Assignment 9 ENGR 220

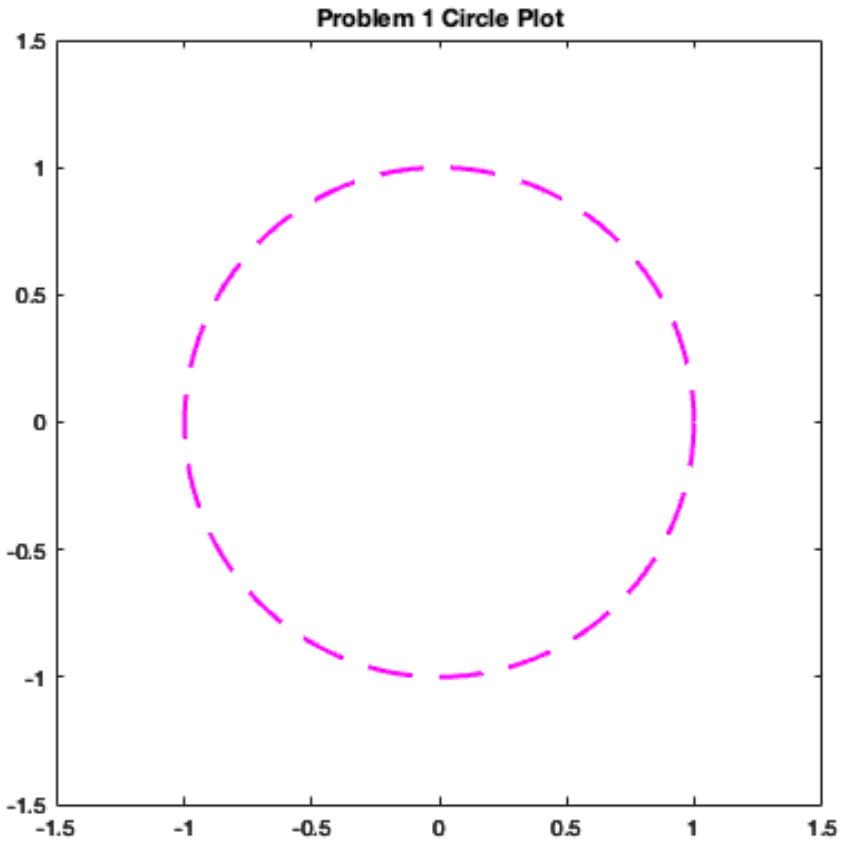
Table of Contents

Ethan Vosburg

Problem 1

Write a MATLAB program to plot a circle using a loop. While (no pun intended) you can use any method to generate and plot the circle, try this method that I used first to get practice concatenating vectors:

```
% Problem 1 variables
p1counter = 1;
p1r = 1;
p1theta = 0;

% Problem 1 loop
for p1counter = 1:91
    p1x(p1counter) = p1r*cosd(p1theta);
    p1y(p1counter) = p1r*sind(p1theta);
    p1theta = p1theta - 4;
end

% Problem 1 plot
figure(10)
problem1Plot = axes;
plot(p1x,p1y,'m--','LineWidth',2);
axis(problem1Plot, [-1.5 1.5 -1.5 1.5]);
axis(problem1Plot, 'square');
title(problem1Plot, 'Problem 1 Circle Plot');
```

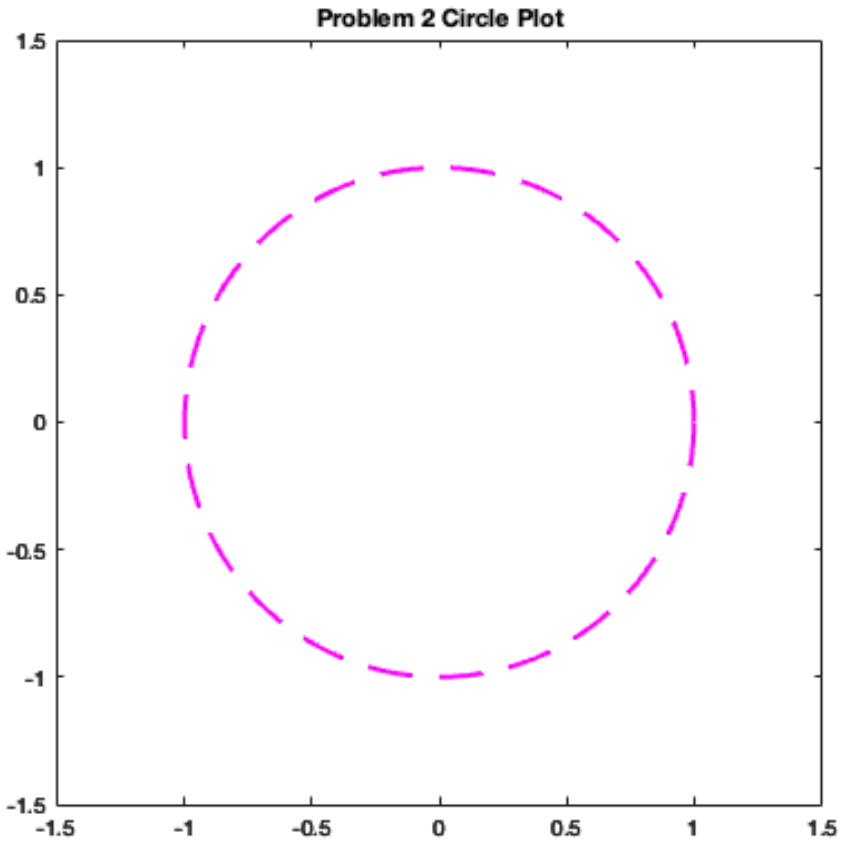


Problem 2

Now that you have used concatenation to create the x & y vectors to plot your circle, notice the "bug" or error message on the right side of the editor window that says: "The variable 'circlex' appears to change size on every loop iteration. Consider preallocating for speed." It turns out concatenating vectors for this program may not be the best programming practice. However, concatenation is a useful skill in some programming situations. Now that you know how to do it, let us improve the "speed" of the program as MATLAB has suggested. "Preallocating" means creating a vector before entering the loop that is the same size of the vector created when the loop completes the concatenation. For example, if your intent is to create x & y vectors with 91 elements, start with a vector of 91 of anything (the zeros(_) or ones(_) functions are useful here), and every time you go through the loop each new generated point will replace or overwrite its corresponding position in the "preallocated" vector. For keeping track of the size of your vectors if you want to vary the number of points around the circle (instead of "hard coding" the number into your code), the size(_) function may be useful.

```
% Write new code in your script file for this new method—do not replace the
 concatenation code. Instead, copy and paste your code from Problem 1 into
 a new section (%%) for Problem 2, then overwrite the code.  After you have
 written your new code with the preallocated vectors, notice the "bug" or
 error message disappears.

% Problem 2 variables
p2counter = 1;
p2r = 1;
```

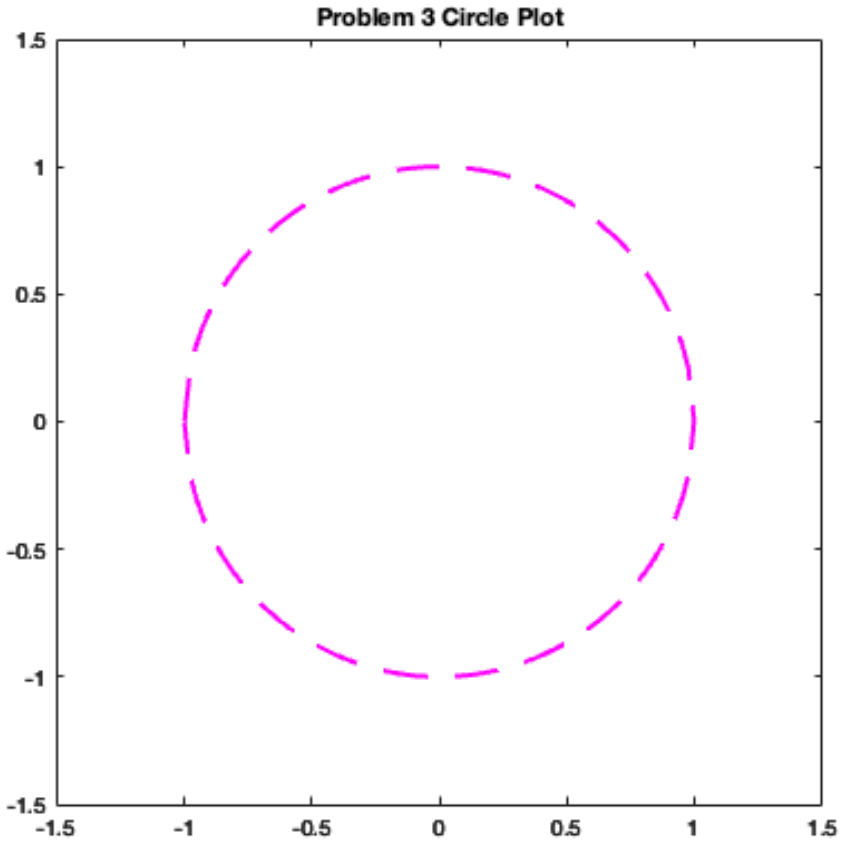
```
p2theta = 0;

% Problem 2 preallocated vectors
p2x = zeros(1,91);
p2y = zeros(1,91);

% Problem 2 loop
for p2counter = 1:91
    p2x(p2counter) = p2r*cosd(p2theta);
    p2y(p2counter) = p2r*sind(p2theta);
    p2theta = p2theta - 4;
end

% Problem 2 plot
figure(20)
problem2Plot = axes;
plot(x,y,'m--','LineWidth',2);
axis(problem2Plot, [-1.5 1.5 -1.5 1.5]);
axis(problem2Plot, 'square');
title(problem2Plot, 'Problem 2 Circle Plot');
```



Problem 3

In this problem you will replace the “loop method” of creating the x & y vectors used to plot the circle with an even better (faster) program. Again, copy and paste your code into a new section for Problem 3, and then replace the loop with code to create the vectors using the linspace(_) function. You will see this is much easier, more efficient, and faster. Better programming practice.

```
% Problem 3 x and y vectors
p3x = linspace(-1,1,91);
p3y = sqrt(1 - p3x.^2);
p3y = [p3y, -p3y];
p3x = [p3x, linspace(1,-1,91)];

% Problem 3 plot
figure(30)
problem3Plot = axes;
plot(p3x,p3y,'m--','LineWidth',2);
axis(problem3Plot, [-1.5 1.5 -1.5 1.5]);
axis(problem3Plot, 'square');
title(problem3Plot, 'Problem 3 Circle Plot');
```



Published with MATLAB® R2022b