University of Pennsylvania

EAS 105: Fall 2011

MATLAB PROJECT: Gershgorin Circle Theorem

Instructor: Donna Dietz

For this project, you will write some functions which display plots representing the circles in Gershgorin's circle theorem, which estimate where eigenvalues of a matrix may lie. This will be overlaid with points representing the actual eigenvalues, thus giving a visual verification for the theorem for whatever matrix is presented. In order to fully understand this theorem, you will/would need to understand enough linear algegra to understand what an eigenvalue of a complex matrix is. However, in order to correctly do (and enjoy) the project, you don't need to understand this. Matlab can quickly find eigenvalues of matrices, using the *eig* command, and these can be plotted. The statement of the theorem is simple, and its necessary calculations are easy, so finding the centers and radii of the circles is quick and easy to code. For display puropses in Matlab, you should draw the shaded circles first, then draw the points, otherwise you won't see the points!

This should be a fast and easy project, to serve as a break from last week's more lengthy project.

You will get some background on this project in class, but you may also check out Wolfram's website or even Wikipedia for more diagrams and explanations.

One great thing to note about Matlab and plotting, is that Matlab is more than happy to plot a point which is a complex number. For example, $4 + 3i$ will be displayed at as $(4, 3)$. This is already a great time-saver, but on top of this, Matlab allows you to then go ahead and use two coordinates if you wish, on the same plot. That is very nice! You will see why when you begin to work. However, you will still have the need to split up complex numbers, and you can use *real* and *imag* to do this.

To get a plot going, use the command 'figure'. Matlab's default figure behavior is that each time you draw something, you lose the previous thing you had on the plot. To change this behavior, you use the command 'hold on'. To revert back to the old behavior, use 'hold off'.

For this project, create these four functions:

function [CR] = MtoCRlist(M) You may assume M will be a square matrix. The height of the output matrix CR will match the dimensions of M, and the width of CR will be 2. The first column of CR is just the diagonal elements of M. This gives the centers of the circles you need to plot. (You may use the diag command to get that!) The second column of CR, as given by Gershgorin's theorem, is the radii of the circles. To calculate these radii, you take each row of matrix M, disregard the diagonal element, and sum up the absolute values of all remaining elements.

function [circle_out] = PlotCircle(c, r, MyColor) Matlab has a nice built-in function for plotting rectangles which is also used for circles, ellipses, and rounded-off rectangles and squares. Experiment with this command:
c_o=rectangle('position',[x1, x2, 2*r, 2*r],'curvature',[1,1],'FaceColor',MyColor);
Note that MyColor can either be something like 'r' or 'g', or it can be a horizontal vector of length 3, having values between 0 and 1. It is your job to figure out what x1 and x2 should be, based on your input of c. The input c is the center of a given circle (from your CR matrix) and it is a complex (or real) value. Likewise, r is from your CR matrix, and this value is always real. You want to plot a circle centered at c with a radius of r.

function PlotGershgorinCircles(M) This function calls MtoCRlist once and PlotCircle repeatedly. You will want to use the command 'axis equal', or your circles will look like ovals. After plotting all the circles, plot the centers. You can use almost any color scheme you wish. However, I request that you avoid using red, as red is used in the *Tester* to highlight portions of your graph. (Pink is fine.) You need to use the command [v d]=eig(M) to get the eigenvaules of M. (They show up on the diagonal of d.) The command plot(diag(d),'o'); will plot the diagonal elements of the matrix d.

function PlotGershgorinCirclesOneByOne(M) This function is nearly identical to PlotGershgorinCircles, except that each circle/point pair is plotted on a separate figure. The tricky part here is that the sequence of points and the sequence of circles need not be in the same order! Using the sort command on d and CR should fix this problem. (This fix is not guaranteed to always work, but it will work for our purposes. Test your code against the *Tester* to see if it's ok.)

HAVE FUN!!!