MAT 667 (Dynamical Systems)
Homework #6, Due Wednesday, May 3, 2017

For this homework set, let $f_a : [-1,1] \to [-1,1]$ be defined by $f_a(x) = -1+a/2(1-x^2)$. This map is conjugate to the logistic map, and it is *even*.

1 (a) Use a computer to find the a values of the superstable periodic orbits of f_a with period 4.
1(b). Use a computer to find the a value of the period doubling bifurcation of the period 4 orbit of f_a. (The one near $a = 3.5$, not the one near $a = 4$.) The equations are $f_a^4(x) = x$ and $(f_a^4)'(x) = -1$.

You can use FindRoot[ ] defined in Mathematica to do this. If you cannot do 1(b), you can still do the rest of the homework. Part 1(b) is there to show how much easier it is to find the superstable orbits than it is to find the period doubling bifurcations. Feigenbaum computed the sequence of superstable orbits, not the sequence of period doublings, with his programmable calculator. The last problem invites you to reproduce Feigenbaum's experiment.

2. Consider the *shifted geometric series* $b + c + c/\delta + c/\delta^2 + \cdots$. If $b = 0$, this is a geometric series with ratio $1/\delta$. Assume $\delta > 1$ so the series converges. The sequence of partial sums is $s_n = b + \sum_{i=0}^{n} c\,\delta^{-i}$ starting with $s_0 = b + c$ and $s_1 = b + c + c/\delta$. The sum of the series is $s_\infty := b + \dfrac{c}{1 - 1/\delta}$.
Show that $\dfrac{s_n - s_{n-1}}{s_{n+1} - s_n} = \delta$.

3. Suppose that s_n is the sequence of partial sums for a shifted geometric series with an unknown b, c, and δ. Show that

$$s_{n+1} = s_n + \frac{(s_n - s_{n-1})^2}{s_{n-1} - s_{n-2}}$$

and

$$s_\infty = s_{n-2} + \frac{(s_{n-1} - s_{n-2})^2}{2s_{n-1} - s_{n-2} - s_n}$$

for $n \geq 2$. Thus, if you know three partial sums in a row, then you can compute the next partial sum and the limit.

4. Let a_k be the smallest a value which has a superstable period k orbit in the logistic map. We know from pencil and paper that $a_1 = 2$ and $a_2 = 1 + \sqrt{5}$. In problem 1(a), you found numerically that $a_4 = 3.49856....$ Problem 2 shows that a_{2^n} is appproximately s_n, the partial sum for some shifted geometric series. Make a prediction for a_8 by assuming the formulas found in problem 3 are exact for $s_n = a_{2^n}$. Then, find a_8 numerically using your prediction as the first guess. If you use Mathematica, you can use the FindRoot command.

5. For the function $f_a : [-1,1] \to [-1,1]$ defined by $f_a(x) = -1 + \frac{a}{2}(1 - |x|)$ for $a \in [0,4]$. For what values of a is f_a in the domain D of the peroid doubling operator $\mathcal{T}$? (Actually, $f_a''(0)$ is not defined, so just check the other conditions.) Compute $\mathcal{T} f_a$ for f_a in the domain of $\mathcal{T}$.
Hint: You only need to consider $\mathcal{T} f_a(x)$ for $x \geq 0$. Then you can replace x with $|x|$ since $\mathcal{T} f_a$ is even.

6. (Extra Credit) Reproduce what Feigenbaum did with his programmable calculator. Write a computer program that uses Newton's method to find a_{2^n} for $n = 3, 4, \cdots, 11$, given the known values of a_1, a_2, and a_4. As we know, Newton's method needs a good guess to work well, and you now have the knowledge to make an excelent guess. Print out a table of a_{2^n}, and the estimates of δ and a_∞ based on a_{2^n}, $a_{2^{n-1}}$ and $a_{2^{n-2}}$. You will use the estimate of $a_{2^{n+1}}$ to compute the next row in the table, but this does not need to be in the table.

Hints:
The table will look like this:

2^n	a_{2^n}	estimate of δ	estimate of a_∞
4	3.498561699327701	4.708943013540513	3.5693348875027704
8	answer to 4(b)		
$\vdots$			
2048		$4.669\cdots$	

For maximum extra credit, do not use the FindRoot command in Mathematica. Instead, write your own Newton's method loop. You need to write a subroutine to numerically compute $f_a^k(0)$ and $\frac{d}{da} f_a^k(0)$ to implement Newton's method. If you can't get Newton's method to work you can use FindRoot for half the extra credit, but FindRoot will choke on period 2^{11} (in 2017 on Swift's pretty new iMac).

You do not need to use Mathematica. Any computer language will do.