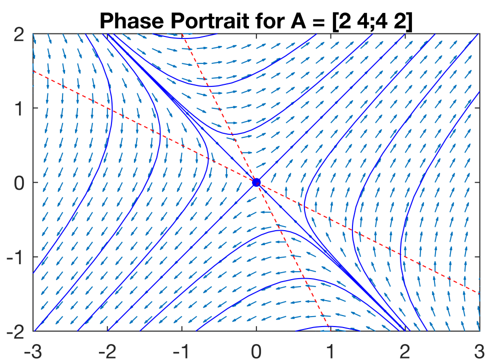
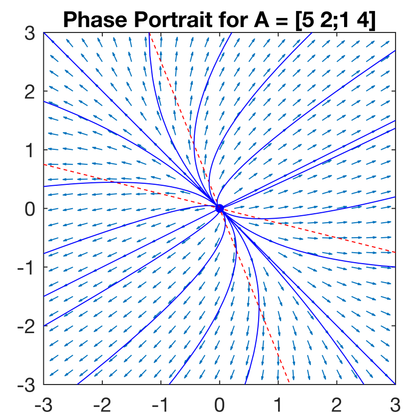
MECH 221 Computer Lab 6

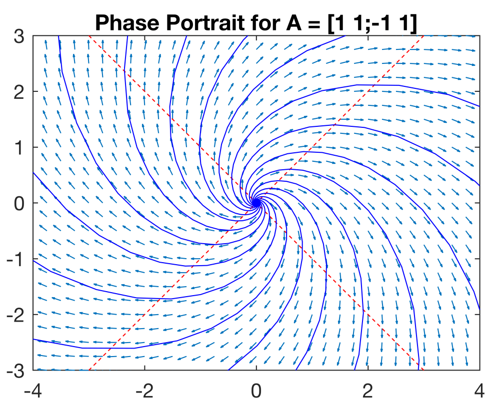
Phase Portraits and Trajectories of 2D Linear Systems

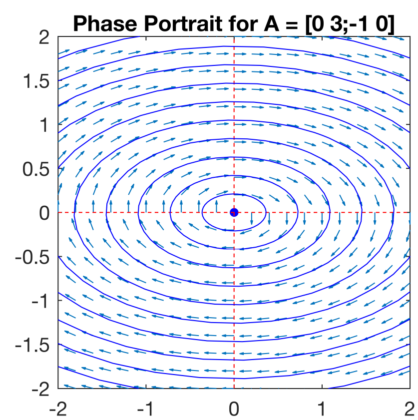





```
>> phase_portrait([2,4;4,2],3,2,0.2,12);
```

```
>> phase_portrait([5,2;1,4],3,3,0.25,15)
```






```
>> phase_portrait([1,1;-1,1],4,3,0.25,5)
```

```
>> phase_portrait([0,3;-1,0],2,2,0.2,10)
```

Instructions

Write a function called phase_portrait which takes 5 input parameters:

```
phase_portrait(A,width,height,h,N)
% Plot the phase portrait of the linear system dxdt = Ax
```

where:

- ☐ A is a 2 by 2 matrix
- ☐ width is the width of the figure window
- ☐ height is the height of the figure window
- ☐ h is the space between arrows in the slope field
- ☐ N is the number of plotted trajectories (excluding eigenvector lines)

The function performs the following tasks:

- ☐ Plot the slope field of the linear system $\dot{\mathbf{x}}(t) = A\mathbf{x}(t)$ using the MATLAB function quiver (see Hints below)
- ☐ Plot the x_1-nullcline and x_2-nullcline as dotted lines
- ☐ If the eigenvalues of A are real, then plot the eigenvectors as solid lines
- ☐ Plot N sample trajectories

When you are satisfied with your function, submit your M-file called phase_portrait.m to Connect.

Hints

1. Look at the MATLAB documentation (including the examples) for the function quiver:

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

   The function is usually used with the MATLAB function meshgrid:

   https://www.mathworks.com/help/matlab/ref/meshgrid.html

   For example:

   ```
   [X,Y] = meshgrid(-2:0.2:2,-1:0.2:1);
   U = X ./ sqrt(X.^2 + Y.^2);
   V = Y ./ sqrt(X.^2 + Y.^2);
   quiver(X,Y,U,V,0.5)
   ```

The code above performs the following tasks:

- ☐ Create matrices (of the same size) X and Y where the *rows* of X are values $-2, -1.8, \ldots, 1.6, 1.8, 2$, and the *columns* of Y are values $-1, -0.8, \ldots, 0.6, 0.8, 1$. Together, the entries in X and Y give the coordinates of points in the grid $-2 \le x \le 2$, $-1 \le y \le 1$ with step size 0.2.
- ☐ Create matrices U and V from the coordinates X and Y. The matrices U and V represent the components of the arrows plotted by `quiver`. In particular, the matrix U contains the x-components and V contains the y-components of the arrows.
- ☐ The function `quiver` plots an arrow at each coordinate specified by X and Y, and each arrows has components defined by U and V.

In the function `phase_portrait`, use `meshgrid` to create the grid of coordinates (matrices `X` and `Y`) using the variables `width`, `height` and `h`, and use the matrix `A` with `X` and `Y` to create the matrices `U` and `V`, and then plot the slope field with `quiver`.

2. To plot the x-nullcline from one edge of the figure window to the other edge, simply plot a line of length $2L$ (centered at the origin) in the direction of the nullcline where L is the length from the origin to the corner of the figure window, and then restrict the x and y limits of the plot. For example:

```
L = sqrt(width^2 + height^2)
xnullcline = L * [-A(1,2), A(1,1)] / norm([-A(1,2), A(1,1)])
plot([-xnullcline(1),xnullcline(1)],[-xnullcline(2),xnullcline(2)],'r--')
xlim([-width,width])
ylim([-height,height])
```

Apply the same idea to the y-nullcline as well as the eigenvectors.

3. Use the function `eig` to compute the eigenvalues and eigenvectors of the matrix `A`. Use the functions `real` and `imag` to extract the real and imaginary parts of the eigenvalues as needed.

4. It's possible to use the formulas for the solution of the system $\dot{\mathbf{x}} = Ax$ to plot trajectories, but it's easier to just use `ode45` to find numerical solutions. It's up to you to decide how to plot trajectories. The following describes the method used to create the figures on the cover page above. When plotting trajectories, consider separate cases. For example:

- ☐ If the origin is a sink (or spiral sink), choose `N` evenly spaced initial positions around the circle of radius `L` (where `L = sqrt(width^2 + height^2)`). Then integrate using `ode45` from 0 to ∞ (that is, use `tspan = [0,Inf]` in `ode45`) using the stopping condition described below.
- ☐ If the origin is a source (or spiral source), again choose `N` evenly spaced initial positions around the circle of radius `L`. Then integrate using `ode45` from 0 to $-\infty$ (that is, use `tspan = [0,-Inf]` in `ode45`) using the stopping condition described below.
- ☐ If the origin is a saddlepoint, choose `N/2` initial positions along each nullcline. Then integrate using `ode45` from 0 to ∞ as well as 0 to $-\infty$ using the stopping condition described below.

☐ If the origin is a center, choose `N` evenly spaced initial positions along the diagonal from the origin to the corner of the figure window. Then integrate using `ode45` from 0 to $2\pi/\beta$ where β is the imaginary part of the eigenvalue $\lambda = i\beta$ (in other words, β is the frequency in the general solution of the system).

5. Create an event function to pass as an option in `ode45`. Event functions force `ode45` to stop if the solution satisfies some criteria. First, look at the documentation:

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

   In the function `phase_portrait`, 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 solution gets very close to 0.

```
    function [value,isterminal,direction] = stop(t,y)
        value = [1,1];
        if norm(y) > norm([width,height])
            value(1) = 0;
        end
        if norm(y) < 1e-5
            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))
```