CS100 B
Programming Assignment 2

Due: Tuesday, September 21

1. Goals

This assignment will help you develop skills in using conditional and repetition statements. You will also compare and contrast different solution approaches with Java and MATLAB.

2. Motivation

Landscaping and construction often require structures to shore up and block portions of the ground from collapsing. Very often, such structures called temporary *piles* help protect workers from being crushed on construction sites. Figure 1 illustrates the cross-section of a *sheet pile* driven into sandy soil. The weight of the ground creates a force P that pushes the pile to left and causes different pressure distributions to the left and right of the pile. Balancing these pressure distributions according to the soil properties yields the proper depth one should drive the pile into soil.

Given unit weight γ (force/volume), find the necessary pile depth D from the following equation:

$$D^4 - \frac{8P}{\gamma(K_p - K_a)}D^2 - \frac{12PL}{\gamma(K_p - K_a)}D - \left(\frac{2P}{\gamma(K_p - K_a)}\right)^2 = 0. \qquad \textbf{(1)}$$

The coefficients

$$K_a = \tan^2\left(45° - \frac{\phi}{2}\right) \qquad \textbf{(2)}$$

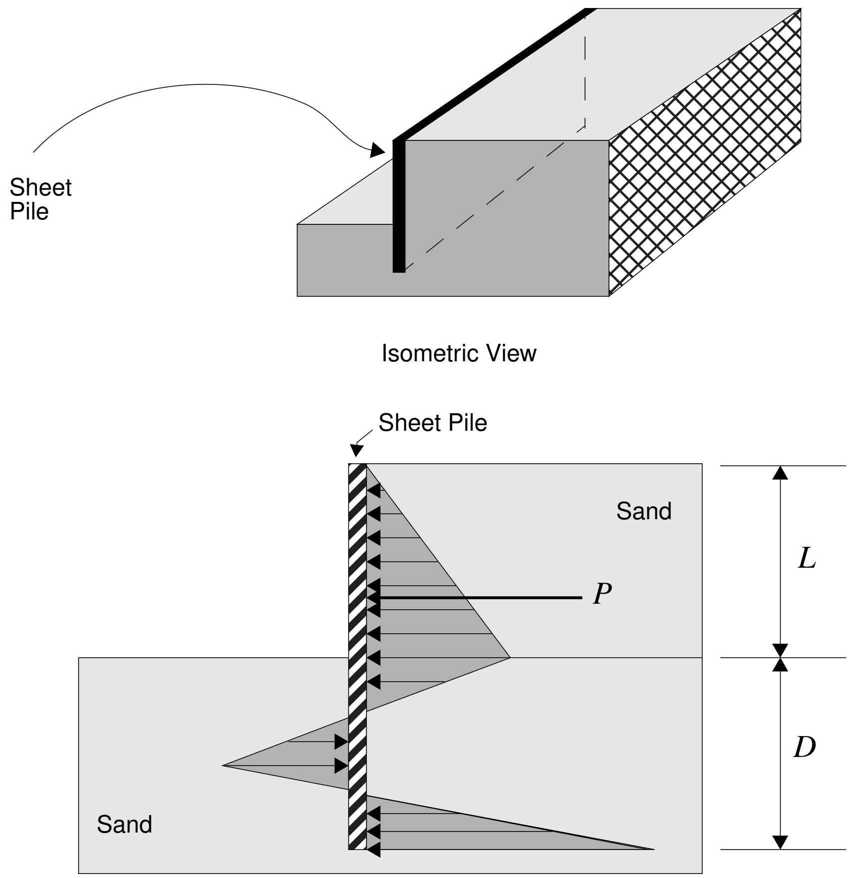
and

$$K_p = \tan^2\left(45° + \frac{\phi}{2}\right) \qquad \textbf{(3)}$$

help determine the pressure the soil places on the pile. The angle ϕ represents the soil's friction.

3. Mathematics

A geotechnical engineer would need to solve Eq. 1 for the value of D to determine pile depth. In this equation, substituting the correct value of D would cause the left-hand side to become zero.



Cross-Section

Figure 1: Sheet Pile

A value that will satisfy an equation is called a ***root***. For instance, solving Eq. 1 for D means find the root of Eq. 1. How does one solve a polynomial for roots?

3.1 Equality and Round-off Error

Before solving for roots, consider the following issue, faced in all numerical computations. When using floating-point numbers, computers rarely compute an exact value of an expression. Try computing **`1.0/3.0`**, for example. Usually, an arithmetic operation introduces a small amount of round-off error at a “distant” decimal place. Though minuscule, this error will typically cause failure of an equality comparison. When comparing two floating-point numbers, use a ***tolerance***, a small numerical value below which you “don’t mind” a difference. Many texts refer to tolerance as ε. The following equation compares two quantities x and y for approximate equality,

$$|x - y| \le \varepsilon . \qquad \textbf{(4)}$$

For instance, the numbers 1.0001 and 1.0002 are essentially “equal” within a tolerance of 0.0001. Refer to Lewis&Loftus, pp. 99-100, for further details.

3.2 LHS/RHS Method

The left-hand side/right-hand side (***LHS/RHS***) technique, as denoted by Dr. Schwartz, is a fancy expression for "controlled guessing." This brute force approach applies this algorithm:

1. Pick a starting value, like **`0`**. Call this value **`root`**.
2. Pick a step size, **`inc`**, for iterating the starting value.
3. Pick a value for **`TOLERANCE`**, as discussed in Section 3.1.
4. Substitute **`root`** into your equation.
5. Assign the result to **`lhs`**. Note that your target **`rhs`** is zero, of course.
6. Test if **`lhs`** and **`rhs`** are approximately equal:
   - If equal, then stop. You have a valid root.
   - If not equal, then increment **`root`** by **`inc`** and test the new value.

Of course, this technique leaves much to be desired, especially if your desired root might be negative. The next section presents a more refined technique.

3.3 Bisection Method

A still-somewhat-brute-force-but-not-as-brutal technique involves a bit more strategy. What happens to a function $f(x)$ when the independent variable x becomes a root? The function either touches or crosses the independent variable's axis. For a simple case, consider the line $y = x + 1$. When $x = -1$, $y = 0$:

- Try a value $x > -1$. To the "right" of $x = -1$, y is positive.
- Try a value $x < -1$. To the "left" of $x = -1$, y is negative.

Thus, tracking where a function changes sign helps locate a root. The ***bisection method*** searches for a root by investigating sign changes within intervals using the following algorithm:

1. Pick a starting point, **`left`**, and an end point, **`right`**. The interval between **`left`** and **`right`** must contain the root! Note, also, that the following steps repeat inside an iteration loop until a tolerance is satisfied, as discussed Section 3.1.
2. Assign **`(left + right)/2`** to **`mid`**.
3. Calculate the value of the function for the points **`left`**, **`right`**, and **`mid`**.
4. Assign the results to **`f_left`**, **`f_right`**, and **`f_mid`**, respectively.

   Review Figure 2:
   - If x_m is to the left of the root, $f(x_m)f(x_R) < 0$.
   - If x_m is to the right of the root, $f(x_m)f(x_L) < 0$.

   The root must lie within the interval where the product, $f(x_m)f(x_R)$ or $f(x_m)f(x_L)$, is negative, as shown in Figure 2.
5. Compute the products **`f_mid*f_left`** and **`f_mid*f_right`**.
   - If **`f_mid*f_right < 0`**, then **`root`** exists between **`mid`** and **`right`**.
     - Assign **`f_mid`** to **`f_left`**, i.e., **`left=mid`**.

- Repeat the analysis for this new interval.

- Otherwise, if **`f_mid*f_left < 0`**, then **`root`** exists between **`left`** and **`mid`**.
  - Assign **`mid`** to **`right`**, i.e., **`right=mid`**.
  - Repeat the analysis for this new interval.

6. Stop iterating when $|f(x_m) - 0|$ no longer exceeds **`TOLERANCE`**.

The bisection method also lacks robustness. For instance, an initial interval containing the solution must be supplied. But, where does that interval come from? An analyst must plot the function first to obtain a rough idea of a root's location before attempting more rigorous analysis.

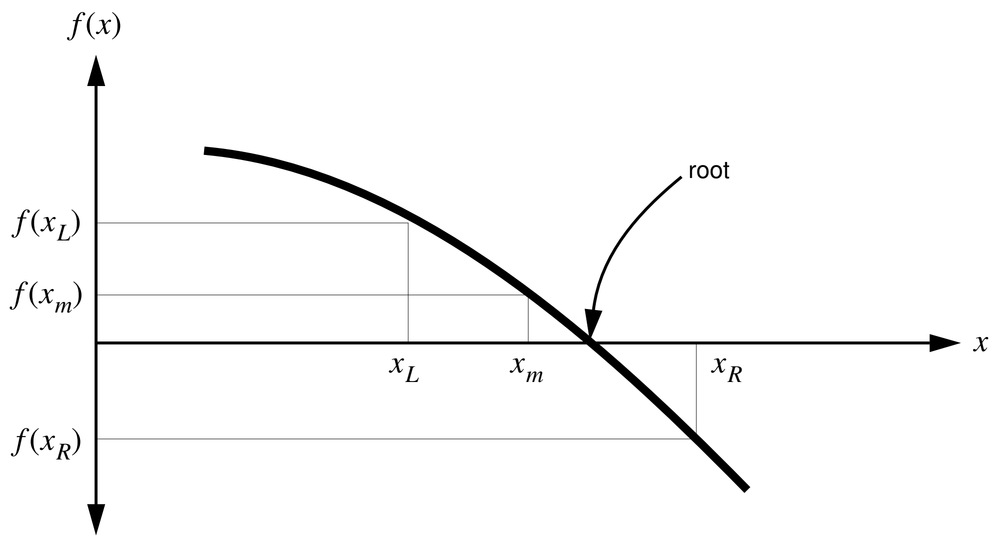


Figure 2: Bisection Method

3.4 Newton's Method

A more efficient method adapts the notion of a Taylor series. Given a point x_0 "close" to a root of $f(x)$, expand $f(x)$ as follows:

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2!}f''(x_0) + \dots \tag{5}$$

Setting Eq. 5 to zero implies that x is a root of $f(x)$. Leaving out all higher order derivatives gives an approximate equation:

$$0 \approx f(x_0) + (x - x_0)f'(x_0). \tag{6}$$

Rearranging terms yields

$$\delta = x - x_0 = -\frac{f(x_0)}{f'(x_0)}. \tag{7}$$

Starting with a value of $x = x_0$, Eq. 7 allows for iteration according to this algorithm:

1. Pick an initial value **`x0`**.
2. Assign **`x0`** to **`x`**.
3. Compute the value of $f(x)$. Assign the result to **`f_of_x`**.
4. Compute the value of $f'(x)$. Assign the result to **`deriv`**.
5. Compute **`delta = -f_of_x/deriv`**.
6. If **`delta`** does not exceed **`TOLERANCE`**, then stop.
7. Otherwise, assign **`x+delta`** to **`x`** and iterate, starting from Step 3.

In general, Newton's Method converges very rapidly and efficiently.

4. Problems

Given the data for Eq. 1, $\gamma = 18$ kN/m^3, $\phi = 30°$, $L = 3$ m, and $P = 30$ kN/m, you need to determine the necessary depth at which to drive the sheet pile, as shown in Figure 1. Assume an error tolerance of $\varepsilon = 0.001$. Create, test, and run three different programs that implement the following numerical techniques:

1. LHS/RHS Method, as discussed in Section 3.2. Use a starting value of 3.
2. Bisection Method, as discussed in Section 3.3. Use an initial interval of $[2, 4]$.
3. Newton's Method, as discussed in Section 3.4. Use a starting value of 3.

Hints: You will need to use **`Math.pow`** for exponentiation. Also, beware that Java requires degrees in terms of radians. To compute the derivative for Newton's Method, note that Eq. 1 is just a fourth-order polynomial.

4. For comparison, plot the equation for the supplied data in MATLAB. Use a range of D values from 1 to 5 at increments of 0.5 m. Provide a plot title that includes your name(s) and section(s). Hints: Write a small program that prints out the values of the polynomial for different pile depths. Then, use MATLAB to plot the list of points as **`plot([x1 x2 x3...], [y1 y2 y3...])`**, where D corresponds to the x-axis.
5. Answer the following questions (except for 5d, which is optional):

   a) How closely do the results match from the programming approaches?

   b) Does your MATLAB plot show a root that corresponds to your programs' results? Why or why not?

   c) How could you modify your code to demonstrate the differences in efficiency of the three programming approaches? (Hint: Think of something simple.)

   d) (This question is optional!) How might you use your coding and MATLAB to determine how increasing the soil friction might effect the required pile depth?

5. Submitting Your Work

5.1 Due Date

This assignment is due in lecture on Tuesday, September 21, 1999. You may turn it in to a consultant before that date in the consulting room in Carpenter. Do not turn it in at Carpenter on the due date. Programs will not be accepted late.

5.2 Labeling Your Work

Always write your name, Cornell ID#, and the day/time/instructor for your section in the first comment of each program you hand in for credit. Otherwise, the program will not be graded. All solutions and commentary must be typed! If you wrote the program with a partner, turn in only one printout with your partner's name and ID# in the comment, as well as your own. The comment must also include the section day/time/instructor for the partner. The program will be returned to the first person listed. Sign your name(s) in the comment. Please staple the pages of your assignment together.

5.3 Grading

This assignment will be given two grades: the first based on correctness, the second on program organization and style. Each grade will be a 0, 1, or 2. Not only should your program work, but it should contain adequate comments to guide the reader who is interested in understanding it. The declaration of every significant variable should include a comment describing that variable. There should be appropriate comments in the code so the reader can see the structure of the program, but not so many that the program text is hard to read.

5.4 What to Hand In

Read Section 4 (Programs) of the *CS100 General Information* packet for explanations of requirements. For this assignment, staple the following sheets together:

- LHS/RHS program listing. (Problem 1)
- Bisection Method program listing. (Problem 2)
- Newton's Method program listing. (Problem 3)
- Printout of MATLAB plot. (Problem 4)
- Typed solutions for Problems 5a, 5b, and 5c.