ME 2450 Lab 06: Solving Linear Equations Using Naïve Gauss Elimination

Objective

Write a computer program, `find_flow_rates`, that determines the unknown flow rates in a pipe network. The unknown flow rates are solutions to a system of linear equations, requiring a linear solver. To receive credit for the lab, write the linear solver, using naïve Gauss elimination, and the driver program `find_flow_rates` and demonstrate the correctness of each to your TA.

References

- An Introduction to Function Handles, *FunctionHandles.pdf*.
- Lecture 9: Solving Systems of Linear Equations: Gauss Elimination, *Lecture09pdf*.

Mathematical Model

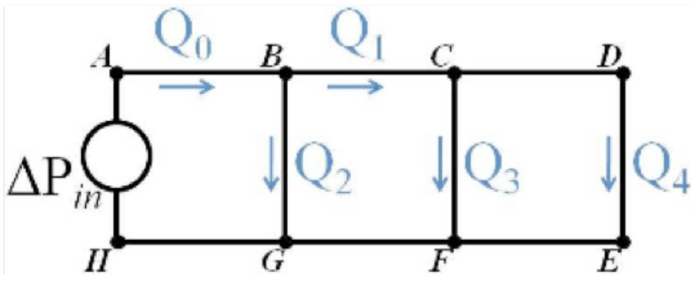

Figure 1: Pipe flow network

The figure above shows a simplified pipe network that could be similar to one found in a small house. Imagine that Q_0 is the total amount of water being used by the house. Q_2 is the flow rate required by the kitchen, Q_3 is the flow rate required by the bathroom, and any left over water is piped through the utility room at some other flow rate, Q_4. The water is circulated through the house's plumbing system using a pump that has some given pressure head, ΔP_0.

Imagine that you are the engineer designing this plumbing system. You will know certain specifications of the system, like the sizes (diameter and length) of the pipes being used and the pump pressure rise (ΔP_0), but will not know the flow rates through all of the pipes or the pressure losses in each pipe. Pressure losses occur for a variety of reasons, including the addition of fittings between the pipes (like elbow bends or tee joints) and from frictional losses along the length of the pipes. Losses due to pipe fittings are called minor losses and losses due to friction along a pipe's length are called major losses.

Although you know a limited amount about the system, you can still write a system of equations by applying the conservation of mass at each pipe node (a node occurs anywhere that a elbow bend or tee fitting connects two pipes). There are eight nodes in this pipe system. We can apply the principle of conservation of mass at nodes B and C, and assume the fluid density is constant throughout the system. This will result in a system of equations involving the flow rates:

$$\begin{aligned} Q_0 &= Q_1 + Q_2 \\ Q_1 &= Q_3 + Q_4 \end{aligned} \tag{1}$$

You could use conservation of mass at every node in the network, but in the end you would only have two meaningful equations with five unknowns (Q_0, Q_1, Q_2, Q_3, Q_4). Since there are more unknown flow rates than governing equations, it is necessary to introduce more equations (three more to be exact!) to solve the system. We'll do this by writing energy conservation equations for each pipe

As you learned in class, the principle of conservation of energy, when related to pipes, results in an equation that gives us the relationship between the flow rate and the pressure drop. Consider the flow shown in Figure 1 in the segment from B-C. The energy equation may be written as:

$$P_B + \rho g z_B + \frac{1}{2}\rho V_B^2 = P_C + \rho g z_C + \frac{1}{2}\rho V_C^2 + h_{L,major} + h_{L,minor} \tag{2}$$

where ρ is the fluid density, V_B is the velocity at the inlet to pipe section B-C and z_B is the vertical location of the pipe at the inlet to pipe section B-C. V_C and z_C are similarly defined at the outlet of pipe section B-C. The final two terms in the energy equation are the head losses, which come in the form of both minor ($h_{L,minor}$) and major losses ($h_{L,major}$). For flow through a pipe, major losses are defined by the formula:

$$h_{L,major} = f\frac{l}{D}\frac{\rho V^2}{2} \tag{3}$$

where f is the friction factor (which you saw in Lab 03), l is the pipe length, D is the pipe diameter, g is the acceleration due to gravity, and V is the average velocity in that section of pipe. The friction factor is a dimensionless variable that relates the pressure in a pipe to the flow rate. As you saw in Lab 03, the friction factor depends on the flow's Reynolds Number, the dimensionless ratio of inertial forces to viscous shearing forces in the fluid flow. Remember that the formula for Reynolds Number is:

$$Re = \frac{\rho V D}{\mu} \tag{4}$$

To simplify our analysis of the pipe flow network, we will assume three things:

1. There are no minor losses in the system ($h_{L,minor} = 0$).
2. The change in potential energy is negligible ($z_B = z_C$).
3. The flow rate does not change while passing through the pipe ($V_B = V_C = V_{BC}$)

Using these assumptions, we can eliminate terms from the energy equation and substitute for the major head loss, as follows:

$$P_B = P_C + f_{BC}\frac{l_{BC}}{D_{BC}}\frac{\rho V_{BC}^2}{2} \tag{5}$$

If we then define ΔP_{BC} as the change in pressure between B and C, we can rearrange the energy equation for each pipe as:

$$\Delta P_i = f_i \frac{l_i}{D_i} \frac{\rho V_i^2}{2} \tag{6}$$

We can then write the energy equation for each pipe section (i.e., between each of the nodes). If we neglect repeated equations, the result is five new equations. However, by writing these five equations, we also introduce several more unknowns, because we do not know the friction factor for each pipe or the pressure drops between the nodes of each pipe. We also do not know the average velocity in each pipe. We now have 7 equations but we have 20 unknowns! What other equations can we use to "close" our system?

We can add another assumption to our model of the system:

4. The density of the fluid is constant throughout the system ($\dot{m} = \rho Q = \rho V A$)

This allows us to rewrite the energy equation in terms of volumetric flow rate (since $V = \frac{Q}{A}$):

$$\Delta P_i = f_i \frac{l_i}{D_i} \frac{\rho}{2} \left(\frac{Q_i}{A_i} \right)^2 \tag{7}$$

This reduces our system of equations to 15 unknowns, although we still only have 7 equations. Another assumption that will help us is:

5. The flow in each pipe section is laminar. (For flow in a pipe to remain laminar, the Reynolds number must not exceed ~ 2000.)

This is particularly useful if we know that, for laminar pipe flow, the following holds true:

$$f_i = \frac{64}{Re} = \frac{64\mu}{\rho V_i D_i} = \frac{64 \mu A_i}{\rho Q_i D_i} \tag{8}$$

Since we know the length and diameter of each pipe, this means that we can substitute this formula for the friction factor into the simplified energy equation. Then the pressure drop between nodes is only a function of the flow rate in each pipe connecting those nodes. This further simplifies the energy equation as follows (assuming circular pipes):

$$\Delta P_i = \frac{64 \mu A_i}{\rho Q_i D_i} \frac{l_i}{D_i} \frac{\rho}{2} \left(\frac{Q_i}{A_i} \right)^2 = \frac{32 \mu l_i Q_i}{D_i^2 A_i} \tag{9}$$

We can then make one final assumption to simplify our calculations:

6. The pipes have a circular cross-section of diameter D_i.

Making the appropriate substitution for the area Ai in the previous equation yields:

$$\Delta P_i = \frac{128 \mu l_i}{\pi D_i^4} Q_i \tag{10}$$

This helps us cut down on the number of unknowns, as we now have only 10 (the five volumetric flow rates and the five pressure drops). However, we still do not have enough equations to close our system. Yet, if we look closely at this equation it resembles another relationship we've encountered in engineering, Ohm's Law. Ohm's Law relates the voltage drop across a wire to the resistance in the wire multiplied by the current running through the wire (from node to node). This sounds very similar to our fluid system problem. We have a pressure drop across a pipe with fluid running through the pipe (the flow rate). We can fully relate the fluidic system to an electrical circuit if we find an equivalent fluidic resistance.

$$\begin{matrix} V \sim \Delta P \\ I \sim Q \\ R \sim R_{fluidic} \end{matrix} \quad \Longrightarrow R_{fluidic,i} = 128\frac{\mu l_i}{\pi D_i^4} \tag{11}$$

where I is the current, V is the voltage, and R is the resistance. Furthermore, this means that we can apply the fact that the pressure drop around a closed loop sums to zero. This is analogous to another electrical concept, Kirchhoff's Voltage Law. Using this principle, we can add three new equations that rely only on variables that have already been introduced:

$$\begin{aligned} \Delta P_{HA} - \Delta P_{AB} - \Delta P_{BG} - \Delta P_{GH} = 0 \\ \Delta P_{GB} - \Delta P_{BC} - \Delta P_{CF} - \Delta P_{FG} = 0 \\ \Delta P_{FC} - \Delta P_{CD} - \Delta P_{DE} - \Delta P_{EF} = 0 \end{aligned} \tag{12}$$

Finally, we have the same number of equations (10) as unknowns! The mass and energy conservation laws resulted in seven equations with ten unknowns. With the introduction of the fluidic version of Kirchhoff's Law, we found the final three equations we needed to close our system.

To expedite solving these equations simultaneously, we must write them in the matrix equation form. The pressure drop for each pipe segment, 10, can be substituted into Equations 12 to cast 12 in terms of the flow rates, Q_i. With Equations 1 and 12 in terms of flow rates, we can form the following system of 5 equations in 5 unknowns (the volumetric flow rates):

$$\begin{bmatrix} 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 \\ R_{AB}+R_{GH} & 0 & R_{BG} & 0 & 0 \\ 0 & -R_{BC}-R_{FG} & R_{GB} & -R_{CF} & 0 \\ 0 & 0 & 0 & R_{FC} & -(R_{CD}+R_{DE}+R_{EF}) \end{bmatrix} \begin{Bmatrix} Q_0 \\ Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ \Delta P_0 \\ 0 \\ 0 \end{Bmatrix} \tag{13}$$

This system of linear equations in the familiar form $[A]\{x\} = \{b\}$ can be solved using any of the methods learned in this class. This week you will be using Naïve Gauss elimination to solve this system of equations for a laminar pipe flow network.

Numerical Methods

Naïve Gauss Elimination

For this week's lab, you will be writing a function that solves a system of linear equations using the Naïve Gauss elimination method. Consider the 3×3 system of equations in (14). Gauss elimination works by using forward elimination to eliminate one unknown at a time until there is one equation with one unknown, as depicted in (14). This equation is solved directly and the result is back-substituted into one of the original equations to solve for the remaining unknowns.

$$\left(\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right) \longrightarrow \left(\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ 0 & a_{22}^* & a_{23}^* & b_2^* \\ 0 & 0 & a_{33}^{**} & b_3^{**} \end{array}\right) \tag{14}$$

The problem with this approach is that it does not avoid the division of an unknown by zero. This is why the method is called “naïve.” We have discussed more sophisticated techniques that will improve upon this method. See Lecture 09 for more details on Gauss elimination.

Naïve Gauss Elimination: Algorithm

The algorithm for implementing naïve Gauss elimination is shown in Algorithm 1.

```
//
// Forward elimination
//
for $k = 1$ to $n - 1$ do
    for $i = k + 1$ to $n$ do
        $s = a_{ik}/a_{kk}$
        for $j = k$ to $n$ do
            $a_{ij} = a_{ij} - s\, a_{kj}$
        end
        $b_i = b_i - s\, b_k$
    end
end
//
// Backward solve
//
$x_n = b_n/a_{nn}$
for $i = n - 1$ to $1$ by $-1$ do
    $s = 0$
    for $j = i + 1$ to $n$ do
        $s = s + a_{ij}x_j$
    end
    $x_i = (b_i - s)/a_{ii}$
end
```

Algorithm 1: Naïve Gauss elimination algorithm.

Structure arrays and Dictionaries

Structure arrays, `structs` and dictionaries `dicts` are two similar ways of managing data in MatLab and Python, respectively. They can be used to store, access, and easily manipulate large, complex datasets. The documentation for `structs` is available here: https://www.mathworks.com/help/matlab/ref/struct.html and a tutorial with `dicts` examples is available here: https://www.datacamp.com/community/tutorials/python-dictionary-comprehension.
In the context of this lab, we will be using dictionaries and `structs` to manage the different pipe sections' properties in a compact, easy to access format.

As an example, consider a three pipe system, consisting of the first three segments from table 2. Instead of initializing two separate arrays that need to be tracked separately, we can link the data together in a `struct` or a dict. While the lab may be easy to complete without `structs` and `dicts`, they are essential for working with large datasets that can have many parameters. Codes 1 and 2 show how `structs` can be be created from cells (the building blocks of `structs` or lists of data (for the case of python). There is much more to `structs` and `dicts` than what will be used in this lab, but the above links provide supplemental information if you would like to use `structs` and `dicts` further.

Code 1: Python code example to demonstrate dictionaries.

```python
#initialize each array of values and the empty dict
pipeNameArr = ['AB','BG','CF']
lArr = [5,5,5]
dArr = [1,.5,.2]
pipeSegments = {}

#create a nested dict using the pipe names as keys and assigning
# the pipe's dimensions to nested dictionary elements
for idx, key in enumerate(pipeNameArr):
    pipeSegments[key] = {}
    pipeSegments[key]['l'] = lArr[idx]
    pipeSegments[key]['d'] = dArr[idx]

#perform a basic calculation to demonstrate the dict is working
for idx, key in enumerate(pipeSegments):
    pipeSegments[key]['aspectRatioLD'] = pipeSegments[key]['l']/pipeSegments[key]['d']
```

Code 2: MatLab code example to demonstrate structs.

```matlab
clear all
clc
% Initialize arrays needed to build the structure "pipeData":
pipeNames = {'AB','BG','CF'};
lengths = [5,5,5]; % lengths
diameters = [1,.5,.2]; % diameters

% Create a nested structure called "pipeData" where the first field is
% the pipe name, and the second field is the pipe property:
% (i.e. length or diameter)
for i = 1:length(pipeNames)
        pipeData.(pipeNames{i}).p_length = lengths(i);
        pipeData.(pipeNames{i}).p_diam = diameters(i);
end


% Perform a basic calculation (find the pipe's aspect ratio) to demonstrate
% how to add a new pipe property to pipeData:
for i = 1:length(pipeNames)
pipeData.(pipeNames{i}).aspectRatio = ...
                                pipeData.(pipeNames{i}).p_length / pipeData.(pipeNames{i}).p_diam;
end
```

Gauss Elimination

Define a function, `naive_gauss_elimination`, that takes as input the coefficient matrix, A, and right-hand side vector, b, and returns the unknowns, x.

Input Arguments:

- `A` (matrix): Coefficient matrix.
- `b` (vector): Right-hand side vector. Size must be consistent with `A`.

Output Arguments:

- `x` (vector): Solution to the system $Ax = b$. Returned shape is identical to `b`.

Requirements

- Your function should check for appropriate sized matrices. This includes making sure the A is a square matrix and that the b vector is of the appropriate size compared to the A matrix.
- Your function should check for a zero along the diagonal. Think about why a zero on the diagonal would cause the script to fail by referring to Algorithm 1.

- Your function should test for diagonal and upper triangular matrices. Gaussian Elimination won't have an effect on these. Your function should immediately transition to backward substitution.

Resistance Function

Define a function that takes as input viscosity, a `struct` (for MatLab) or a `dictionary` for Python, which stores the pipe-segment names, lengths, and diameters, and gravity (as an optional parameter with default set to 9.81 $[\frac{m}{s^2}]$. The function should calculate the fluid resistance of the pipe-segment and return the updated data structure.

Input Arguments:

- `mu` (scalar): Viscosity.
- `pipeParameters` (struct/dict): A struct or dict that contains the pipe names, lengths, and diameters.
- g (scalar)(optional): Acceleration due to gravity.

Output Arguments:

- `pipeParameters` (struct/dict): The updated struct or dict that contains the pipe names, lengths, diameters, and fluid resistances.

Driver Script

Write the program `find_flow_rates`. The program:

1. Is a standalone script that sets up and calls `naive_gauss_elimination`;
2. Should determine the five unknown flow rates for $\Delta P_0 = 100\text{kPa}$, $\mu = 1 \times 10^{-3}$N s/m^2, $\rho = 1 \times 10^3$kg/m^3 $g = 9.81 m/s^2$ and the following input parameters:

$l_{AB} = 5\text{m}$	$D_{AB} = 1\text{cm}$
$l_{BG} = 5\text{m}$	$D_{BG} = 0.5\text{cm}$
$l_{CF} = 5\text{m}$	$D_{CF} = 0.2\text{cm}$
$l_{DE} = 5\text{m}$	$D_{DE} = 0.3\text{cm}$
$l_{GH} = 5\text{m}$	$D_{GH} = 0.5\text{cm}$
$l_{BC} = 10\text{m}$	$D_{BC} = 0.5\text{cm}$
$l_{CD} = 10\text{m}$	$D_{CD} = 0.35\text{cm}$
$l_{EF} = 10\text{m}$	$D_{EF} = 0.25\text{cm}$
$l_{FG} = 10\text{m}$	$D_{FG} = 0.45\text{cm}$

3. Should produce the values required to make a table of the flow rate solutions: Feel free to use excel.
4. Calculate the Reynolds number in each of the pipes (Note: there are more than five pipes), and report if any of these indicates that the flow is not laminar in the pipe.
5. Hints: Create a function handle for calculating fluidic resistance using eq. 11. Use the Reynolds Number Formula $Re = \frac{\rho V D}{\mu}$.

Scoring of Lab Exercises

- ☐ (5 points) Use a `struct` in MatLab, or `dictionary` in Python to organize the data given in Table 2. The data structure should use the "AB", "BG", etc. as names to match the pipe segment with its parameters.
- ☐ (5 points) Send the `struct` or the `dictionary` to a function, `fluid_resistance`, which calculates the fluid resistance, adds it to the data structure, and returns the updated `struct` or `dictionary`.
- ☐ (10 points) Write a function, `naive_gauss_elimination`, using Algorithm 1.
- ☐ (6 points) write a driver script, `find_flow_rates`, that determines the unknown flow rates. `find_flow_rates` should print a table of the computed flow rates, calculate the Reynolds number in each of the pipes, and report if any of pipes are experiencing nonlaminar flow. Report the flow rate in each pipe along with the Reynolds number and determination of whether the flow is laminar to your TA.
- ☐ (2 point) How can you verify that your answer is correct? Demonstrate this in a couple short sentences and turn your report in with all the other code used/written in this lab. Hint: remember you're solving a system of equations.