CS100M Fall 2007 Project 3 due Thursday 10/4 at 6pm

You must work either on your own or with one partner. You may discuss background issues and general solution strategies with others, but the project you submit must be the work of just you (and your partner). If you work with a partner, you and your partner must first register as a group in CMS and then submit your work as a group.

Objectives

Completing this project will help you learn about nested loops, user-defined functions, and vectors. You will also continue to explore graphics.

1 π via tiling—revisited

Turn script `tilePi` from Project 2 into a *function* for estimating π using tiling. Write a function `piTiling(n,f)` where `n` is an integer radius of the circle to be tiled and `f` is the fraction of a tile assumed for any cut tile. The function *returns* the estimated value of π and does not print any output. You may use the posted solution or your own script from Project 2 as a starting point.

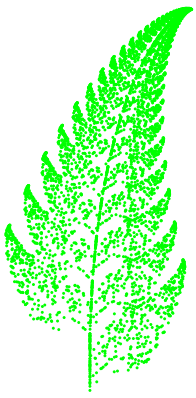
Write a *script* `piTilingError` to approximate π with `n` = 100, 1000, 10000 and plot the error as a function of `f`. (Draw three graphs on the same set of axes, one for each value of `n`.) Use a reasonable number of `f` values. Label the axes and give the figure a title.

If you use a single call to the plot function to draw the three graphs (see, for example, `ShowQuadratic.m` from Lecture 10), you can easily add a legend to the figure:

```
plot(t,u, v,w,':', x,y,'.k')
legend('string1', 'string2', 'string3')
```

The above commands draw three graphs and put a legend in the figure window. The first graph is labeled "`string1`", the second "`string2`", and so on.

2 Barnsley's Fern



Barnsley's Fern, shown above, is a fractal, not a real fern. A fractal is "a rough or fragmented geometric shape that can be subdivided in parts, each of which is a reduced-size copy of the whole."[1]

[1] Mandelbrot, B. B. (1982). *The Fractal Geometry of Nature*. W. H. Freeman and Company

It is the graphical result of a set of iterated calculations. Michael Barnsley "discovered" the set of equations for drawing the above "fern."[2] Starting from an arbitrary point (x, y) and repeatedly choosing one of the following four functions randomly to compute a new point, one will obtain a set of points that resemble a black spleenwort fern:

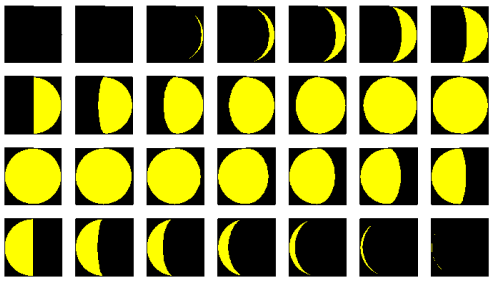
$$\begin{array}{lll} f_1(x,y) = & (0.85x + 0.04y, -0.04x + 0.85y + 1.6) & \text{probability } 0.85 \\ f_2(x,y) = & (-0.15x + 0.28y, 0.26x + 0.24y + 0.44) & \text{probability } 0.07 \\ f_3(x,y) = & (0.2x - 0.26y, 0.23x + 0.22y + 1.6) & \text{probability } 0.07 \\ f_4(x,y) = & (0, 0.16y) & \text{probability } 0.01 \end{array}$$

These are called the *fern functions*. Note that each function's likelihood, or probability, of being chosen in a single iteration varies.

Write a set of five functions to draw Barnsley's fern. Function `f1(x,y)` returns the coordinates (two values) of the next point based on fern function f_1 above. Similarly, write the functions `f2(x,y)`, `f3(x,y)` and `f4(x,y)`. Write a function `fern(n)` to draw Barnsley's Fern using `n` points and your functions `f1` through `f4`. Generate a random starting point in an interval of your choice. In this question you will want to use the graphics window command `axis equal off` to use the same scale in the x- and y-axis but hide the axes.

3 Phases of the Moon

Complete Problem **P4.2.4** on page 13 of *FVL* Chapter 4. Write and submit a function `PaintMoon` and a script `moons`. Example output is shown below.



Hint: In the function draw the full moon (whole disk) on a black background first. Then stack thin rectangles or small squares in black to create the shading! During program development, use a small number of rectangles to keep the computation time short. Again you will want to use the graphics window command `axis equal off`.

[2]Barnsley, M. (1993) *Fractals Everywhere*, 2nd ed. Boston, MA: Academic Press