Project 1: Getting to know Matlab

In the coffee.m example, we created an array of values $x(n)$ satisfying

$$x(n+1) = x(n) - \frac{1}{N} \cdot x(n). \tag{1}$$

In this project we will study the [1]equations

$$x(n+1) = x(n)^2 - y(n)^2 + c, \tag{2}$$
$$y(n+1) = 2x(n)y(n) + d. \tag{3}$$

Unless otherwise specified, let $c = -0.8$ and $d = 0.156$.

a. Modify (or write your own) code to solve the above equations.

b. For specific starting point `x(1)` and `y(1)` (let's say, `x(1)=0.1, y(1)=0.1`), plot the first 22 values of `x(n)` versus `n`.

c. For specific starting point, plot first 22 values `y(n)` versus `x(n)`.

   In the above, the variable `n` was the loop iteration variable (unless you picked another variable name), and this variable went up to `nMax=22`.

d. Write code to create 100 numbers, uniformly randomly selected from the interval (-2,2). Save these in a `1x100` array called `xStart`. Create another 100 uniform random numbers in the interval (-2,2), and store these in an array called `yStart`. Plot all 100 pairs of `xStart` versus `yStart`. This should be a uniformly random spread of dots in a square.

e. Now, in a loop, for each of the 100 pairs `xStart,yStart`, compute the equations for 22 steps (so, just like in part b, but now with a different `x(1), y(1)`). For each of these, check if each `x(22),y(22)` is outside the box (-2,2). If so, plot the corresponding `x(1),y(1)` in red. If not, plot it in blue.

   Hint: You will need another loop iteration variable that goes up to 100.

   Hint: Since you are plotting the initial values, all values — both red and blue — should be in the square (-2,2).

f. Do the same, but now instead of for 100 pairs, do it for `NStartingPoints=1e5`.

g. Change parameter value c and d (to whatever you want) and repeat. Hint: very small changes will give best results.

h. Bonus Part: Make a version that records what n each initial x,y leaves the (-2,2) box. Call this `n_at_exit`. Then, when you plot the points that exist, color the points by the value of `n_at_exit`.

   Hint: the Matlab function `scatter()` is useful for this.

[1] first studied by Gaston Julia, not to be confused with JuliaLang.