Due Wednesday, 9/26/2018

Scientific Computing

Assignment 3: Root-Finding with an Energy Balance Climate Model

In this assignment, we will build two programs which find roots of a given input function -- that is, they find an *x* such that *f*(*x*) = 0.

Warm-up exercise:

Write a MATLAB .m-file which defines a simple function:

quartic(*x*) = $x^4 + x - 2$

It's almost impossible to find where *quartic*(*x*) = 0 by using algebra – go ahead and try it. But it's easy to confirm that *x* = 1 is a solution.

Question 1: Use MATLAB to plot a graph of *quartic*(*x*) between -2 < x < 2. Use the graph to estimate the zero-crossing points by eye.

Root-finding Algorithms

In class, we covered two different root-finding algorithms: the bisection method and the secant method. Here's a review of the algorithms:

The Bisection Method

1) Goal: find x to satisfy $f(x) = 0$.

2) Pick a point x_0. Find $f_0 = f(x_0)$.

3) Pick another point x_1. Find $f_1 = f(x_1)$.

4) Make sure that f_0 and f_1 have opposite signs, or the algorithm will fail!

5) Let $x_2 = (x_0 + x_1)/2$. Find $f_2 = f(x_2)$.

6) If f_1 and f_2 have the same sign, replace x_1, setting $x_1 = x_2$.

7) Otherwise, replace x_0, setting $x_0 = x_2$.

8) If f_2 is close enough to zero, stop. Otherwise, go to step 5.

The Secant Method

1) Goal: find x to satisfy $f(x) = 0$.

2) Pick a point x_0. Find $f_0 = f(x_0)$.

3) Pick another point x_1. Find $f_1 = f(x_1)$.

4) Draw a straight line between (x_0, f_0) and (x_1, f_1), and find the point where that line crosses the x axis. We showed in class that this was equal to:

$$x_2 = \frac{x_0 f_1 - x_1 f_0}{f_1 - f_0}$$

5) Set $x_0 = x_1$, $f_0 = f_1$.

6) Set $x_1 = x_2$, $f_1 = f(x_2)$.

7) If f_1 is close enough to zero, stop. Otherwise, go to step 4.

Write two .m-files which implement these algorithms, and use them to demonstrate that they find the roots of *quartic* (*x*).

Question 2: What are *all* the real roots of $x^4 + x - 2 = 0$?

<u>If you've taken a computer science class or two,</u> you should create a general root-finding function that takes a function as input, and returns a root. To evaluate arbitrary functions in MATLAB, you use the 'feval' function, as follows:

```
funcname = 'quartic'
feval(funcname, 0)
```

runs the function named 'quartic' with the input 0, which is to say quartic(0), which equals -2. You can also create a "function pointer" using the @ command:

```
f = @quartic;
```

Then f(0) returns quartic(0) which is -2. If you're new to programming, just put everything in a single MATLAB script: your code will be harder to use, but easier to write.

Once you've created your root-finding functions, it's time to use them in a real-world application:

Energy Balance Climate Model

Here's a simple model for the energy budget of the Earth, which balances solar energy input against infrared radiation output:

Net energy flow: Power in - Power out = 0 (1)

Let's consider "Power In" first. How much solar energy does the Earth intercept? Let S_0 be the intensity of sunlight at Earth, in W/m^2. Then

Incoming solar = $S_0(\text{cross-sectional area of Earth}) = S_0 \pi R^2$

Now, not all of that energy is absorbed by the Earth: some of the sunlight is reflected away. The fraction of reflected light is the earth's albedo, designated α, so the fraction absorbed is (1-α):

Power in = (1-α) • (Incoming solar) = $(1-\alpha) S_0 \pi R^2$

Now let's think about "Power out". The Stefan-Boltzmann blackbody law gives the amount of energy radiated by a body of temperature *T* (units of *T* are Kelvin):

Power out per unit area = σT^4

Now, the Earth has a greenhouse effect which traps infrared energy and prevents it from escaping easily. We can include a greenhouse effect factor γ like this:

Power out per unit area, including greenhouse effect = $\gamma\sigma T^4$

For the Earth, $\gamma \approx 0.6$. Thus, the total power out over the whole globe is:

Power out = (surface area of Earth) • $\gamma\sigma T^4$ = $4\pi R^2 \gamma\sigma T^4$.

In general, a stable climate can only be achieved when (1) is satisfied: since net energy flow depends on temperature, we want to find *T* such that net_energy_flow(*T*) = 0 – this should give us the average temperature of the Earth.

Constant	Value
Solar intensity S_0	1367 W/m^2
Stefan-Boltzmann const σ	5.67e-8 $W/(m^2\text{-}K)$
Earth radius *R*	6.3e6 m
Greenhouse constant γ	0.6

I'll ask you to consider two forms of this equation. In the first, we assume the Earth's albedo is constant, and find a value for *T* which satisfies (1). We'll solve this by hand as well as numerically (using your secant and bifurcation techniques), and compare the answers. In the second part of this assignment, we'll consider that the Earth's albedo depends on its temperature, and look at the consequences numerically.

Planetary Temperature: Fixed-Albedo Model

The Earth's average albedo α is about 0.3, meaning that 30% of incoming sunlight is reflected and 70% is absorbed.

Question 3: Using plain old algebra, find the average temperature of the Earth predicted by this model by finding the value of *T* which satisfies (1), assuming α = 0.3. Does your result seem high, low, or roughly accurate? (Remember, you will find T in Kelvin! Convert it to Celsius or Fahrenheit.)

Question 4: Use the bisection method to find a *T* which satisfies (1). Keep your code well-organized! You should write a separate function to compute the net energy flow (the left-hand-side of (1)) as a function of temperature: your bisection method code should call this function. That is, you want one .m-file that handles the model physics, and a second one that handles the bisection-method stuff.

Question 5: Now, solve the same problem using the secant method. It should give the same answer. Now you see why well-organized code using MATLAB functions is valuable: you can re-use the functions and scripts you built earlier, and do this task much faster!

Planetary Temperature: Variable Albedo

Earth's albedo depends on its temperature. If the Earth were very cold, it would be covered by snow and ice, which are white and reflect sunlight. If it were very warm, it would be ice-free.

On OnCourse, you'll find a file named "albedo.m". This is a function I wrote which roughly describes how the Earth's albedo depends on its average temperature. Copy it into your working directory.

Question 6: Create a graph which shows how the output of my albedo function depends on temperature, for T ranging from 200 to 330 K. Does it do what I said it should do?

With this added complication, it becomes impossible to use algebra to solve for Earth's temperature. But you can still use the secant or bifurcation methods to find a solution.

Question 7: Modify your code to make use of this variable albedo, instead of the constant value we used earlier. Then plot the net energy flow (the left-hand side of (1)) as a function of temperature, between 200 and 330 K. You should see the plot cross the x-axis more than once: this model has *multiple solutions*! How many solutions do you see?

Only the warmest and coldest of these solutions are stable, and could actually occur. The middle one(s) satisfy equation (1), but even the slightest change in temperature will cause the model's state to drift away: they are unstable.

Question 8: Use either the secant or bisection method to find all possible states of this climate model: that is, find all values of T which satisfy (1). To do this, you'll need to choose initial values for x_0 and x_1 which are in the ballpark of the solution you're looking for. What is the temperature and albedo of the planet in its warmest and coldest possible climate states?

Question 9: If the Earth were a little farther from the sun, so that S_0 was reduced to 1000 W/m^2, would there still be multiple stable climate states? What if the Earth was a little closer, so that S_0 increased to 2500 W/m^2?

Challenge Question 10: At what values of S0 does the system switch from having multiple solutions to having only one?

These transition values are called "bifurcation points". I'll show you why next week.