Homework 2

Problem 1: The Bisection Method

Consider the function

$$f(x) = x^5 - 2x^4 + 3x^3 - 2x^2 + x - 1 - \cos(30x).$$

Calculate $f(0)$ and $f(1.2)$ and save their product as `A1.dat`. Does this tell you if f has a zero on the interval $[0, 1.2]$? Does it tell you how many zeros f has on the interval $[0, 1.2]$.

Use the bisection method to find a zero of f between $x = 0$ and $x = 1.2$. Your final answer should be a number x such that $-10^{-10} < f(x) < 10^{-10}$ (i.e., use a tolerance of 10^{-10}). Save the resulting x in `A2.dat`. Save the number of steps this process took in `A3.dat`. (Each time you calculate a midpoint counts as one step.)

Problem 2: Equilibria

We will now continue to investigate the logistic map that we first encountered in the previous homework:

$$P(t+1) = rP(t)\left(1 - \frac{P(t)}{K}\right),$$

where $P(t)$ represented the population density of a species after t years. Throughout this problem, we will assume that $P(0) = 10$ and $K = 50$. (Note that these are not the same as in the previous homework. In particular, notice that time starts at year 0 now, not year 1.)

If $r = 2.5$, create a vector containing the population densities for years 0 through 100. That is, the first entry of your vector should be $P(0)$, the second entry should be $P(1)$, etc., and the last entry should be $P(100)$. If you plot these densities, you should see that the population density approaches a single value and stays there. (If you called your vector `x`, you could use the command `plot(x)` to visualize this data. Make sure you remove all plots before submitting to scorelator.) This final value is called an *equilibrium* or *steady state* of the logistic map.

Repeat this process for $r = 3.2$ and $r = 3.5$. You should see very different behavior in your plot. In particular, these densities should not approach an equilibrium.

Make a matrix

$$A = \begin{pmatrix} P_{2.5}(98) & P_{2.5}(99) & P_{2.5}(100) \\ P_{3.2}(98) & P_{3.2}(99) & P_{3.2}(100) \\ P_{3.5}(98) & P_{3.5}(99) & P_{3.5}(100) \end{pmatrix},$$

where $P_r(t)$ is the population density P at time t using the parameter r. (E.g., the first row of this matrix should be the last three entries of the vector you obtained when $r = 2.5$.) Save this matrix as `A4.dat`.

We would like to determine which values of r lead to an equilibrium. As a somewhat crude method, we can check if the population densities stop changing (or at least do not change very much) after a long enough wait. We will say that P_r reaches an equilibrium for some value of r if $P_r(501) - P_r(500)$ is sufficiently small. In particular, we will say P reaches an equilibrium if $-10^{-8} < P_r(501) - P_r(500) < 10^{-8}$ and we will say that the equilibrium value is $P(501)$.

For each value of r between 2 and 3.4 in steps of 0.1 (i.e., for every r in `r = 2:0.1:3.4`), decide if the population density reaches an equilibrium and, if it does, find the equilibrium value. Create a 1×15 vector `is_equilibrium` whose entries are either 1 or 0. The kth entry of `is_equilibrium` should be one if and only if P_r reaches an equilibrium for the kth value of r (i.e., `r(k)` if you defined `r = 2:0.1:3.4`). Save `is_equilibrium` in the file `A5.dat`.

In addition, create a 1×15 vector `equilibrium_value` whose entries are the equilibrium value of P_r if such a value exists and zero otherwise. Save this vector in the file `A6.dat`.

As an example, if $r = 2.1$ reached an equilibrium value of 20, then `is_equilbrium(2)` should equal 1 and `equilibrium_value(2)` should equal 20.

Things to think about (not required): For some values of r, the population density appears to oscillate between two values. Can you test for those values? Can you find oscillations between three different values? Four? Can you find a better method to test for equilibria? In particular, do you think there is an equilibrium when $r = 3$?

Finally, it seems clear that there is no equilibrium for $r = 4$, and some experimentation will show that the same is true for a range of r values below four. In a previous quarter, I made the mistake of asking people to calculate P for these values of r and scorelator marked many people wrong even though their code was fine. Why do you think that happened? (This full explanation is fairly challenging.)

Problem 3: Protein Configurations

Proteins are long chains of amino acids that can fold into different configurations (shapes). The configuration of the protein determines (at least in large part) how the protein works. Suppose that we have a particular protein which can be folded into four different configurations: c_1, c_2, c_3 and c_4, but that only has a useful function when it is in configuration c_1. We can select a protein in the correct configuration, but because of random molecular motion we cannot guarantee that it will stay in this configuration forever. This means that we can only find the probability that the protein will be in any given configuration at any given time. Define the vector

$$\mathbf{p}(t) = \begin{bmatrix} p_1(t) & p_2(t) & p_3(t) & p_4(t) \end{bmatrix},$$

where $p_i(t)$ is the probability that the protein will be in configuration c_i after t hours. If we choose a protein that we know is in configuration c_1 at time 0, find $\mathbf{p}(0)$ and save it in the file `A7.dat`. (Remember that the total probability of all four configurations must add up to 1.)

We can summarize the dynamics of this protein with something called a transition probability matrix P. In our case,

$$P = \begin{pmatrix} 0.99 & 0.00 & 0.01 & 0.00 \\ 0.00 & 0.56 & 0.25 & 0.19 \\ 0.10 & 0.22 & 0.42 & 0.26 \\ 0.00 & 0.36 & 0.15 & 0.49 \end{pmatrix}.$$

The entry P_{ij} (i.e., the number in the ith row and jth column) gives the probability that the protein will switch to configuration c_i in the next hour if it is already in configuration c_j. For example, if the protein is already in configuration c_3 then there is a 1% chance that it will switch to c_1 after one more hour, because $P_{13} = 0.01$. This matrix allows us to calculate future probability distributions very easily. We have

$$\mathbf{p}(t+1) = \mathbf{p}(t)P.$$

We will say that the protein is ineffective once it has less than an 80% chance of being in configuration c_1. How many hours will it take until the protein is ineffective? (That is, find the first time that $p_1(t)$ drops below 0.80. Remember that we started at time 0.) Save this time in `A8.dat` and save the corresponding vector $\mathbf{p}(t)$ in `A9.dat`.

If you calculate $\mathbf{p}(t)$ for sufficiently large times, you should see that it approaches a constant vector. Just as in the previous problem, this vector is called the *equilibrium value* of $\mathbf{p}$. Find this equilibrium value and save it in the file `A10.dat`.

More things to think about: Probabilities have to add up to 1. In particular, this means that at any time, $p_1(t) + p_2(t) + p_3(t) + p_4(t) = 1$. You need to enforce

this for $t = 0$, but for all later times we didn't check this. Does it appear to be true for all t? What does this have to do with the matrix P? (There is a fairly simple property of P that forces this to be true.) Does the equilibrium value depend on your initial condition? That is, can you come up with a different $\mathbf{p}(0)$ that results in a different equilibrium value? Does the equilibrium value depend on the matrix P? Can you come up with a matrix so that the equilibrium value does depend on the initial condition? (Remember to keep whatever condition you found so that the sum of the probabilities is always one.)