

GNG1106 – Fundamentals of Engineering Computation

Course Project

Deliverable #3

Pipe Friction - The Friction Factor

Team Members

Amit Nayak - 0300066780

Jake Martin - 0300071920

Date:

November 25th, 2018

1 Problem Identification and Statement

In many industries, understanding the relationship between a pipe's diameter and the resistance to flow can be integral to designing efficient fluid transportation systems, such as liquid cooling and fluid transportation. This program seeks to remedy this problem by plotting the relationship between a pipe's diameter and the friction factor of flow. The program will take in the following parameters:

- Max, min pipe diameter (m)
- Roughness (entered in mm, converted to m internally)
- Flow rate (m/s)
- Fluid density (kg/m³)
- Fluid viscosity (s/m²)

The user can either load these parameters from file or type them in manually. Using these parameters and the assumption that flow rate is constant and that the flow is turbulent rather than laminar, the program will create a graph representing the relationship between pipe diameter and fluid friction factor.

2 Gathering of Information and Input/Output Description

2.1 Reynolds Equation

The Reynolds equation used to generate a Reynolds number, a dimensionless value for which the fluid's behaviour is defined by. A Reynolds number lesser than 2300 indicates laminar flow, while a Reynolds number greater than 4000 indicates turbulent flow. The Reynolds number can be found from the equation:

Eqn. 1:

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

Where Re is the Reynolds number, dimensionless

Where ρ is the fluid's density in kg/m³

Where V is the fluid's velocity in m/s

Where D is the pipe's diameter in m

Where μ is the pipe's dynamic viscosity in N·m/s²

2.2 Colebrook Equation

The Colebrook equation will be used to calculate the friction factor, a dimensionless value represented by f . The Colebrook equation only applies under turbulent flow, which is when the Reynolds number (generated by the Reynolds equation, Eqn.1) exceeds a certain threshold: $Re > 4000$. Furthermore, the Colebrook equation uses the Reynolds number in its own calculation, solidifying the ties between the two equations. All values in the Colebrook equation are either entered by the user (ε , D) or calculated (Re) in a previous step *except* f . This leaves one unknown (f) in a polynomial equation with multiple roots. The process of finding these roots is found in Subsection 3. The Colebrook equation, arranged to one side, is as follows:

Eqn. 2:

$$0 = \frac{1}{\sqrt{f}} + 2.0 \log \left(\frac{\varepsilon}{3.7D} + \frac{2.51}{Re\sqrt{f}} \right)$$

Where ε is the pipe roughness in m

Where D is the pipe diameter in m

Where Re is the Reynold's number calculated for the same dataset

Where f is the friction factor, dimensionless

2.3 Solving the Colebrook Equation: Bisection of Roots Method

The roots of the Colebrook equation can be found using the bisection method. The bisection method can be used to find the roots of the Colebrook equation since it is a nonlinear function. It starts with 2 initial guesses for the bounds of the interval. This is a repetitive approach as the bisection method keeps dividing the interval in half for accurately finding the root. When the function change sign over the interval, the program calculates the midpoint of the interval. Then the root estimated to be at the midpoint of the subinterval where the sign of the function changes. This process is looped until an accurate value is found. Once a root is found, the next interval is set with the lower bound being slightly higher than the root that was found. The process is then repeated. The following is a set of steps which acts as the algorithm for the bisection method:

Make two guesses of a lower bound (Left) and an upper bound (Right) for the search interval. The function must change signs therefore $f(x_L) * f(x_U) < 0$. If the interval does not satisfy these conditions, find another interval.

Calculate an estimate of the root x_R with $x_R = (x_L + x_U) / 2$

Use the following guidelines to determine which subinterval the root lies in:

If $f(x_L)f(x_R) < 0$, the root lies in the lower subinterval. Therefore, set $x_U = x_R$ and go back to step 2.

If $f(x_L)f(x_R) > 0$, the root lies in the upper subinterval. Therefore, set $x_L = x_R$ and go back to step 2.

If $|f(x_L)f(x_R)| < \text{some small tolerance value}$ (that is, can be considered equal to 0, ie. 0.0001), the root equals x_R ; terminate the computation.

2.4 Input and Output

Input

The program begins by asking the user to user select a file containing the values for the calculation, or to manually enter them by keyboard.

Should the user opt to open a file, the program will ask the user for the filename of the file to open. The program will display:

Please enter the filename of the file to open:

Valid filenames are:

- Less than 260 characters (windows/ntfs)
- Do not contain any characters of the following: / ? < > \ : * |

Should the user opt to enter the values manually, the program will prompt the user for each value as follows:

Maximum and Minimum diameter of the pipe. These values must be given in m. It must be noted that the minimum fluid velocity for a specific flow rate can be determined by the maximum diameter. The pipe diameter will be calculated and used in the Colebrook equation.

The program will display:

Please enter the Minimum and Maximum diameters in m:

Valid values are:

- Greater than 0
- Maximum must be greater than Minimum

The pipe roughness is to be given in $\epsilon = \text{mm}$. This value is to be converted into m for use in the Colebrook equation.

The program will display:

Please enter roughness of the pipe in mm:

Valid values are:

- Between 0.0001 and 3

The flow rate is to be given in $V = \text{m/s}$. This will be used for the calculation of the Reynolds number using the Reynolds equation.

The program will display:

Please enter the flow rate in m/s:

Valid values are:

- Greater than 0

The fluid density is to be given in $\rho = \text{kg/m}^3$. Fluid density is to be used in the Reynolds equation.

The program will display:

Please enter the fluid density in kg/m^3 :

Valid values are:

- Between 0.5 and 2000

The fluid viscosity is to be given in μ in N s/m^2 . This will be used in the Reynolds equation.

The program will display:

Please enter the viscosity in N*s/m^2 :

Valid values are:

- Between 1×10^{-6} to 300

After entering the values, the program will ask the user whether or not to save the entered values as a file. The program will display:

Do you want to save these values to file (y/n):

Should the user desire to save the values to file (y), then the program will ask the user for a filename to save the values under. The program will display:

Please enter the filename of the file to open:

For which the program will attempt to open that file and read in the data.

Output

The program will output a graphical plot of the relationship of friction factor vs pipe diameter, while keeping the fluid flow rate constant. The plot will have friction factor in the y-axis and the change in diameter in the x-axis.

3 Test Cases and Design

3.1 Test Cases

These are the criteria for the test cases:

- Reynolds Number calculated from the input data must be greater than 4000.
- Roughness converted to m: 0.0001 to 3
- Fluid Density: 0.5 to 2000
- Dynamic Viscosity: 10^{-6} to 300.
- All values must be positive.

Fluid Density, (ρ) (kg/m^3)	flow rate, (V) (m/s)	Dynamic viscosity, (μ) (N s/m^2)	pipe diameter, (D) (m)	Roughness, (ϵ) (m)	Reynolds Number, (Re)	Friction factor, (f)
0.5	1.5	1.5×10^{-6}	0.7	3.7×10^{-4}	350000	0.0183
1300	4	10	11	0.003	5720	0.0367
2000	40	300	16	3×10^{-5}	4270	0.039
1070	25	50	7.5	1×10^{-7}	4010	0.040
0.1*	100*	400*	10*	4*	0.25*	0.147* *invalid case

3.2 Design

The following section provides an introduction to the code as well as a breakdown of the code by function. The breakdown includes user interaction, plotting, calculations and file use. This program relies heavily on the use of function in order to keep the code readable and modular.

3.2.1 Introduction

The software being develop relies on the user entering values for a set number of required variables used in computation. The program can only calculate the friction factor using the Colebrook equation if the values inputted provide turbulent results. This can be found once the Reynolds number is calculated ($\text{Re} > 4000$). Once all the user's values are inputted and the values provide turbulent results, the algorithm calculates the friction factor using the bisection root finding method. The variables used by the code are: fluid density, viscosity, roughness, diameter, and Reynolds number. At the end of the calculations the results will be plotted. After that the user will be prompted to whether they would like to repeat the program with a different set of values. If the user does not wish to continue the program will quit otherwise the program will restart.

3.2.2 Functions for Interacting with the User

```
void GetInput(char name[], FACTOR *frictPTR);
```

Parameters:

Factor friction: Structure that contains all the variables for calculations including: fluid density, velocity, dynamic viscosity, pipe diameter, roughness, and reynolds number. These variables are to be used in calculations.

Return Value

This function does not return any value as it is a void function. Instead it stores user inputted data into the friction struct.

Logic/Algorithm

This function prompts the user to enter in a value then calls the getPositiveValue function to take in user input and make sure that it is greater than 0. This function repeats this procedure of prompting then taking in user input until all of the data required is inputted.

```
double getPositiveValue()
```

Parameters:

There are no parameters for this function.

Return Value:

The number inputted by the user is returned as a double. These values will be stored in the friction structure variable for use later in calculations.

Logic/Algorithm:

The function initializes a double variable. Then the function uses a do-while (indeterminate) loop that scans a double and stores it into the double variable. Afterwards the function uses an if statement to check if the value of the variable is less than or equal to 0. If it is less than or equal to 0 then it will prompt the user to enter a value that is greater than 0. This do-while loop runs the first time without any conditions, then continues looping if the value is less than or equal to 0. If the value is greater than 0, it will return the value.

```
void getValidFileName(char name[])
```

Parameters:

char name[]: character array of potential file name; as it is an array, it is inherently passed by reference/address rather than value.

Return Value:

This function does not return any value as it is a void function.

Logic/Algorithm:

The function uses a do-while (indeterminate) loop that read in a string from the user and stores it in the name array. Then the function checks if this name is valid for a file system (length below 260 characters, does not contain characters like / ? < > \ : * |). If the input is invalid, it will loop back and ask the user again to input a name for the file, until a valid name is found.

3.2.3 Functions for Calculations

```
double calcReynolds (FACTOR *frictPTR)
```

Parameters:

Factor friction: Structure that contains all the variables for calculations including: fluid density, velocity, dynamic viscosity, pipe diameter, roughness, and Reynolds Number. These variables are to be used in calculations.

Return Value:

This function returns the calculated Reynolds number as a double.

Logic/Algorithm:

The function uses values stored in the friction struct and calculates the Reynolds number using Eqn. 1 (Reynolds Equation). Since input filtering is done in other steps, we can assume that the values used in the equation are valid. Thus this equation can be expressed in just one line of code. The calculated result of the equation is returned by the function for use elsewhere.

```
double calcFx(double FNum, FACTOR *frictPTR)
```

Parameters:

Factor friction: Structure that contains all the variables for calculations including: fluid density, velocity, dynamic viscosity, pipe diameter, roughness, and Reynolds Number. These variables are to be used in calculations.

double FNum: the value of the friction factor.

Return Value:

This function returns $f(x)$ given the friction factor with the values passed in.

Logic/Algorithm:

Initializes a variable and then uses the equation 2 to calculate and save $f(x)$ to that variable. Afterwards the function returns the variable.

```
double rootFind(double leftpt, double rightpt, FACTOR *frictPTR)
```


Parameters:

Factor friction: Structure that contains all the variables for calculations including: fluid density, velocity, dynamic viscosity, pipe diameter, roughness, and Reynolds Number. These variables are to be used in calculations.

low is the value of the lower limit of the interval/function.

high is the value of the lower limit of the interval/function.

Return Value:

This function returns the calculated root as a double.

Logic/Algorithm:

This function initializes a (xr) variable for the root. The function uses the calcFx function to calculate f(x). The checks if $f(\text{low}) * f(\text{high})$ is less than 0. If this is not the case the function needs new intervals. The root found flag is set to false. In a while loop where the flag (root found) is false, function then makes an estimate root $\text{root_temp} = (\text{low} + \text{high}) / 2$. Then using an if elseif statement series, if the product of $f(\text{low})$ and $f(\text{xr})$ is less than 0 then the function sets high equal to xr and goes back to the estimate root step. If the product of $f(\text{low})$ and $f(\text{xr})$ is greater than 0 then the function sets low equal to xr and goes back to the estimate root step. If the product of $|f(\text{low})$ and $f(\text{xr})|$ is less than 0.0001, the root is xr. The root found flag is set to true and xr is returned.

```
void calcPoints(FACTOR *frictPTR, double points[][MAXSIZE])
```

Parameters:

Factor friction: Structure that contains all the variables for calculations including: fluid density, velocity, dynamic viscosity, pipe diameter, roughness, and Reynolds Number. These variables are to be used in calculations.

low is the value of the lower limit of the interval/function.

high is the value of the lower limit of the interval/function.

double points[][MAXSIZE] is the array that contains the diameters and the friction factors after calculations. This is used for plotting.

Return Value:

This function does not return anything as it is a void function.

Logic/Algorithm:

The function create an incremental value for the plotting values of the function with $\text{high-low}/(\text{number of increments}-1)$. A for loop based on the number of increments will use the function calcFx to calculate the values. The values will then be saved to the 2D array in the Factor friction structure array.

3.2.4 Functions for Plotting

```
void plot(FACTOR *frictPTR, double points[][MAXSIZE])
```

Parameters:

`Factor friction`: Structure that contains all the variables for calculations including: fluid density, velocity, dynamic viscosity, pipe diameter, 2D array of calculated values and roughness.

`double points[][MAXSIZE]` is the array that contains the diameters and the friction factors after calculations. This is used for plotting.

Return Value:

This function does not return any value as it is a void function.

Logic/Algorithm:

Graphing is done by PLplot; first, the compiler set to wingcc and PLplot is initialized. Then, the parameters of the graph (axis names, colours, line thickness, etc) are established using their respective functions. For our application, the pipe diameter becomes the independent (x) axis and the friction factor becomes the dependant (y) axis. Then the relation is graphed, by calculating the friction factor at different pipe diameters. The bounds of the x axis are defined by the user-entered max and min values for the pipe, which are found in the friction struct.

3.2.5 Functions for Files

```
int writeFile(FACTOR *frictPTR, char name[])
```

Parameters:

`Factor friction`: Structure that contains all the variables for calculations including: fluid density, velocity, dynamic viscosity, pipe diameter and roughness.

Return Value:

Boolean (0 or 1) whether or not the write was successful. 0 = success, 1 = fail

Logic/Algorithm:

This function will write the user-inputted data contained within the friction struct to a file. First, a filename character array is created and passed by reference to `getValidFileName`, where it will receive a valid filename. Then the file (and its pointer) is initialized with the name. We then check if the file (and its pointer) exist; if it doesn't, the function returns 1. Once the file is established to exist, `fprintf` is used to write the max, min pipe diameter, roughness, flow rate, fluid density, and fluid viscosity to file, all of which are found in the friction struct. The function will conclude by returning 0 indicating everything worked properly.

```
int readFile(FACTOR *frictPTR, char name[])
```

Parameters:

`Factor friction[]`: Structure (in an array of 5) that contains all the variables for calculations including: fluid density, velocity, dynamic viscosity, pipe diameter and roughness.

Return Value:

Boolean (0 or 1) whether or not the read was successful. 0 = success, 1 = fail

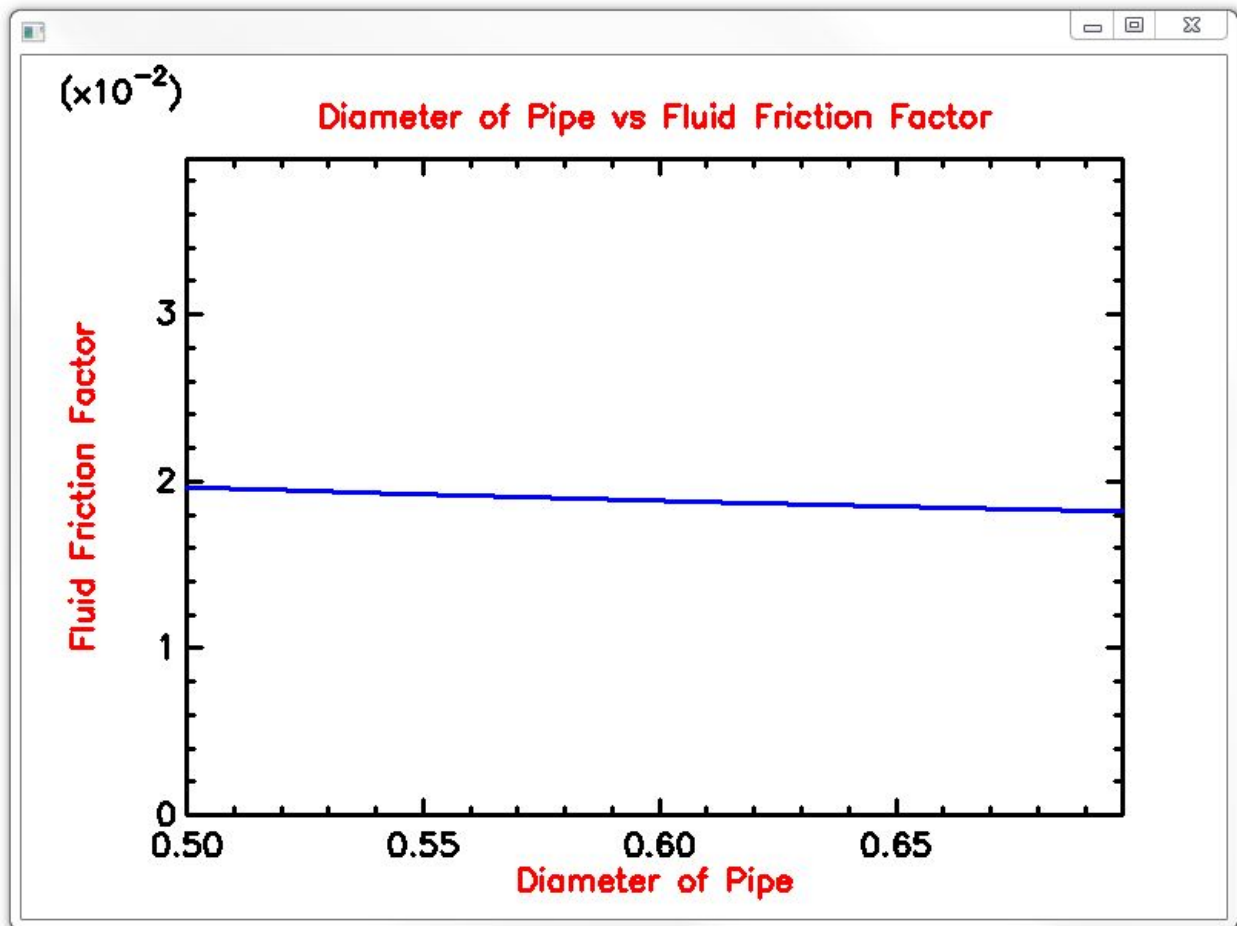
Logic/Algorithm:

This function will read files created for this program, taking in values for max, min pipe diameter, roughness, flow rate, fluid density, and fluid viscosity. First, a filename character array is created and passed by reference to `getValidFileName`, where it will receive a valid filename. A pointer to that file will be created and the function will attempt to open the file; should it fail to open the file, it will loop back and ask the user to input a file name through the `getValidFileName` function. Should it succeed in opening the file, `fscanf` will be used to read the values in the file and place them in their respective members of the friction struct. Once this is done, the function will return 0 indicating everything worked expectedly.

4.0 Implementation

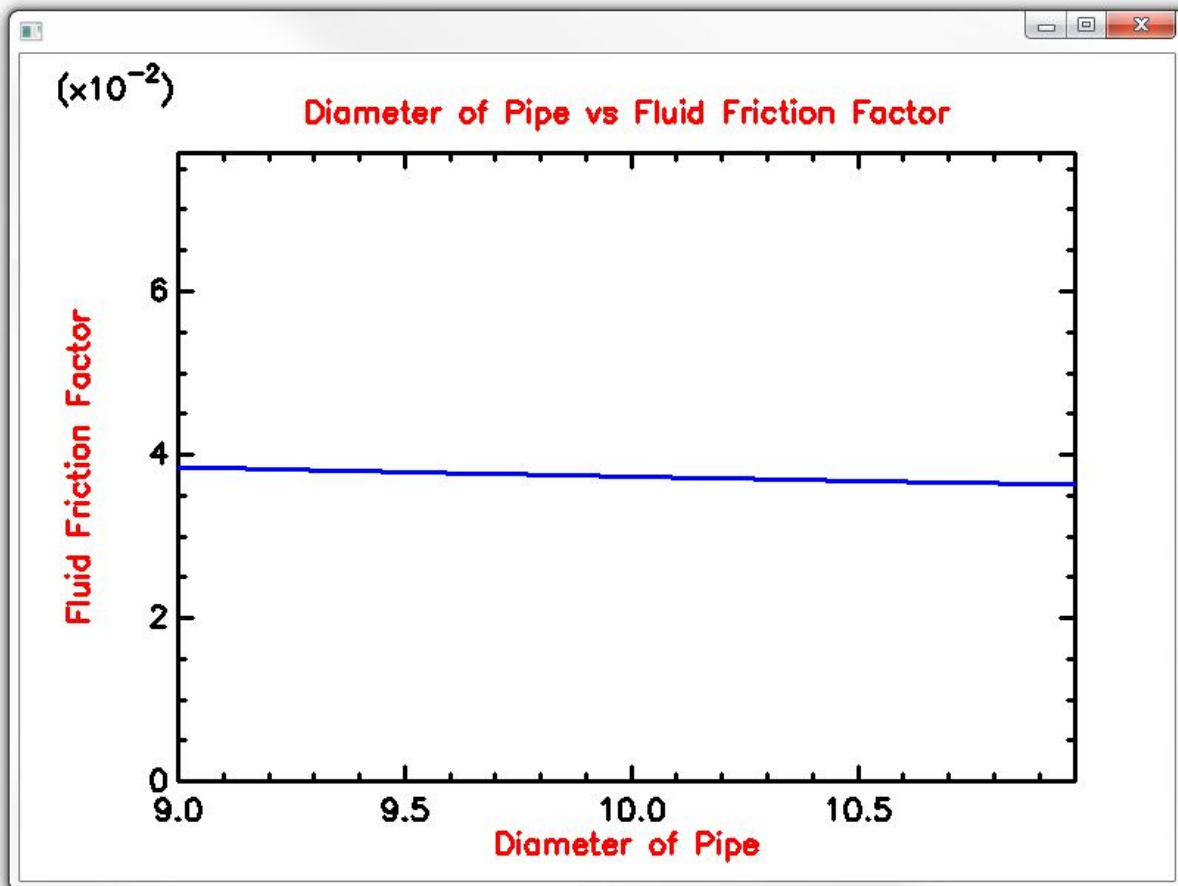
See Project Files ZIP for source code.

5.0 Software Testing and Verification

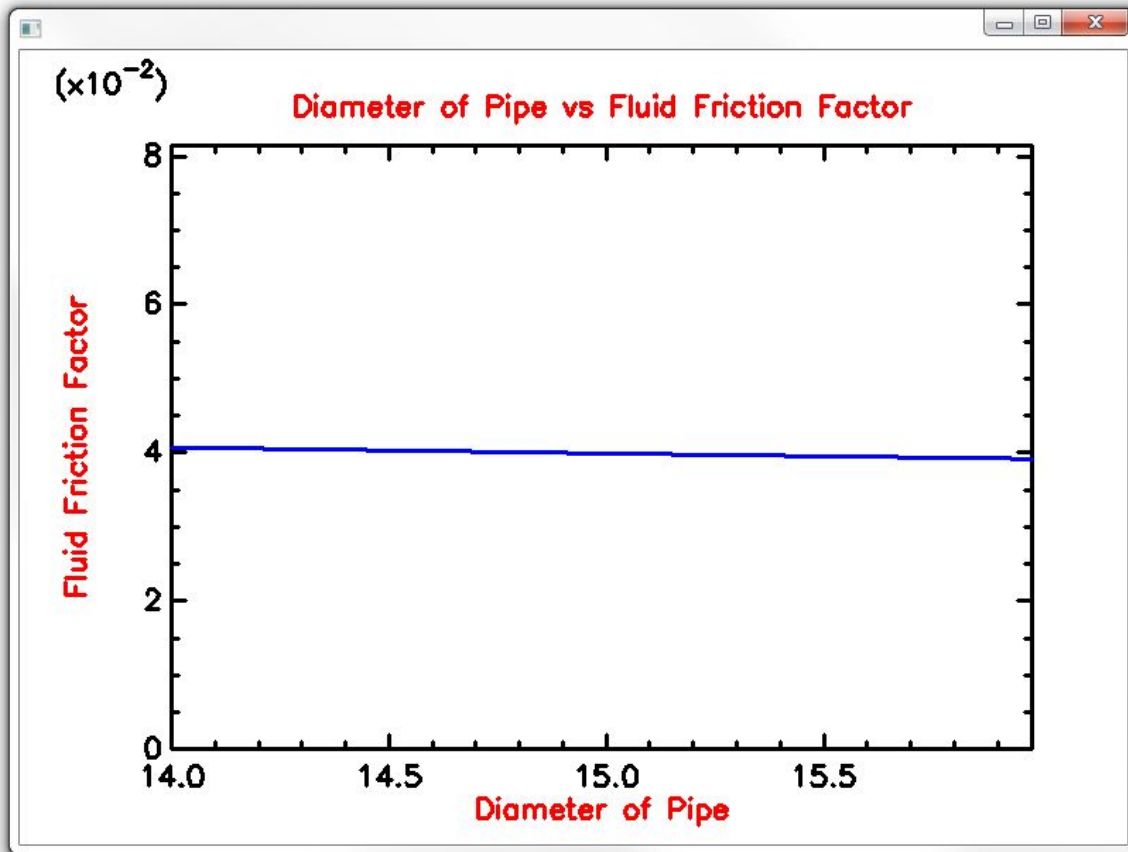


```
C:\Users\Jake\Desktop\deliverable\bin\Debug\deliverable.exe
Pipe Friction Factor v1.1 - By Amit Nayak & Jake Martin
Open file or enter data manually? <o/m>
m
Please enter the Minimum and Maximum diameters in m:
0.5
0.7
Please enter roughness of the pipe in m:
3.7e-4
Please enter the flow rate in m/s:
1.5
Please enter the fluid density in kg/m^3:
0.5
Please enter the viscosity in N*s/m^2:
1.5e-6
Save entered data? <y/n>
y
Enter a valid filename:
testcase1
Friction factor at minimum diameter is 0.019659
Friction factor at maximum diameter is 0.018198

Process returned 0 (0x0)   execution time : 98.717 s
Press any key to continue.
```

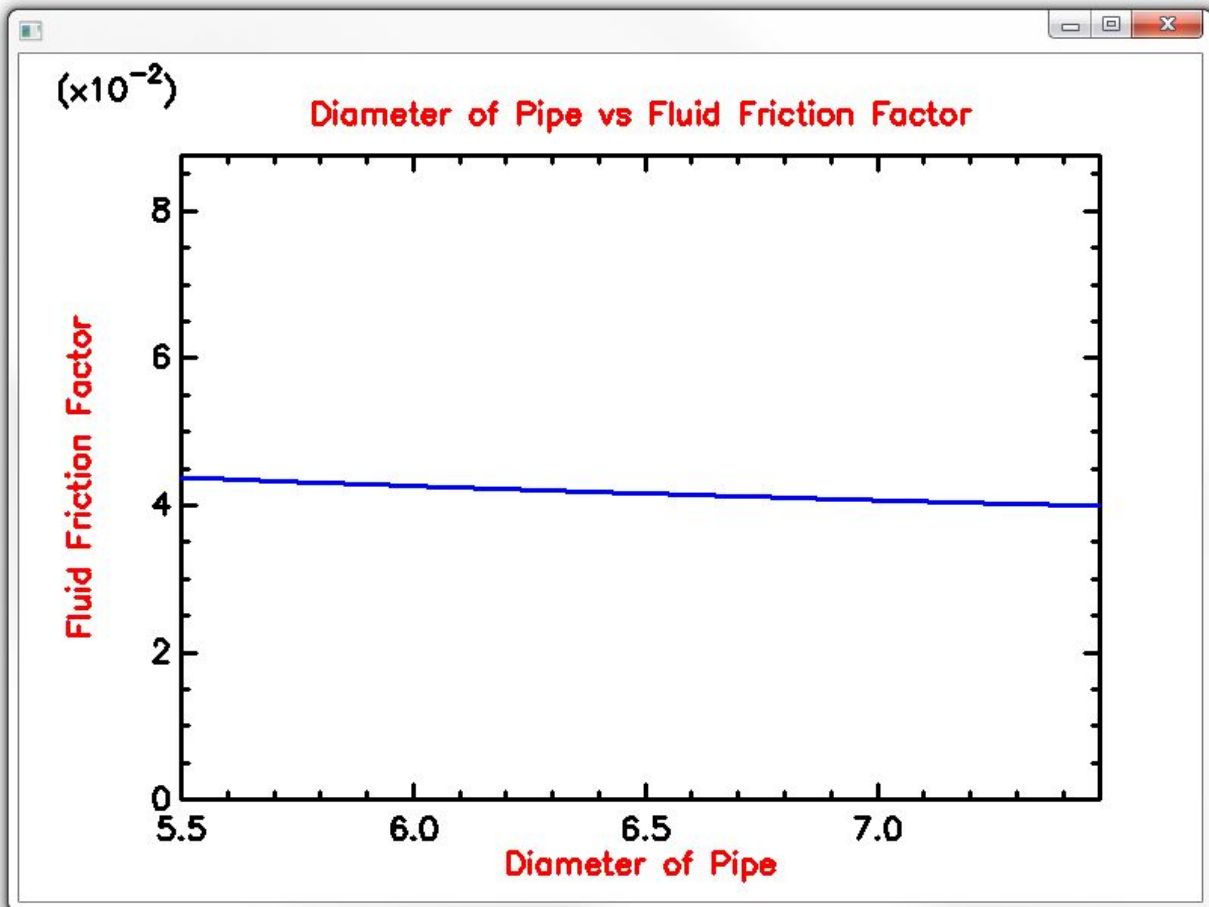


```
CA\Users\Jake\Desktop\deliverable\bin\Debug\deliverable.exe
Pipe Friction Factor v1.1 - By Amit Nayak & Jake Martin
Open file or enter data manually? <o/m>
m
Please enter the Minimum and Maximum diameters in m:
9
11
Please enter roughness of the pipe in m:
0.003
Please enter the flow rate in m/s:
4
Please enter the fluid density in kg/m^3:
1300
Please enter the viscosity in N*s/m^2:
10
Save entered data? <y/n>
y
Enter a valid filename:
testcase2
Friction factor at minimum diameter is 0.038475
Friction factor at maximum diameter is 0.036327
Process returned 0 (0x0)   execution time : 75.905 s
Press any key to continue.
```



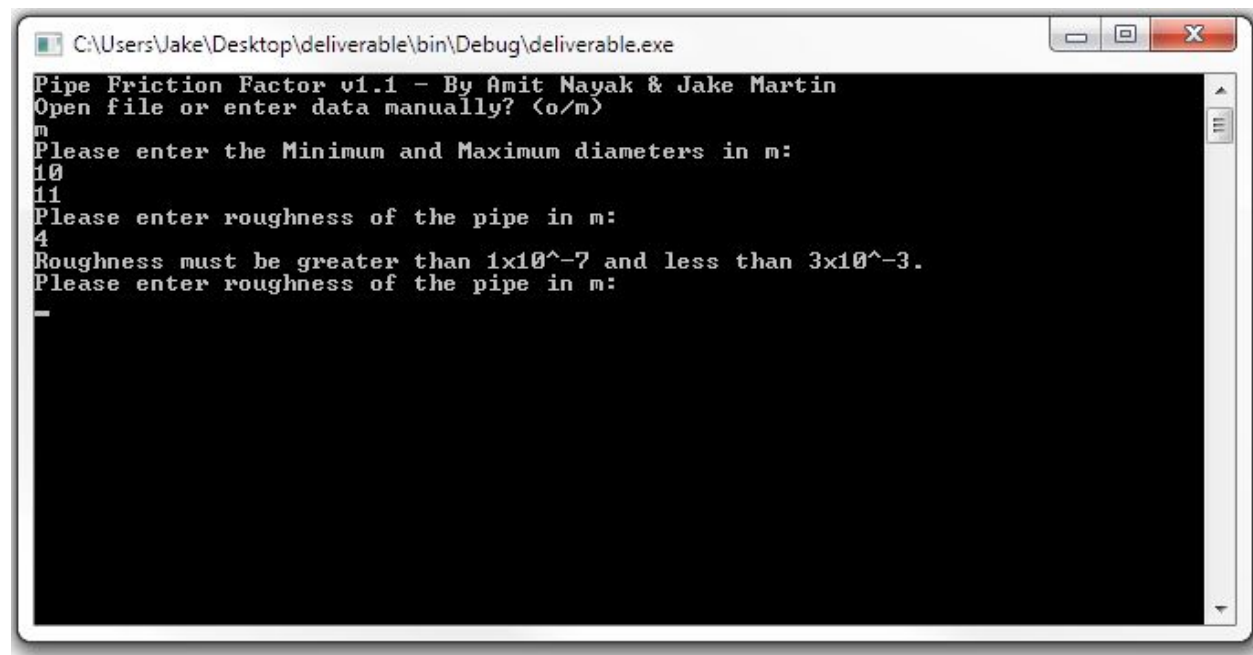
```
C:\Users\Jake\Desktop\deliverable\bin\Debug\deliverable.exe
Pipe Friction Factor v1.1 - By Amit Nayak & Jake Martin
Open file or enter data manually? (o/m)
m
Please enter the Minimum and Maximum diameters in m:
14
16
Please enter roughness of the pipe in m:
3e-5
Please enter the flow rate in m/s:
40
Please enter the fluid density in kg/m^3:
2000
Please enter the viscosity in N*s/m^2:
300
Save entered data? (y/n)
y
Enter a valid filename:
testcase3
Friction factor at minimum diameter is 0.040735
Friction factor at maximum diameter is 0.039172

Process returned 0 (0x0)   execution time : 76.054 s
Press any key to continue.
```



```
C:\Users\Jake\Desktop\deliverable\bin\Debug\deliverable.exe
Pipe Friction Factor v1.1 - By Amit Nayak & Jake Martin
Open file or enter data manually? (o/n)
n
Please enter the Minimum and Maximum diameters in m:
5.5
7.5
Please enter roughness of the pipe in m:
1e-7
Please enter the flow rate in m/s:
25
Please enter the fluid density in kg/m^3:
1070
Please enter the viscosity in N*s/m^2:
50
Save entered data? (y/n)
y
Enter a valid filename:
testcase4
Friction factor at minimum diameter is 0.043779
Friction factor at maximum diameter is 0.039902

Process returned 0 (0x0)   execution time : 67.747 s
Press any key to continue.
-
```



```
C:\Users\Jake\Desktop\deliverable\bin\Debug\deliverable.exe
Pipe Friction Factor v1.1 - By Amit Nayak & Jake Martin
Open file or enter data manually? (o/m)
m
Please enter the Minimum and Maximum diameters in m:
10
11
Please enter roughness of the pipe in m:
4
Roughness must be greater than 1x10^-7 and less than 3x10^-3.
Please enter roughness of the pipe in m:
-
```