BME695 Numerical Methods in BME

Warm-up Coding Assignment II

Feb 1 2021

Please review these notes before you start!

- Each person will get 30 minutes to discuss the warm-up coding assignment in a one-on-one meeting on either Tuesday (10 a – noon on 2/9) or Thursday (10 a – noon on 2/11).
- Please sign up for a 30-minute time slot on the google Excel spreadsheet "Warm-up Coding Assignment Discussion" I have created.
- I have placed a number of documents in the supplemental materials folder under Warm-up Coding Assignment on Google drive.
- As you can imagine, you can easily find Matlab codes for some of the numerical methods in the following. I ask you to study and implement. Please DO NOT look at them unless you absolutely have to!

1 Solving a Nonlinear Single Equation

In this exercise, you will implement the bisection method and the Newton's method for solving a nonlinear single equation. The specific tasks are in Subsections 1.2, 1.4, and 1.5.

1.1 Bisection method

One of the first numerical methods developed to find the root of a nonlinear equation was the bisection method (also called binary-search method). The method is based on the following theorem.

Theorem 1 *An equation $f(x) = 0$, where $f(x)$ is a real continuous function that has at least one root between x_l and x_u if $f(x_l)f(x_u) < 0$ (see Figure 1).*

Note that the above theorem only guarantees one root between x_l and x_u. Following the theorem, an algorithm for **Bisection method** can be stated as follows.

The steps to apply the bisection method to find the root of the equation $f(x) = 0$ are:

1. Choose x_l and x_u as two guesses for the root such that $f(x_l)f(x_u) < 0$, or in other words, $f(x)$ changes sign between x_l and x_u.

2. Estimate the root, x_m, of the equation $f(x) = 0$ as the mid-point between x_l and x_u as
$$x_m = \frac{x_l + x_u}{2}.$$

3. Now check the following
a) If $f(x_l)f(x_m) < 0$, then the root lies between x_l and x_m; then $x_l = x_l$ and $x_u = x_m$.
b) If $f(x_c)f(x_m) > 0$, then the root lies between x_m and x_u; then $x_l = x_m$ and $x_u = x_u$.
c) If $f(x_l)f(x_m) = 0$, then the root is x_m. Stop the algorithm if this is true.

4. Find the new estimate of the root
$$x_m = \frac{x_l + x_u}{2}.$$
Find the absolute relative approximate error as
$$\epsilon_b = \left| \frac{x_m^{new} - x_m^{old}}{x_m^{new}} \right|,$$
where x_m^{new} = estimated root from present iteration; x_m^{old} = estimated root from previous iteration.

5. Compare the absolute relative approximate error ϵ_b with the pre-specified relative error tolerance ϵ. If $\epsilon_b > \epsilon$, then go to Step 3, else stop the algorithm.

Note that perhaps one also wants to check whether the number of iterations is more than the maximum number of iterations allowed. If so, one should also stop the algorithm for this alternative termination condition.

I have placed a pdf document on **Bisection Method** in the supplemental materials folder.

1.2 Task list 1-A

- Before you review the pdf document on the method, please conduct the following thought exercises. For the given function $f(x)$ and the initial root-finding range $[x_l, x_u]$,

    1. if $f(x_l)f(x_u) > 0$, there may or may not be any root between x_l and x_u? Please provide graphic illustrations for different cases (i.e., different numbers of roots).

2. if $f(x_l)f(x_u) > 0$, there may be more than one root between x_l and x_u. Please provide graphic illustrations for different cases (i.e., different numbers of roots).

- Now you can review the pdf note, can you provide flowchart to illustrate the algorithm?
- Now you get to implement the algorithm. Please set ϵ, x_l, and x_u as input arguments of your function. You may set $f(x)$ as an input argument as well. Please go through your code line by line when we meet.
- You are working for a company that makes floats for ABC commodes. The floating ball has a specific gravity of 0.6 and has a radius of 5.5 cm. You are asked to find the depth to which the ball is submerged when floating in water. Thus, the equation that gives the depth x to which the ball is submerged under water is given by $x^3 - 0.165x^2 + 3.993 \times 10^{-4} = 0$.
  Use the bisection method to find the depth x to which the ball is submerged under water.
- Find the positive minimum point of the function $f(x) = x^{-2}\tan x$ by computing the zeros of f'. Set the lower and upper bounds of the initial guess are 0 and 2, respectively.

1.3 Newton's method

In class, we talked about the Newton's method. Here to remind you, Newton's method is used to find the solution of $f'(x) = 0$ as follows:

$$x_{k+1} = x_k - \frac{f'(x)}{f''(x)}.$$

Let us consider the most commonly used stopping criterion/condition for the algorithm is that the relative change at some iteration is lower than the prespecified relative error tolerance ϵ, i.e., if

$$\epsilon_n(x_k) = \left| \frac{x_{k+1} - x_k}{x_{k+1}} \right| = \left| \frac{f'(x_k)}{f''(x_k)x_{k+1}} \right| > \epsilon \ ,$$

then continue with the next iteration, else stop the algorithm.

I have placed a pdf document on **Newton's Method** in the supplemental materials folder.

1.4 Task List I-B

- Please provide a flowchart to illustrate this iterative algorithm.
- Now you get to implement the algorithm. Please set ϵ and the initial guess x_0 as input arguments of your function. You may set $f(x)$ as an input argument as well. Please go through your code line by line when we meet.

- Bring back the floating ball problem (i.e., the root-finding problem) in Section 1.2. Please use the Newton's method to solve it. Output a table containing information on iteration index, x_k, and $\epsilon_n(x_k)$. Set a sufficiently small threshold ϵ and verify the convergence rate.

- Bring back the minimization problem in Setion 1.2. Please use the Newton's method to solve it. Again output a table containing information on iteration index, x_k, and $\epsilon_n(x_k)$. Again check out the convergence rate.

1.5 Task List I-C

By now, you have tested two root-finding or nonlinear optimization algorithms. Next let us compare the two methods in terms of wall-clock running time. Can you please some online tutorial to justify your observations? You may want to test the two algorithms on additional nonlinear single equations.

2 Runge-Kutta Methods

The Runge–Kutta methods are a family of implicit and explicit iterative methods, which include the well-known Euler Method, used in temporal discretization for the approximate solutions of ordinary differential equations. These methods were developed around 1900 by the German mathematicians Carl Runge and Wilhelm Kutta.

Here, we attempt to implement three commonly used forms. We provide a brief description of the Euler method, which is of the simplest form within the family of the Runge-Kutta methods.

Let us consider the following ODE,

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0.$$

Choose a value h for the size of every step and set $t_n = t_0 + nh$. Now, one step of the Euler method from t_n to $t_{n+1} = t_n + h$ is:

$$y_{n+1} = y_n + hf(t_n, y_n).$$

Then the value of y_n is an approximation of the solution to the ODE at time $t_n : y_n \approx y(t_n)$.

2.1 Your Tasks

- Implement the Euler method by yourself. Go through the code line by line when we meet.

- Look for a description of the second order Runge-Kutta method (RK2) by yourself. For example, like the pdf note I've placed in the supplemental materials folder. Implement the method. Go through the code line by line when we meet.

- Look for a description of the fourth order Runge-Kutta method (RK4). Implement the method. Go through the code line by line when we meet.
- Now you get to identify the corresponding Matlab routine(s).
- Use the above three methods to approximate the following ODE, $y' = -y, y(0) = 1$. Set the step size of h = 0.1 to find approximate values of the solution t = 0.1 to 5 with increment of 0.1. Can you solve this ODE analytically? If so, please evaluate the closed-form expression of the true solution at t = 0.1 to 5 with increment of 0.1. Comment on the comparative results (i.e., with the three numerical methods).
- Use the above three methods to approximate the other ODE, $y' + 2y = 2 - e^{-4t}, y(0) = 1$. Set the step size of h = 0.1 to find approximate values of the solution t = 0.1, 0.2, 0.3, 0.4, and 0.5. Compare the three approximate solutions and compare them with the results via the Matlab routine(s).

3 The Nelder-Mead Optimization Algorithm

Read about the Nelder-Mead optimization algorithm and identify the corresponding Matlab routine. Think about the following questions:

1. What types of optimization problems can be solved by the Nelder-Mead algorithm?
2. Consider the 2-D function $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$. Find its minimizer by setting the start point to $x_0 = [-1.2, 1]$.