The University of Nottingham

UNITED KINGDOM • CHINA • MALAYSIA

Coursework 1

Hassan Miah
4254974

Contents

Part a)

For turbulent flow inside the pipes, friction factor/coefficient (f) is dependent on the dimensionless surface roughness (ε/D) of the pipe inner surface and the dimensionless Reynolds number (Re), as given by the Colebrook equation below:

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

Equation 1

Where Re can be obtained from: $Re = \frac{\rho D Q}{\mu A}$.

In the equation of Reynolds number above, Q is the volumetric flowrate of the fluid, D is the inner diameter of the pipe, A is the cross-sectional area of the pipe, and ρ and μ are the density and viscosity of the fluid, respectively.

a . **Develop your own code and solver in Matlab**; and compute the friction factor for a general turbulent flow i.e the code will return friction factor, f, for any Re, ε and D values. Compute the friction factor of a turbulent flow of an oil through a smooth pipe of 4-in inner diameter at a volumetric flowrate of 2000 bbl/day. The dimensionless surface roughness of a smooth pipe is zero. The density and viscosity of the oil are 0.9 g/cm3 and 8 cp, respectively. Unit conversion: 1 bbl (oil barrel) = 42 gal; 1 cp = 0.001 Pa·s; 1 gal = 3.785×10−3 m3. Compare your result with that computed using built-in functions such as *vpasolve*. Discuss and compare both approaches. Discuss the application of turbulent flow pipe

Bisection Method

Matlab Code:

```
clear all;
clc;

error = 1;
tol = 10e-5;
a = -5; %the lower endpoint of the [a,b] interval
b = 5; %the higher endpoint of the [a,b] interval
k = 0; %main algorithm
while error > tol % Loop ends when condition is violated.
```

Figure 1 - Part 1 of code

Figure 1 explains to Matlab the range for this bisection to calculate x-values between a limit, -5 and 5. A tolerance is set for the error to be less than 0.00001 therefore the error is small enough to be negligible. If error is greater than tolerance the loop will stop iterating. K = 0 is the initial number of iteration.

```
p = 0.9*1000; % density (kg/m3)
u = 8*0.001; % viscosity (Pa.s)
D = 4*0.0254; % Inner Diameter of pipe (m)
Ar = (pi*D^2)/4; % Area of the pipe (m2)
Q = (2000*42*(3.785*10^-3))/(24*60*60); % Volumetric flowrate (m3/s)
e = 0; % Dimensionless surface roughness of pipe
```

Figure 2 - Part 2 of code

The input variables above is given in the question above, however conversion is required for output to be in the same units.

```
Re = (p*Q*D)/ (u*Ar); % Reynold's Number
x = (a + b)/2; % Calculates the midpoint of b and a.
f_x = -2*log10(((e/D)/3.7)+(2.51/(Re*x^0.5)))-(1/x^0.5) % Colebrook
equation
```

Figure 3 - Part 3 of code

Reynold's number is calculated from input variables. An equation for x is set for bisection to find midpoint of range between -5 and 5 set in figure 1. The Colebrook equation given in question is rearranged to make it equal 0 on one side. The code uses the x values calculated and plugs it in the f_x equation where an output is calculated.

```
if f_x < 0
a = x; % Reassigns a
elseif f_x > 0
b = x; % Reassigns b values
end
error = abs(f_x); %converts value to a magnitude to be used in calcs.
k=k+1; %number of iterations
end %output
fprintf('Iteration needed: %d, converged answer, x=%.4f, err=%.8f\n', k,
x, error);
```

Figure 4 - Part 4 of code

A loop is created in figure 4; if and else if statements are used to execute a group of statements. Figure 5 shows a better description on how the loop works.

Iteration needed: 21	converged answer	x=0.036997795104980	err = 0.00007824

Table 1

If we set the tolerance to equal 0:

Iteration needed: 59	converged answer	x=0.036998749246770	err = 0

Table 2

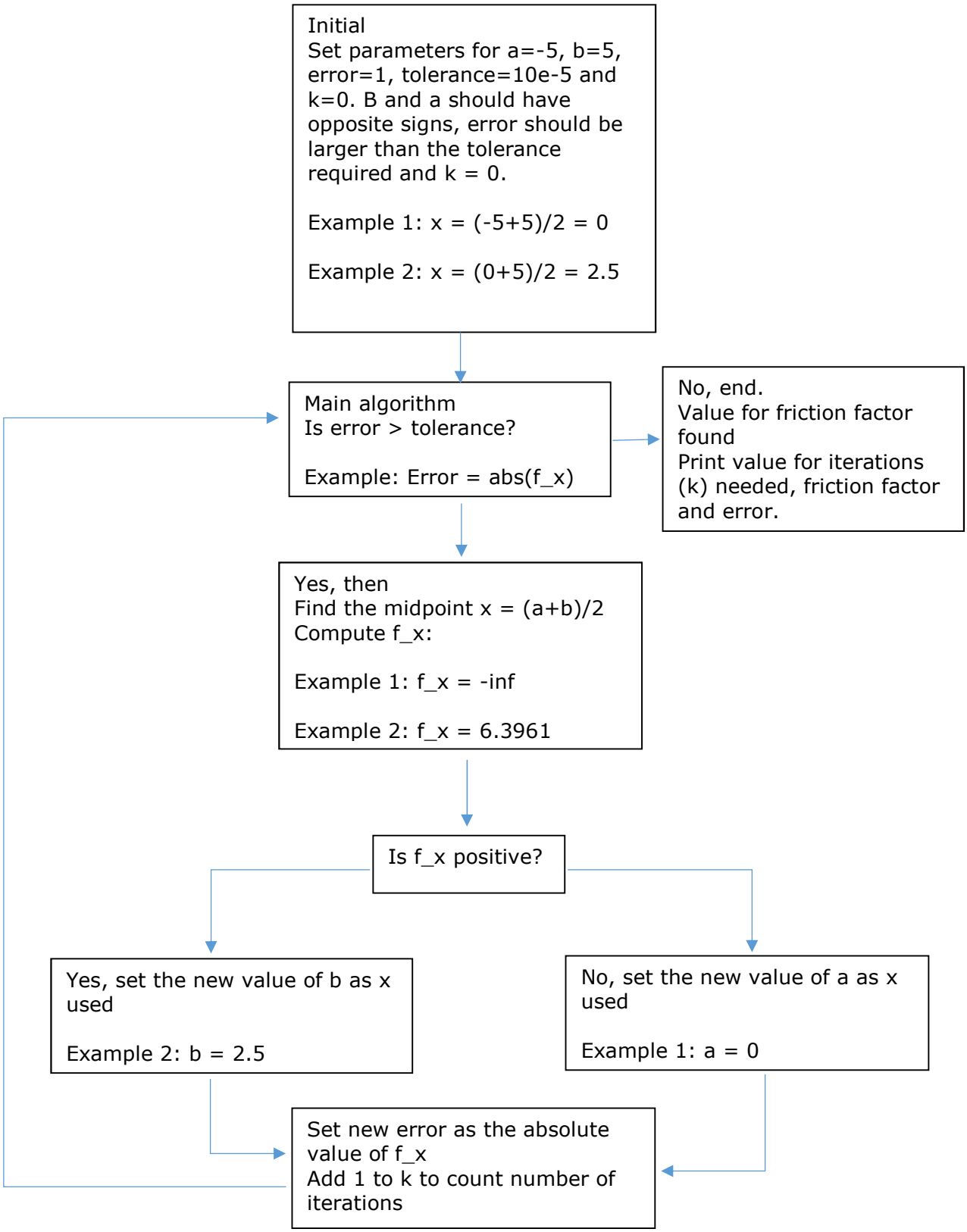

Figure 5 - Loop

Built in function, vpasolve:

```
clear all;
clc;

p = 0.9*1000; % density (kg/m3)
u = 8*0.001; % viscosity (Pa.s)
D = 4*0.0254; % Inner Diameter of pipe (m)
Ar = (pi*D^2)/4; % Area of the pipe (m2)
Q = (2000*42*(3.785*10^-3))/ (24*60*60); % Volumetric flowrate (m3/s)
e = 0; % Dimensionless surface roughness of pipe
Re = (p*Q*D)/ (u*Ar); % Reynold's Number
```

Figure 6 - part 1 of code

These are the same input variables as used in first part of question.

```
syms x
f_x = -2*log10(((e/D)/3.7)+(2.51/(Re*x^0.5)))-(1/x^0.5);
x = double(vpasolve(f_x==0,x,[0 inf]))
```

Figure 7 - Part 2 of code

x = 0.036998749246770

The x-value calculated from vpasolve has no errors therefore it is more accurate than the value calculated from the bisection method with a tolerance of 10e-5. When the tolerance of bisection is set to 0 we get the same x value output from figure 7; however more convergence is required, a total of 59 as shown in table 2 compared to the 21 at tolerance 10e-5.

There are many applications of turbulent flow in pipes generally it is used for mixing. This is extremely useful for industries such as the chemical industry where mixing is vital for reaction to occur, an example of this is a plug flow reactor (PFR); PFR are used in applications such as gold leaching of ores (SUVALAXMI SAHOO, 2014) . Another industry found useful is in sewage systems as the mixing properties caused by turbulence avoids waste build up and blockage in pipes (Swathmore, n.d.).

Part b)

Plot on the same graph friction factor vs. Re (5000 ≤ Re ≤ 100,000) for ε/D = 0, 0.002, 0.004, 0.006, 0.008. From these plots, provide comments on the effects of volumetric flowrate and surface roughness on the friction factor in turbulent flow.

```
clear all;
clc;
for i = 0: 0.002: 0.008 % i = e/D
Re = linspace (5000, 100000, 1000); %Array created with Reynolds number
error = linspace (1, 1, 1000); %Array created with error
tol = linspace (10e-10, 10e-10, 1000); %Array created with tolerance
a = linspace (0, 0, 1000); %Array created with a
b = linspace (1, 1, 1000); %Array created with b
k = 0;
```

Figure 8 - Part 1 of code

For part b) the similar loop is used in figure 5 but the difference is this time multiple values of variables is tested, recorded and then plot. This required the linspace function to create an array of matrix 1x1000 to allow the function to compute.

```
while error > tol % Loop ends when condition is violated
x = (a+b)/2; % Calculates the midpoint of b and a.
f_x = -1./sqrt(x)-(2.*log10((((i)/3.7)+(2.51./(Re.*sqrt(x))))));%
Colebrook equation
for j = 1:1000
if f_x(j) < 0
a(j) = x(j);
elseif f_x(j) > 0
b(j) = x(j);
end
error=abs(f_x(j));
end
k=k+1;
end
disp(x)
```

Figure 9 - Part 2 of code

Figure 9 provides statements set for a loop to output values until the error is less than the tolerance, the loop will terminate and print off.

```
hold on
plot(Re,x)
grid on
title ('friction')
xlabel('Re')
ylabel('friction factor')
legend ('0', '0.002', '0.004', '0.006', '0.008');
fprintf('Iteration needed: %d, converged answer, x=%.4f, err=%.8f\n', k,
x, error);
hold on
end
```

Figure 10 - part 3 of code

Figure 10 plots the friction factor vs Re graph found in figure 11.

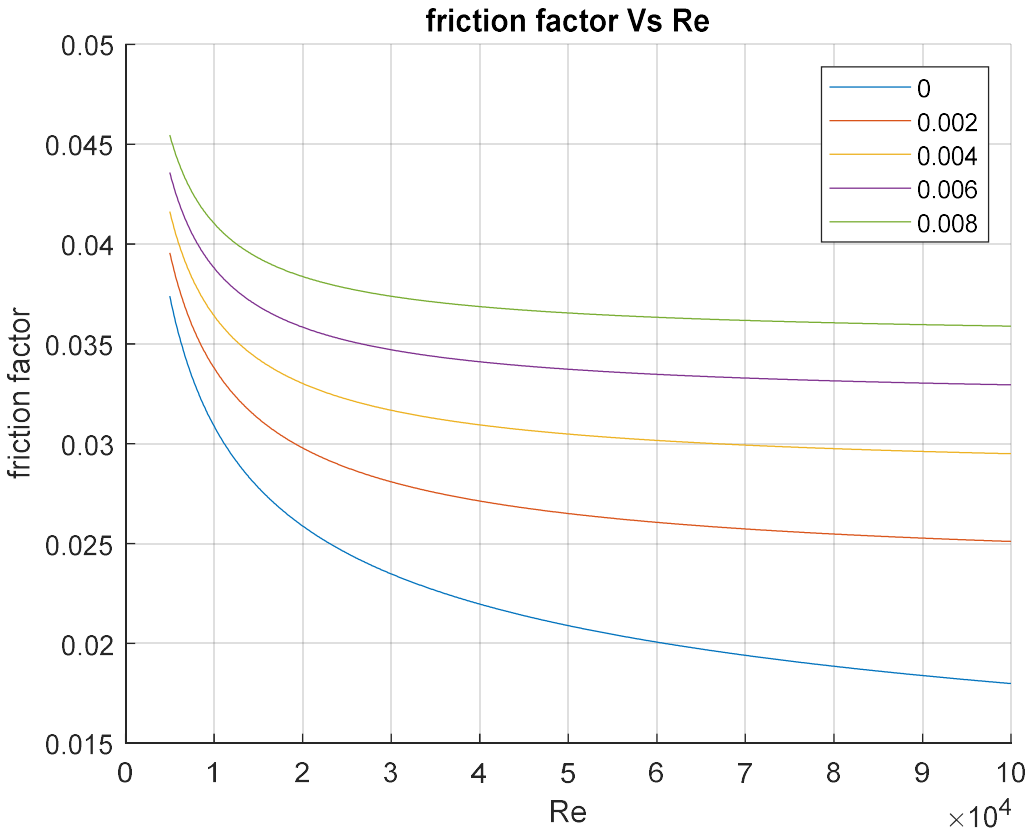

Figure 11 - Friction factor Vs Re

Figure 11 shows turbulent flow, it also describes a relationship as surface roughness increases friction factor Vs Re curve is shifted up which means increase in friction factor. The curve decrease slowly and then eventually plateaus. Re number is directly proportional to the volumetric flowrate.

Part c)

c . In the case of the flow being laminar (instead of turbulent), the velocity profile of an incompressible laminar flow is given by:

$$u(r) = u_{max}\left[1 - \left(\frac{r}{R}\right)^2\right],$$

where r is the distance from the center of the pipe. R is the radius of the pipe. u_{max} is the maximum velocity at the center of the pipe. Write a Matlab code to **determine the maximum velocity at the center of this pipe** using the diameter d, and average velocity $u_{avg} = Q/A$; then plot the velocity profile, *u(r)* of the flow. The velocity at the center, u_{max}, is related to the average velocity by the following equation:

$$u_{max} = \frac{u_{avg}\pi R^2}{\int_0^{2\pi}\int_0^R \left[1 - \left(\frac{r}{R}\right)^2\right] r dr d\theta}$$

Develop your own numerical integration code to solve it (hint: start with trapezoidal method).

$$\int_{x_i}^{x_{i+1}} f(x)dx = h\left(\frac{f(x_{i+1}) + f(x_i)}{2}\right) + O(h^3).$$

Equation 2 - Trapezium Rule (RHUL Dept. of Physics, n.d.)

A concave-down function yields an underestimate of a curve because area is unaccounted for under the curve, and when the integrand is concave up then the error is negative, and the trapezoidal rule overestimates the true value. If the interval of the integral being approximated includes an inflection point, the error is harder to identify.

```
clear all;
clc;

D = 4*0.0254; % Inner Diameter of pipe (m)
R = D/2;
Ar = (pi*D^2)/4; % Area of the pipe (m2)
Q = (2000*42*(3.785*10^-3))/ (24*60*60); % Volumetric flowrate (m3/s)
```

Figure 12 - Part 1 of code

These Variables are constant in the maximum velocity (u max) equation. The umax equation includes a double integral which needs to be dealt with to solve for u max.

```
a = 0;
b = R;
n = 100;
x = linspace (a, b, n);
f = x.*((1-((x)/R). ^2));
I_trap = trapz(x, f);
```

Figure 13 - Part 2 of code

The trapezium function is used to integrate the equation f above. The first integration takes place between a distance starting from centre of pipe diameter (0 m) to the wall of pipe at length R (0.0508 m) hence the limits given for a and b.

```
uavg = Q/Ar;
r=linspace(-0.0508,0.0508,100);
umax = ((((uavg)*pi*(R)^2)/ (I_trap*2*pi)));
velocity = umax*(1-(r/R).^2);

plot(velocity,r)
```

Figure 14 - Part 3 of code

The trapezium integral calculated from paragraph 2) I_trap is a variable input of the umax equation in 3); the integral of I_trap with respect to θ between 0 and 2*pi equals to f*2*pi hence the double integral is evaluated and is input in to umax equation

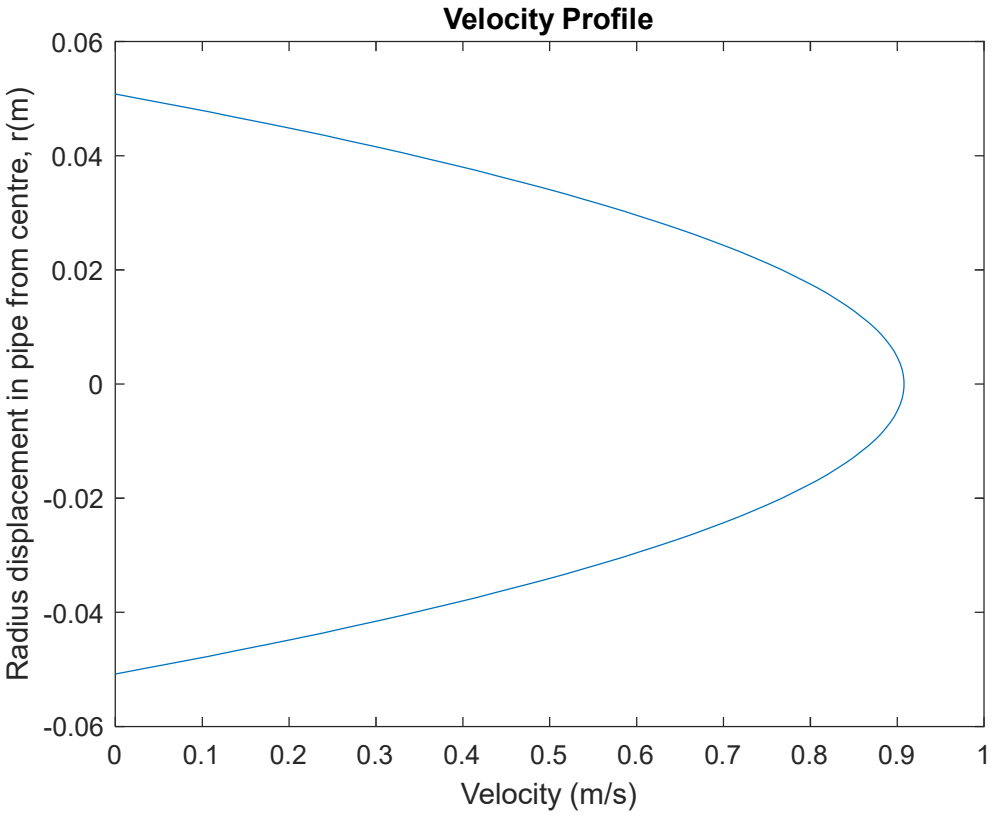

Figure 15 - Velocity Profile

As shown from figure 1 the maximum velocity is at 0 m experienced at the centre of the pipe, velocity decreases as r increases getting closer to the wall; at the wall there is no velocity of the flow so we can assume there is no slip at the wall. As evidently shown, this is a parabolic graph and therefore the average velocity experienced is close to half the maximum velocity.

Conclusion:

Numerical techniques have proven to be useful as it is able to model many applicational behaviours such as the fluid model described in this report. Matlab is a program which allows users to build their own numerical methods, this is useful because predictive models can be used to develop engineering applications such as plug flow reactors.

Reference

RHUL Dept. of Physics. (n.d.). *Trapezium Rule*. Retrieved from http://www.pp.rhul.ac.uk/~george/PH2150/html/node62.html

SUVALAXMI SAHOO, R. B. (2014). *DESIGN AND CONSTRUCTION OF BATCH,CSTR,TUBULAR AND PFR REACTER.* BHUBANESWAR: C.V. RAMAN COLLEGE OF ENGINEERING.

Swathmore. (n.d.). *REAL WORLD APPLICATIONS*. Retrieved from http://www.engin.swarthmore.edu/~dluong1/E41/Lab2/applications.htm