**Lab 5: Exploring second-order differential equations numerically**

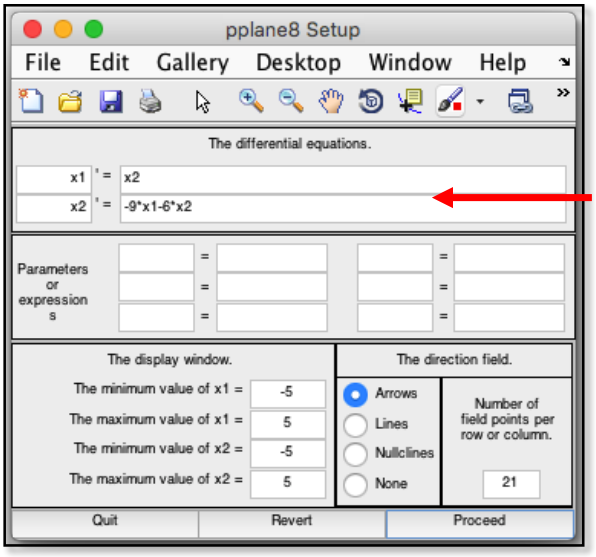
**Introduction:** In this lab, we will explore second-order differential equations using MATLAB. For numerical explorations of such equations, **pplane8** is very useful. You will also get to look "under the hood", and build your own custom version.

**Questions 1-2:** Consider the initial value problem:

$$y'' + 6y' + 9y = 0 \quad y(0) = 1 \quad y'(0) = 0$$

Note this 2nd-order equation is linear and homogeneous.
Using pplane8, create a phase diagram, for this DE.

You can enter the DE into pplane8 using the two equations: $x'_1 = x_2$ $x'_2 = -9x_1 - 6x_2$



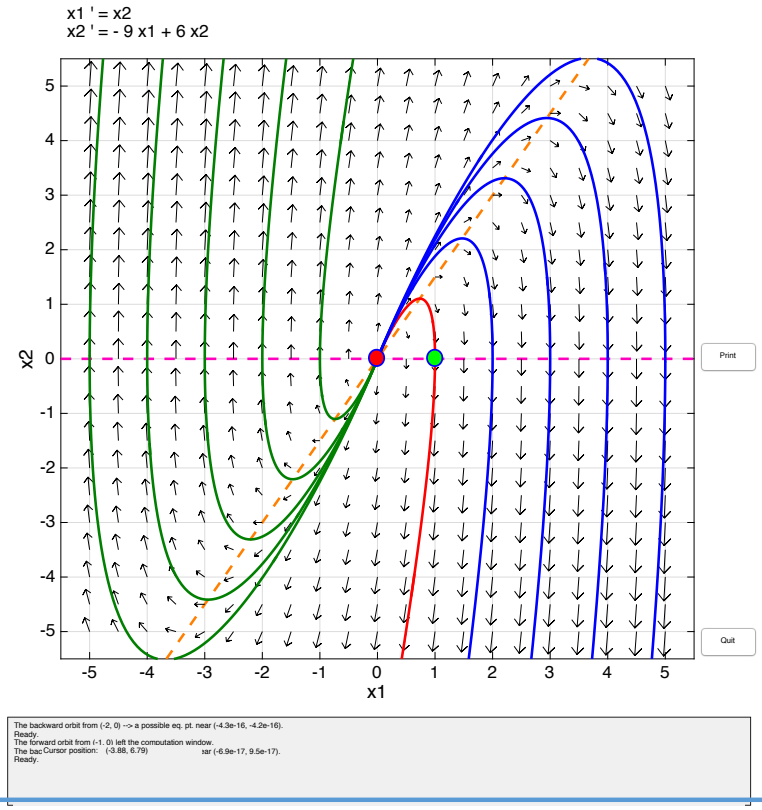

x1' = x2
x2'= -9*x1 - 6*x2

Using **Keyboard input**, find the solutions satisfying $y(0) = n$, $y'(0) = 0$ for each value of $n$ from –5 to +5, for a total of 11 solutions. The one through (0,0) is just the equilibrium point.

Use the Property Editor to set the curve through (1,0) to **red**. Note that all solutions do indeed approach the equilibrium point (0,0). Under **Solutions**, select **Show nullclines**. Notice all solution curves that cross the nullclines do so at a local max or min. Using the Insert menu choice, place a green circle (ellipse) at the starting point (1,0) and place a second red circle at the equilibrium point (0,0). (Red, for stopping point.)

**Questions 1-2: Paste your completed phase plot here.** Be sure the solution with $y(0) = 1,\ y'(0) = 0$ is shown in **red**. Be sure the two null lines are shown.
**Replace** this sample plot with your own plot which will look different as you are using different coefficients.
The sample plot is for the different DE: $y'' - 6y' + 9y = 0$ (Note the sign change.)
You should use the +6 value instead.

**Sample for a modified DE.**




**Questions 3-4-5: Build your very own custom pplane 8 in three simple steps!**

Now that you have explored the solutions **qualitatively** using **pplane8.m**, it's time to "look under the hood" so to speak. Let's try to make our own "**pplane8**". We'll need to plot the direction field and solution curves given a number of initial conditions. For the solution curves, we'll use **ode45**. In the next portion of this lab, we'll try to duplicate the graph you just obtained using **pplane8** from scratch – and even add a few bells and whistles!

The code below shows how you can create a direction field for any second-order differential equation. In practice, you will need to tweak the settings to obtain a polished look. I'll give the code in three blocks. Each block should remind you of settings that are given when you first open **pplane8**.

**Block 1:** Start with a differential equation of the form: $ay'' + by' + cy = 0$
(It doesn't matter what you name the variables, as long as you are consistent.)
For now, we'll treat *a*, *b* and *c* as constants but that can be relaxed later. We'll need to recast the DE in state vector form:

$$\vec{\mathbf{x}} = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} y(t) \\ y'(t) \end{bmatrix}$$

Using the original second-order DE we find:

$$x_1' = x_2$$

$$x_2' = -\frac{c}{a}x_1 - \frac{b}{a}x_2$$

Let's create a block to enter our differential equation: $y'' + 6y' + 9y = 0$

In our case,

$$a = 1, b = 6, c = 9$$

```
%% Custom Direction Field for second-order differential equation:
%  ay'' + by' + cy = 0
clear, clc, close all
% BLOCK 1: Convert 2nd-order DE to a system of first order DEs.
% The new system will have the form dx/dt = F(t, x) where x is the state
% vector. So, x = [x1; x2]

% Define the constants here.
a=1; b=6; c=9

% Define F(t, x) as an anonymous function.
% Below, x is a column vector with two components
F = @(t, x) [x(2); -(c/a) .* x(1) - (b/a) .* x(2)]
```

Notice we have defined the slope field $F(t, \vec{\mathbf{x}})$ using an **anonymous function**. Be sure to include the variable t, even though this particular DE is time-independent, as we will need that to invoke **ode45** to plot the solutions.

---

**Block 2:** Setting up your plotting window.
Much of the code below is just to polish the appearance of the resulting graph.
Adjust the parameters as desired.

```
% BLOCK 2: Set the plot window dimensions here.
% Customize the plot settings.
x1min = -5; x1max= 5;  x2min = -5;  x2max =5;
figure % Pop up a new figure
axis([x1min x1max x2min x2max ])
axis square
grid on; hold on
set(gca, 'FontSize', 20)
title('Second Order DE')
xlabel('x1 = y'); ylabel('x2 = dy/dt')
```

**Block 3:** Graph your Direction Field as a grid of tick marks with slope defined by the RHS of $\frac{d\vec{x}}{dt} = F(t, \vec{x})$
Focus on the **nested for loops** which is the heart of where the tick marks for the direction field are drawn.
We will soon encounter a problem which you will be asked to fix.

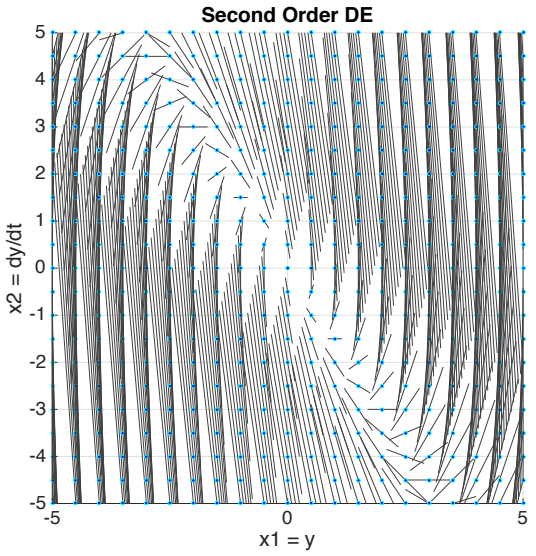
```
%% BLOCK 3 - The Direction Field
radius = 0.1                % Control the size of the tick marks
spacing_horizontal = 0.5;   % Control their horizontal spacing.
spacing_vertical = 0.5;     % Control their vertical spacing.
my_color = [0.25, 0.25, 0.25];  % Control their color.
optional_dots = 1;           % Control whether a dot is placed. Use 0 or 1


time = 0
for x1p =x1min: spacing_horizontal: x1max
    for x2p = x2min: spacing_vertical: x2max
        x=[x1p;x2p]; % Update the state vector x.
        dx = F(time,x) ; % Find dx/dt
        % Add a line here to replace dx by a unit vector.

        dx1 = dx(1) *radius;  dx2 = dx(2)*radius; % Scale tick marks.
        % Plot the tick marks!
        plot([x1p - dx1 , x1p + dx1 ], [x2p - dx2 , x2p + dx2] , 'Color', my_color);

        if optional_dots
            plot( x1p, x2p, 'cyano', 'MarkerSize', 4,  'MarkerFaceColor', 'blue')
        end
    end
    pause(0.1)
end
```

**Question 3:** Once you have all three blocks entered, run them to produce this direction field for our 2nd-order DE. Fix this graph so all tick marks have the same length.




**Ack! Not so good.**
The problem is some of the tick marks for **dx** are very long, resulting in a confusing picture. Where you see the red arrow above, add a line of code to convert the vector dx, into a **unit vector** by dividing it by its length.
Use the **sqrt()** command.
Then adjust the **radius** until you are satisfied with your tick marks.

Next, we need to add some solution curves. And not just one! Let's add a whole bunch.

Here is some code, which will plot **one solution** moving forward in time from the initial point (1,0).

```
% Plot a solution moving forward in time from the given initial point.
tStart = 0;  tEnd = 10;
both_directions = 1;      % Change to 0 to draw in the forward direction only.
show_initial_point = 1;   % Change to 0 to suppress drawing the initial points.


x0 = [1;0];
[~, x_out] = ode45(F, [tStart, tEnd], x0);
plot(x_out(:,1), x_out(:,2), 'blue', 'LineWidth', 3)
% Show initial point using a green dot.
if show_initial_point
    plot(x0(1), x0(2), 'bo', 'MarkerSize', 8, 'MarkerFaceColor', 'green')
end
```

**Question 4: Forward Solutions:** Using a for loop, plot solutions forwards in time, from the 11 points $\vec{\mathbf{x}}(0) = \begin{bmatrix} n \\ 0 \end{bmatrix}$ as *n* varies from –5 up to +5 in steps of 1. Draw each forward solution in **blue** (as above) and indicate the starting point using a green dot.

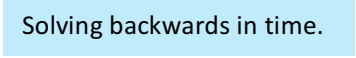
**Hint:** No need to repeat the first three lines above. But the lower part, should be revised to fit inside the for loop:

```
for n = -5:5
      x0 = [n; 0]
      % Block of code here
end
```

**Question 5: Solutions moving backwards in time:** Using another for loop, plot solutions backwards in time, from the same 11 points in the previous part. Draw each backwards solution in **red**. Use an if statement so this is only done if the variable `both_directions is 1.`

**Hint:** Clone your solution from the last part, but adjust the call to **ode45** as follows.

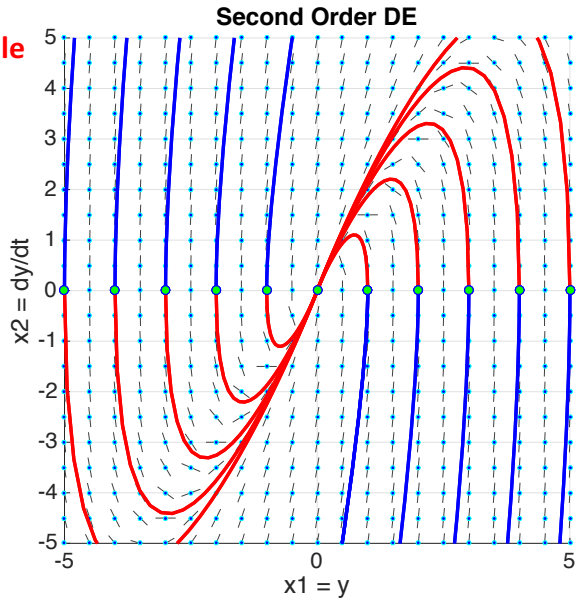
```
[~, x_out] = ode45(F, tStart: -0.05: -tEnd, x0); % Backwards!!!
```



Solving backwards in time.

**Questions 3-4-5: Replace this sample image with your completed graph.**
Your graph will look different, because the sample uses a different DE, with the sign of *b* reversed.

**Sample**




Grader will award **three** points as follows.
**i.** The tickmarks all have the same length.

**ii.** Forward solutions are all shown in blue, and green marker dots appear at each initial point.

**iii.** Solutions backwards in time are shown in red.

Note your blue and red curves will be the reverse of the sample.

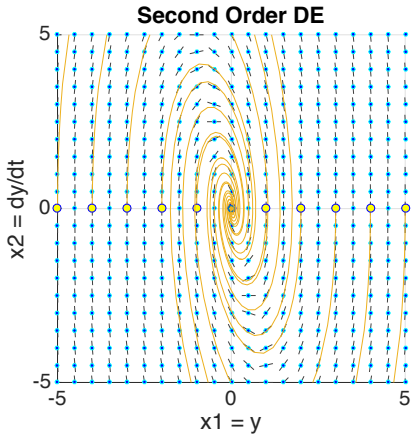
**Question 6: Comet Plots!** Start over, and run the parts necessary to produce the direction field - but **don't show any solutions**, just the direction field. Now, locate your for loop for the forward solutions only. Copy it, and paste it in a new section below all your other code. Now **replace** the plotting command for the solution curves with the **comet** command, which has the synyax:
>> comet(x,y)
Above, the x and y vectors are the columns inside x_out, which we get from the **ode45** command.

```
comet(x_out(:,1), x_out(:,2))
```

**Tip:** You'll need to redefine tStart and tEnd too.
Change the marker dots to **yellow**. Be sure to watch the comets whiz around to get a good feeling for how solution curves follow the direction field. Enjoy your comet animation!

**Question 6:** **Replace** this sample comet plot with your completed comet plot. Yours will look quite different than this sample which is for a DE with different coefficients.

**Sample for a different DE:**



**Question 7: Find the exact solution to your DE $y'' + 6y' + 9y = 0$, for the initial point: $\vec{\mathbf{x}}(0) = \begin{bmatrix} n \\ 0 \end{bmatrix}$**

Here's some starter code:

```
%% Question 7:  Exact solution:
clear, clc
syms y(t) n
Dy = diff(y, t);  D2y = diff(y,t,t);
```

Now build your DE and find the solution using **dsolve**. Answer will be a multiple of *n*.

**Question 7:** The exact solution for $y'' + 6y' + 9y = 0$ satisfying $\vec{\mathbf{x}}(0) = \begin{bmatrix} n \\ 0 \end{bmatrix}$ is:

$y(t) =$ _____

**Questions 8-10: Van der Pol Oscillator**

Let's start over with a new DE, but now one that is non-linear. It can be considered a mass-dashpot-spring system with non-linear damping. You can read more about this famous oscillator here:

https://en.wikipedia.org/wiki/Van_der_Pol_oscillator

**Van der Pol Oscillator:** $y'' - \left(\frac{1-y^2}{2}\right) \cdot y' + y = 0$

So now $a = 1,\ b = -\left(\frac{1-y^2}{2}\right), c = 1$

The middle term is non-linear!

**Question 8:** Start over in Block 1, and redefine F(t,x) so it corresponds to this new non-linear system. Don't destroy your old answer. Just comment it out, and add the new function. Or you can clone all your code by pasting it at the bottom of your script.

In terms of the state vector representation, you can see that if:

$$\vec{\mathbf{x}} = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} y(t) \\ y'(t) \end{bmatrix} \quad \text{then} \quad \frac{d}{dt}\vec{\mathbf{x}} = \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y' \\ y'' \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_1 + (1 - x_1^2)x_2/2 \end{bmatrix}$$

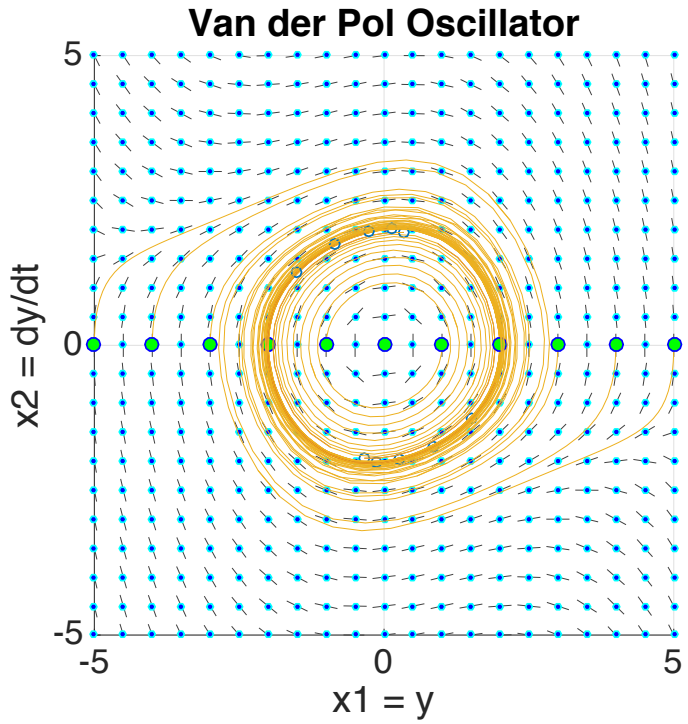
**Tip:** You will need to make careful use of pointwise operations by placing a dot before any ^ or *. You are just changing the code for the anonymous function F for question 8.

**Question 9: Change the title to 'Van der Pol Oscillator' in Block 2.**

**Question 10: Using comet plots, find the solution for the same 11 points we used above.**
$\vec{\mathbf{x}}(0) = \begin{bmatrix} n \\ 0 \end{bmatrix}$ as *n* varies from –5 up to +5 in steps of 1. Adjust the stopping time tEnd to a larger number like 30, so you can see that all curves approach the same **limit cycle**.

**Questions 8-9-10: Replace this sample plot, which uses a different constant for the middle term with your completed comet animation for the Van der Pol Oscillator.** Grader will award one point for each of the three steps 8-9-10.

**Sample** using this slightly **different** oscillator: $y'' - \left(\frac{1-y^2}{8}\right) \cdot y' + y = 0$




Suggested Grades: i. Curves look good ii. Title is correct iii. Traces are comets. (not plots)

Done? Place your answers in the Answer Template and Submit before the deadline.
Be sure all questions are answered, then submit your answers for this lab as a single PDF.
Submission must be a single PDF file! Only one submission is allowed.