ME 2450 Lab 3: Root Finding - Closed Methods

Objective

Write a computer program that calculates head loss in a section of a pipe. A partial implementation of `calculate_head_loss` is provided. The lab should be completed in either matlab or python. To receive credit for the lab, one must:

- ☐ (3points) write a function `colebrook_equation` that correctly implements the Colebrook equation as described by eq (2).
- ☐ (5points) write a function `bisection` that correctly finds the roots of a nonlinear equation.
- ☐ (1points) run the `calculate_head_loss` program and report to your TA the resultant head loss. Note that `calculate_head_loss` requires that `bisection` be implemented.
- ☐ (1points) Does the solution from the `calculate_head_loss` program and `bisection` make sense? Explain why your answer is right in relation to the physics of the problem. Note: put this paragraph and calculated head loss in a small report.

References

- A Brief Introduction to MATLAB, *MatlabIntro.pdf*.
- A Brief Introduction to Python, *PythonIntro.pdf*.
- Optional Function Arguments in MATLAB and Python, *OptionalArgs.pdf*.
- An Introduction to Function Handles, *FunctionHandles.pdf*.
- Lecture 05: Introduction to Roots of Equations, *Lecture05.pdf*.
- Chapter 5.2, *Chapra, Numerical Methods for Engineers (7th Edition)*.

Physical Model

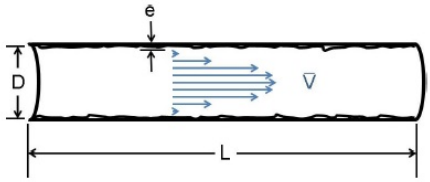

Figure 1: Pipe Flow Parameters

For this lab, you will be applying root-finding techniques to the analysis of fluid flow through a pipe (Figure 1). When analyzing such internal flow, we often want to calculate the head loss (a relative measure of pressure drop) in a section of pipe. One of the components of head loss (h_L) can be calculated using the formula:

$$h_L = f\frac{L}{D}\frac{\bar{V}^2}{2} \tag{1}$$

where L is the length of the pipe, D is the diameter, $\bar{V}$ is the average velocity of the flow in the pipe and f is the friction factor. An estimate for the friction factor can be found using the Colebrook equation:

$$\frac{1}{\sqrt{f}} = -2.0\log_{10}\left(\frac{e/D}{3.7} + \frac{2.51}{Re\sqrt{f}}\right) \tag{2}$$

where e is the roughness of the pipe and $Re = \frac{\rho \bar{V} D}{\mu}$ is the Reynolds number. ρ and μ are the density and viscosity of the fluid, respectively. The solutions to Equation 2 are plotted in Figure 2.

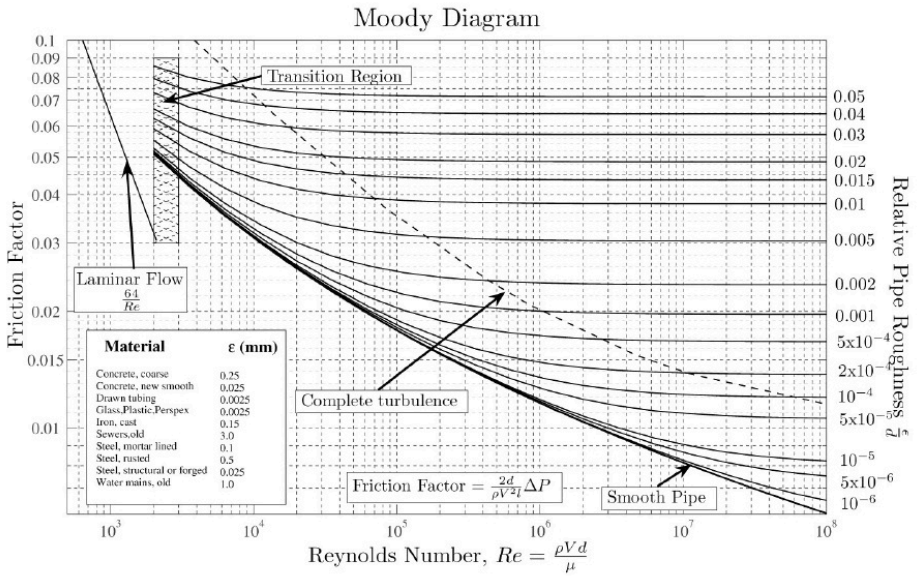

Figure 2: The Moody Diagram - Solutions to the Colebrook Equation [2]

While it is possible to solve Equation 2 explicitly for f [1], doing so is not easy or straightforward. It is therefore a perfect candidate for numerical root-finding methods.

Numerical Methods

Root Finding: The Bisection Method

The bisection method is a numerical root-finding technique that searches for roots of an equation by repeatedly decreasing the size of the interval in which the root exists. The bisection method is robust - it guarantees convergence, albeit slowly. To be a candidate for the bisection method, the function $f(x)$ must be continous on an interval $[a, b]$ that "brackets" the root, meaning that the signs of $f(a)$ and $f(b)$

must be opposite. By the intermediate value theorem, the function f will have at least one root in the interval (a, b).

For an acceptable function f defined on the interval $[a, b]$ containing at least one root (solution to $f(x) = 0$), the bisection method works by determining if the root is contained in the lower half of the interval $[a, (a + b)/2]$ or the upper $[(a + b)/2, b]$. If the root is contained in the lower half of the interval, that is if the signs of $f((a+b)/2)$ and $f(b)$ are opposite, the interval is redefined as $[a, (a+b)/2]$. Likewise, if the root is contained in the upper half, the interval is redefined as $[(a+b)/2, b]$. The process is continued until the interval size is sufficiently small such that $f((a + b)/2) = 0$ is satisfied to within a desired tolerance. Another possibility that should also be checked at each step of the process is that the dividing point of the intervals $(a + b)/2$ is the root.

The process is illustrated in Figure 3. In this figure, the bisection method is used to find the root of the function $f(x) = 6x^3 - 5x^2 + 7x - 2$. The root is determined to lie in the interval $[0, 1]$ and so these limits are chosen to begin the bisection method. The first step is to find the midpoint ($x = 0.5$). The sign of the function values at the end points and the midpoint are found. Since $f(0)$ is negative, while $f(0.5)$ and $f(1)$ are positive, the next interval is chosen to be $(0, 0.5)$. This interval is plotted, and the process is repeated. The first four steps of the Bisection Method are shown. The root estimate can be seen to converge toward the actual value of $\frac{1}{3}$.

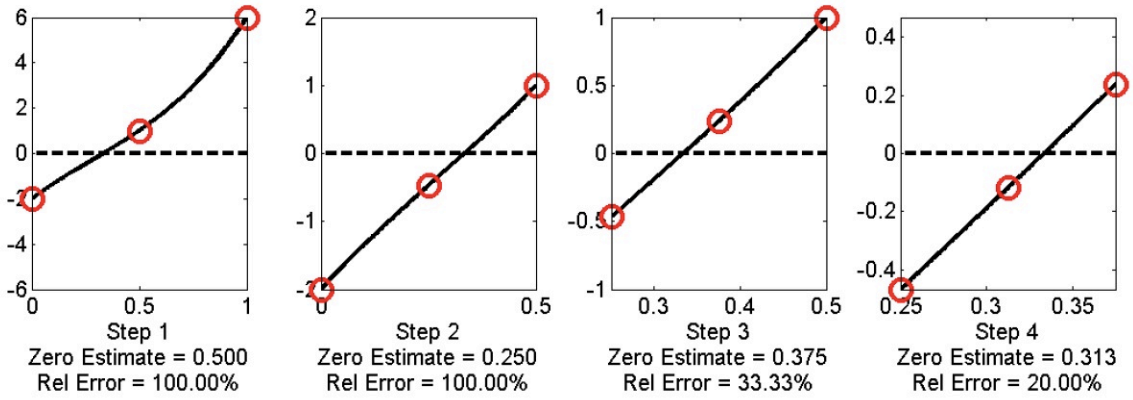

Figure 3: The bisection method used to find the root of $f(x) = 6x^3 - 5x^2 + 7x - 2$ in the interval $(0, 1)$.

The bisection method is implemented as shown in the following algorithm:

Data: $[a, b]$: initial interval, bracketing the root; tol: the tolerance
Result: c: the root, if found
while *iters < maxiter* **do**
    iters = iters + 1
    Calculate the new midpoint $c = \frac{a+b}{2}$
    if *iters>1* **then**
        Calculate approximate relative error: $\epsilon_a = \frac{(c - c_{old})}{c}$
        if $|f(c)| == 0$ *or* $|\epsilon_a| \leq tol$ **then**
            Exit loop
        end
    end
    if *Sign of* $f(c)$ *is same as the sign of* $f(a)$ **then**
        Set $a = c$
    else
        Set $b = c$
    end
    $c_{old} = c$
end
return c

There is another version of pseudocode for this algorithm in Figure 5.10 on page 130 of your textbook, and yet another version in the course lecture slides. It may be helpful to compare these two if you are having trouble understanding what you are supposed to do.

Lab Assignment

The program `calculate_head_loss` runs the head loss calculation by setting the appropriate simulation parameters.To be functional, it must be provided implementations of `bisection` and `colebrook_equation`, which you will write.

Bisection Method

Write a function `c = bisection(fun, a, b, [optional arguments])` that computes the root `c` of the function `fun`. The root `c` must be bracketed by the values `a` and `b`. `bisection` function should be implemented in a file named `bisection.[m,py]` (MATLAB or Python, respectively).

Input Arguments:

- `fun` (callable): Function for which you are to find a root. The function `y = fun(x)` for scalar `x` must return a scalar `y`.
- `a`, `b` (scalar): initial limits. The interval `[a b]` must bracket (at least) one root.

Output Arguments:

- `c` (scalar): the estimated root.

Optional Input Arguments:

- `tol` (scalar): Stopping tolerance. Defaults to `1e-6` if not provided.
- `maxiter` (int): Maximum number of iterations. Defaults to `10` if not provided.
- `plot_output` (boolean): Graphical output display flag. If `true` (`True` in python), plot the updated guess at each iteration. Defaults to `false` (`False` in python) if not provided.

Additional Requirements:

1. In the bisection function, three of the input parameters are optional, with sensible defaults set if the parameter is not specified by the user. See "Optional Function Arguments in MATLAB and Python" in *OptionalArgs.pdf* for an explanation of how optional arguments with defaults are handled in MATLAB or Python.
2. The graphical output you should display is similar to that shown in Figure 3. At the very least, it should plot the three points $(a, f(a))$, $(c, f(c))$, and $(b, f(b))$ at each iteration and pause for a short time between the iterations.
3. Your function should check whether either of the limits is a root. If one of the limits is a root, it should return that limit as the correct answer.
4. Your function should also verify that the value of the function `fun` at the limits `a` and `b` is such that these limits bracket a root (i.e., if $f(a) > 0$ then $f(b) < 0$, or vice versa). If the limits do not bracket a root, an error should be issued (use the `error` function in MATLAB or raise a `ValueError` in Python).
5. If the root is not determined to be within the desired tolerance in the maximum number of allowed iterations, an error should be issued (use the `error` function in MATLAB or raise a `RuntimeError` in Python).

Colebrook equation

Write a function `left_right_error = colebrook_eqution(f,e,D,Re)` that is used by Bisection to estimate the friction factor, the variable in `left_right_error`. The equation must be returned to Bisection method as a function (friction factor) = 0. Thus the right side must be subtracted from the left side or vice versa.`colebrook_equation` should be implimented in a file named `colebrook_equation.[m,py]` (MATLAB or Python, respectively).

Input Arguments:

- `f`: The friction factor which is a variable as described by the function handle written in the driver script.
- `e`: Roughness of the pipe
- `D`: Pipe diameter
- `Re`: Reynolds number

Output Arguments:

- `left_right_error` (equation): The Colebrook equation as a function of friction factor.

Calculating Head Loss

The script `calculate_head_loss.[m,py]`, which is the driver script, will be covered in class. The scripts implements (1) to determine head loss. The friction factor, modeled by (2), is determined using *your* `bisection` function.

References

[1] Ajinkya A. More, *Analytical solutions for the Colebrook and White equation and for pressure drop in ideal gas flow in pipes*, Journal of Chemical Engineering Science, **61** (2006) 5515-5519.

[2] Attributed to Donebythesecondlaw at the English language Wikipedia, (From the article titled Moody Chart, http://en.wikipedia.org/wiki/Moody_chart).