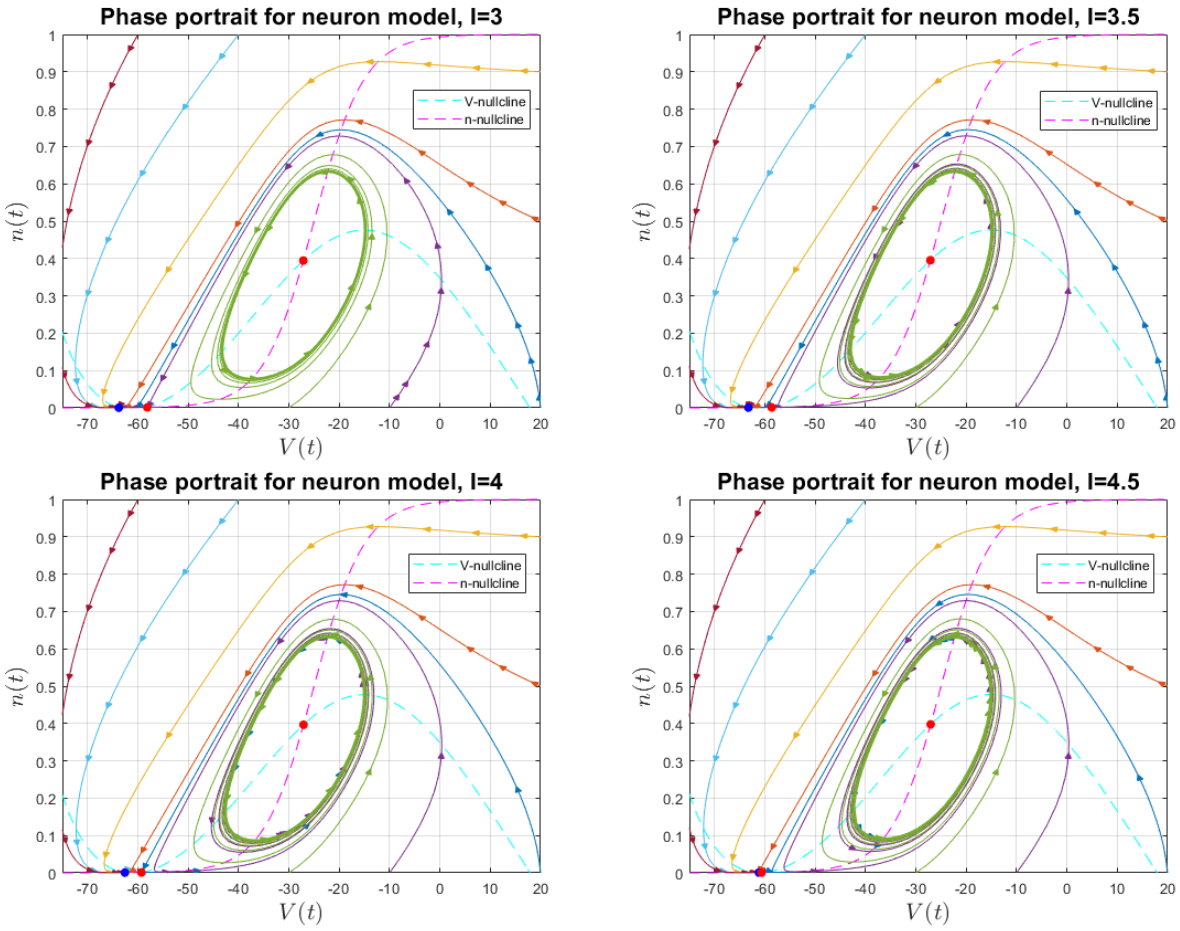

Figure 4: The first four ($I = 3, 3.5, 4, 4.5$) phase portraits for (1) with parameters $C = 1$, $g_L = 8$, $E_L = -80$, $g_{Na} = 20$, $E_{Na} = 60$, $V_{1/2,m} = -20$, $k_m = 15$, $g_K = 10$, $E_K = -90$, $V_{1/2,n} = -25$, $k_n = 5$, and $\tau(V) = 0.14$ as I varies from 3 to 6.

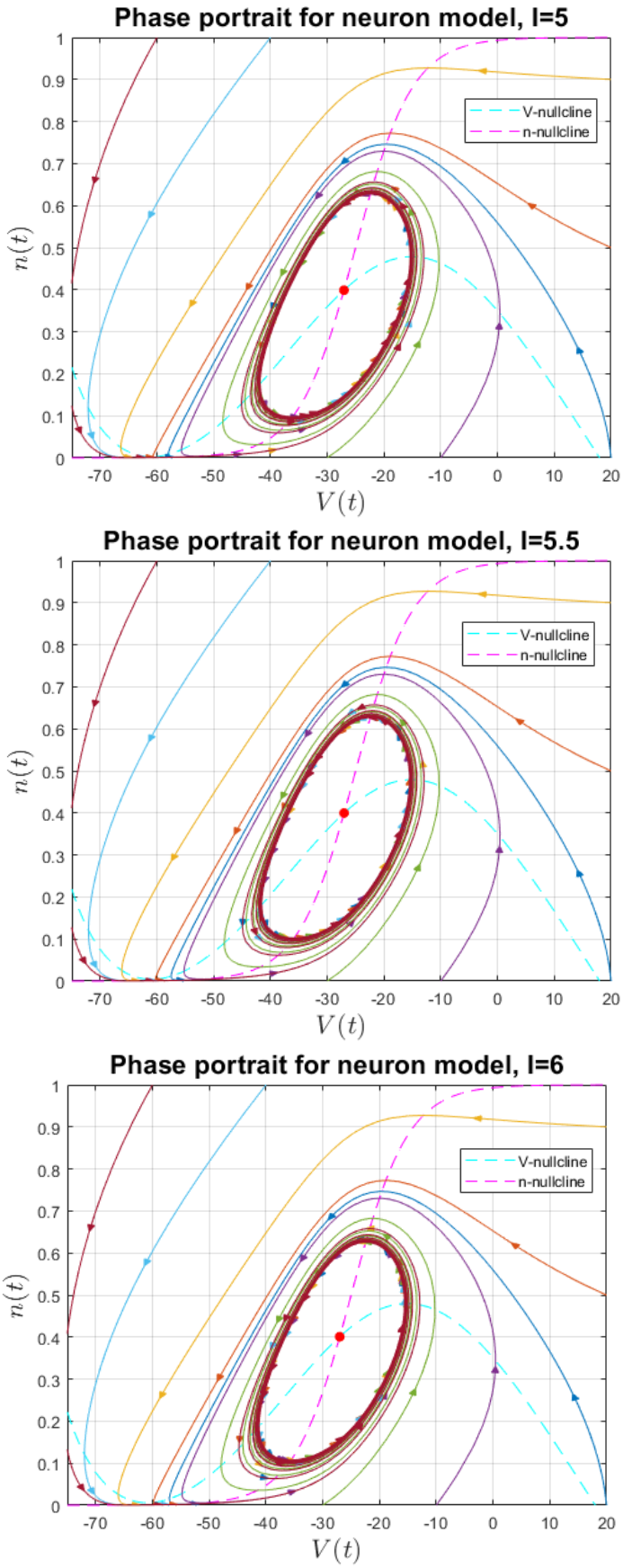

Figure 5: The last three ($I = 5, 5.5, 6$) phase portraits for (1) with parameters $C = 1$, $g_L = 8$, $E_L = -80$, $g_{Na} = 20$, $E_{Na} = 60$, $V_{1/2,m} = -20$, $k_m = 15$, $g_K = 10$, $E_K = -90$, $V_{1/2,n} = -25$, $k_n = 5$, and $\tau(V) = 0.14$ as I varies from 3 to 6.

```
gK = 10;
EK = -90;
Vn = -25;
kn = 5;
tauV = 0.14;

for I = Iq
    simulation(tol, V0, n0, tspan, Vq,...
    I, C, gL, EL, gNa, ENa, Vm, km, gK, EK, Vn, kn, tauV)
end
```

In all cases for I that we tested, we see a stable limit cycle centered around the unstable fixed point near $\approx (-27, 0.4)$ that hardly changes in shape or size as I varies. Of course, we know that such a stable limit cycle should exist by the Poincaré-Bendixson Theorem applied to the annular region surrounding the fixed point and just large enough such that it does not contain the two fixed points at $n = 0$. Furthermore, between $I = 4.5$ and $I = 5$, the two fixed points of opposite stabilities at $n \approx 0$ annihilate each other, and hence, we have a saddle-node bifurcation at some $I^* \in (4.5, 5)$.

Furthermore, the bifurcation at $I = I^*$ is *not* a saddle-node bifurcation on invariant circle (SNIC bifurcation) because the pair of fixed points at $n \approx 0$ are nowhere near the stable limit cycle which persists through $I > I^*$, nor do they lie on two heteroclinic orbits which connect a pair of unstable and stable fixed points when $I < I^*$. No new limit cycle appears when $I > I^*$. In fact, we have a stable limit cycle even when $I \leq I^*$, and the disappearance of these two fixed points allows trajectories that start at high enough values of n and low enough values of V (i.e., in the upper-left corner of the phase space domain) to also enter the limit cycle once $I > I^*$. When $I < I^*$, there is a trapping region for the limit cycle, and all trajectories which lie outside of this trapping region will go towards the stable fixed point which lies furthest to the left. Hence, in a sense, this stable fixed point "competes" with the attraction of the limit cycle and pulls nearby trajectories towards it when $I < I^*$. □