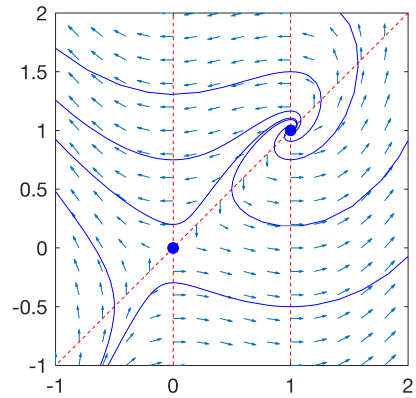
MECH 221 Computer Lab 7

Phase Portraits and Trajectories of 2D Nonlinear Systems




```
>> f1 = @(x1,x2) x1 - x2;
>> f2 = @(x1,x2) x1 .* (x1 - 1);
>> nonlinear_phase_portrait(f1, f2, [-1,2], [-1,2], 0.2)
Find a steady state? (y/n) y
Enter an initial guess [x1,x2]: [0,0.1]
Find a steady state? (y/n) y
Enter an initial guess [x1,x2]: [1.1,1]
Find a steady state? (y/n) n
Plot a trajectory? (y/n) y
Enter initial condition [x1,x2]: [0.5,0.5]
Plot a trajectory? (y/n) y
Enter initial condition [x1,x2]: [1,1.5]
Plot a trajectory? (y/n) y
Enter initial condition [x1,x2]: [0,0.75]
Plot a trajectory? (y/n) y
Enter initial condition [x1,x2]: [1,-0.5]
Plot a trajectory? (y/n) y
Enter initial condition [x1,x2]: [-0.5,-0.5]
Plot a trajectory? (y/n) y
Enter initial condition [x1,x2]: [0,0.2]
Plot a trajectory? (y/n) y
Enter initial condition [x1,x2]: [1,0.75]
Plot a trajectory? (y/n) n
```

Instructions

Write a function called `nonlinear_phase_portrait` which takes 5 input parameters:

```
function nonlinear_phase_portrait(f1, f2, x1_range, x2_range, step)
% Plot the phase portrait of the nonlinear system dxdt = [f1(x); f2(x)]
```

where:

- ☐ `f1` and `f2` are vectorized function handles for scalar functions representing the right side of a 2-dimensional, autonomous, nonlinear system

$$\begin{aligned}\dot{x}_1 &= f_1(x_1, x_2)\\ \dot{x}_2 &= f_2(x_1, x_2)\end{aligned}$$

- ☐ `x1_range` is a vector of length 2, the range of x_1 values of the figure window
- ☐ `x2_range` is a vector of length 2, the range of x_2 values of the figure window
- ☐ `step` is the space between arrows in the slope field

The function performs the following tasks:

- ☐ Plot the slope field of the nonlinear system

$$\begin{aligned}\dot{x}_1 &= f_1(x_1, x_2)\\ \dot{x}_2 &= f_2(x_1, x_2)\end{aligned}$$

  using the MATLAB function `quiver`
- ☐ Plot the x_1-nullcline $f_1(x_1, x_2) = 0$ and x_2-nullcline $f_2(x_1, x_2) = 0$ using the MATLAB function `fimplicit`
- ☐ Ask the user to find steady state values:
  - ☐ Use `input` to ask the user for an initial guess for a steady state value
  - ☐ Use `fsolve` to find the nearest steady state $f_1(x_1^*, x_2^*) = f_2(x_1^*, x_2^*) = 0$
  - ☐ Plot the steady state as a dot in the figure
  - ☐ Repeat until the user declines to find another steady state
- ☐ Ask the user to plot trajectories:
  - ☐ Use `input` to ask the user for an initial value for the trajectory
  - ☐ Use `ode45` to plot the trajectory on the interval $(-\infty, \infty)$ (ie. on the intervals `[0,Inf]` and `[0,-Inf]`)
  - ☐ Use an event function to stop `ode45` if the trajectory leaves the display window or converges to a steady state
  - ☐ Repeat until the user declines to plot another trajectory

When you are satisfied with your function, submit your M-file (called `nonlinear_phase_portrait.m`) to Connect.

Hints

1. Look at the documentation for the MATLAB function `quiver` and how we used it in the previous lab:

   `https://www.mathworks.com/help/matlab/ref/quiver.html`

2. Look at the documentation for the MATLAB function `fimplicit`:

   `https://www.mathworks.com/help/matlab/ref/fimplicit.html`

3. Look at the documentation for the MATLAB function `input`:

   `https://www.mathworks.com/help/matlab/ref/input.html`

   Notice that user input is evaluated as an MATLAB expression. Use the option `'s'` to specify that the input is a character string.

4. Use a `while` loop to repeatedly ask the user for input until they decline. For example, consider the following code:

```
response = 'y'
while response == 'y'
    disp('Hello!')
    repsonse = input('Say hello?  (y/n) ', 's')
end
```

5. Look at the documentation for the MATLAB function `fsolve` (which requires the optimization toolbox):

   `https://www.mathworks.com/help/optim/ug/fsolve.html`

   Note that it is necessary to combine the functions `f1` and `f2` into a single vector function `F` to use with `fsolve` to find a steady state. Also, the function `fsolve` displays information about the process of finding a solution of the equation. Set the option `Display` as `none` to suppress the output. For example, consider the code:

```
F = @(x) [f1(x(1),x(2)); f2(x(1),x(2))];
options = optimoptions('fsolve', 'Display', 'none');
x0 = [0,0]; % Initial guess
steady_state = fsolve(F, x0, options);
```

6. Look at the documentation for ODE event location functions and how we used them in the previous lab:

   `https://www.mathworks.com/help/matlab/math/ode-event-location.html`

   Create an event function to pass as an option in `ode45`. Event functions force `ode45` to stop if the solution satisfies some criteria. Use the following function as an option when calling `ode45`. It forces `ode45` to stop if the solution goes beyond the size of the figure window, and it also stops if the fucnctions $f_1(x_1, x_2)$, $f_2(x_1, x_2)$ get very close to 0:

```
odefun = @(t,x) [f1(x(1),x(2)); f2(x(1),x(2))];
   function [value,isterminal,direction] = stop(t,y)
        value = [1,1];
        x_mid = mean(x_range);
        y_mid = mean(y_range);
        position = y - [x_mid; y_mid];
        if norm(position) > norm([diff(x_range),diff(y_range)])/2
            value(1) = 0;
        end
        if norm(odefun(t,y)) < 1e-4
            value(2) = 0;
        end
        isterminal = [1,1];
        direction = [0,0];
   end
options = odeset('Events', @stop);

[T,U] = ode45(odefun, [0,Inf], y0, options);
plot(U(:,1),U(:,2))
[T,U] = ode45(odefun, [0,-Inf], y0, options);
plot(U(:,1),U(:,2))
```