function [t,v1,v2] = ODE2D_Trajectory_Neuron(v1zero,v2zero,T,epsilon,w11,w12,w21,w22)

% function [t,v1,v2] = ODE2D_Trajectory_Neuron(v1zero,v2zero,T,epsilon,w11,w12,w21,w22)
% 
% Numerically integrate the system of equations
% 
% \[
% \frac{dv1}{dt} = -5v1(t) + w11*\text{atan}(v1(t)) + w12*\text{atan}(v2(t))
% \]
% \[
% \frac{dv2}{dt} = -1v2(t) + w21*\text{atan}(v1(t)) + w22*\text{atan}(v2(t))
% \]
% 
% from 0 to T using a time-spacing of epsilon and with the
% initial conditions \(v1(0) = v1zero\) and \(v2(0) = v2zero\).
% 
% Returns a vector \(t = [0:epsilon:T]\) of time points
% and vectors \(v1\) and \(v2\) (the same size as \(t\)) containing
% the corresponding values of the functions \(v1\) and \(v2\)
% 
% Compare to ODE_Demo_Func.m

% Generate a vector of epsilon-spaced times from 0 to T.

t = [0:epsilon:T];

% The number of values of \(v1\) and \(v2\) is the same as the number of values in \(t\).

n = length(t);

% Initialize the two vectors.

v1(1) = v1zero;
v2(1) = v2zero;

% Iterate through time and recursively update \(v1\) and \(v2\).

for k = 1:n-1
    v1(k+1) = v1(k) + epsilon * ( -5*v1(k) + w11*atan(v1(k)) + w12*atan(v2(k)) );
    v2(k+1) = v2(k) + epsilon * ( -1*v2(k) + w21*atan(v1(k)) + w22*atan(v2(k)) );
end

% Return to where the function was called.

return
% Set the synaptic weights for the two neuron system.

w11 = 17; w12 = -13;
w21 = 4; w22 = -5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Draw the DIRECTION FIELD for the two neuron system.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ODE2D_Quiver_Neuron(-10,1,10,-10,1,10,w11,w12,w21,w22)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot some sample TRAJECTORIES.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Set up the constants for numerical integration.
% Start at time 0 until time 50 with a time step of .01.

T = 50; epsilon = .01;

% Get the (approximate) trajectory starting from (8,8) using numerical
% integration.

[t,v1,v2] = ODE2D_Trajectory_Neuron(8,8,T,epsilon,w11,w12,w21,w22);

% Plot the trajectory on the same graph as the direction field.

hold on
plot(v1,v2)
hold off

% Get the trajectory starting from (8,-8) and plot on the same graph.

[t,v1,v2] = ODE2D_Trajectory_Neuron(8,-8,T,epsilon,w11,w12,w21,w22);
hold on, plot(v1,v2), hold off

% Get the trajectories starting from (-8,8), (-8,-8), (.5,.5), (-.5,-.5)
% and plot on the same graph.

[t,v1,v2] = ODE2D_Trajectory_Neuron(-8,8,T,epsilon,w11,w12,w21,w22);
hold on, plot(v1,v2), hold off

[t,v1,v2] = ODE2D_Trajectory_Neuron(-8,-8,T,epsilon,w11,w12,w21,w22);
hold on, plot(v1,v2), hold off

[t,v1,v2] = ODE2D_Trajectory_Neuron(-8,-8,T,epsilon,w11,w12,w21,w22);
hold on, plot(v1,v2), hold off

[t,v1,v2] = ODE2D_Trajectory_Neuron(.5,.5,T,epsilon,w11,w12,w21,w22);
hold on, plot(v1,v2), hold off

[t,v1,v2] = ODE2D_Trajectory_Neuron(-.5,-.5,T,epsilon,w11,w12,w21,w22);
hold on, plot(v1,v2), hold off
Figure 1: Screen output for ODE2D_Neuron_Demo.m. The thick line is the trajectory starting from (8, 8). The dots show the initial conditions. (The code for these illustrative features is not shown.)
Linear System:
\[
\dot{v}_1 = av_1 + bv_2 \\
\dot{v}_2 = cv_1 + dv_2
\]

Figure 2: Direction field and sample trajectories for linear system with \((a, b, c, d) = (1, -1, 1, 3)\). Initial starting points are all near \((0, 0)\). Trajectories went too far at first (see below), but `axis([-6 6 -6 6])` crops like above.