ME2543: SIMulation Methods for Mechanical Engineers

Exam #01 – Spring 2023

100 Points – OPEN BOOK, OPEN NOTES, MATLAB Documentation, Online sources – NO Chegg, CHATGPT etc

CLASS ID (4 digits): 1681 Print Name: Philip Melton

Signature: Philip Melton

*(I agree to abide by the instructions outlined below)*

Instructions:

1. **Only approved devices are to be used in this exam!**

2. DO NOT OPEN this exam until you are told to do so. The exam pages will be separated for grading; therefore, your *4-digit CLASS ID #* must appear at the top of every page you want graded. Use only your CLASS ID on exam pages as this preserves anonymity in grading. You will have exactly **60** **minutes (1 hour)** to complete this examination and you may ask questions about the exam **ONLY form me!**

3. Work only ***space allocated for that question***.

4. **Define** all *acronyms & abbreviations* the first time you use them in every problem. You are expected and encouraged to utilize your technical insight and intuition in the formulation and resolution of these questions. Remember, however, that others may not share your insights; therefore, briefly explain any unusual shortcuts or assumptions you use in solving the problems.

5. If you get stuck on any problem, **move on** to the next one and return to it when you have completed the remainder of the exam.

6. Make sure you add comments for your MATLAB script. Your code should not have any errors.

7. Write your answer right after the corresponding section. For example, if there is “MATLAB Script”, copy and paste your code right after that. Or, part a) ask for plot the data, place the figure right after that.

8. **IMPORTANT:** You may skip the introduction of each question. Suggestion: begin with question 2,3, and then question 1 part b and finally question 1 part a!

(1) Given (40pts): Solution of equations (Fall 20)

The Darcy-Weisbach equation is an empirical formula used in fluid dynamics that connects the loss of pressure in a length of pipe due to friction to the average fluid flow velocity for an incompressible fluid. This equation includes a non-dimensional friction factor known as the Darcy friction factor:

Where the pressure loss per unit length is a function of:

, the density of the fluid;

, the hydraulic diameter of the pipe;

, the mean flow velocity;

, the Darcy friction factor.

For laminar flows in a circular pipe with diameter , the friction factor varies inversely with the Reynolds number (), which can be calculated using readily available physical measurements or published data. Then, it can be simplified to following equation:

is the dynamic viscosity of the fluid;

Q is the volumetric flow rate (.

**Find:**

Suppose 913 kPa will cause glycerin at 20 deg C to flow at 4.67 L/s. The dynamic viscosity of glycerin

at this temperature is 1.49 [kg/m.s].

In order to solve for pipe dimension (the length and diameter of the pipe) which cause the glycerin has laminar flow under the condition mentioned above and we have design restriction that the surface area of the pipe should be exactly equal to . Which means if we solve a system of nonlinear equations, as shown below:

We will have the specification of the pipe that satisfies the constraints we mentioned for laminar flow.

**Part a)** below the code to find the dimensions (x = [L, D]) is provided to you. I developed a Newton-Raphson Method to solve these nonlinear equations. **BUT**, you may encounter few errors when you try to execute this code by copying and pasting it on your PC’s MATLAB. I want you to debug it and make sure this code works without any errors! Edit the script in below using **RED COLOR!**

It may help us to debug the code, if we first review the Newton-Raphson method:

That’s how we update our x value and try to converge to the solution of equation. In this case that

we are dealing with a set of equations is actually the Jacobian of which means

would be a 2 by 2 matrix because we have two variables (L, D) and two equations. Therefore,

, the first equation… , the second equation…

We can write

**Hint:** There are one major error and one warning that should be fixed… you may place a breakpoint at line 50 (end of the while loop) to pause the code at the end of each iteration… hover the mouse on variable to see if the algorithm is converging or not and compare it with fsolve result!

MATLAB script:

%Program Newton-Raphson

% Solves for the root of a vector function f(x) using an initial guess

% xinit using the Newton-Raphson method. Requires being able to evaluate

% f(x) and the Jacobian J(x) = [df1/dx1, df1/dx2, ...; df2/dx1, ...].

% The function f and the variable x are vectors of length N.

%

% Contains functions:

% >myf vector function f(x)

% >myJ matrix function J(x)

%

% Newton's method in N variables:

% (f(x) - 0) ~= J \* (x - xroot)

% => xnew = xold - inv(J)\*f(xold)

% Iterate until converged: ||xnew - xold|| / ||xnew|| <= tol

clear; clc; close all

%Define initial guess, x0

xnew = [10; 0.01]; %initial guess in meters

%Define tolerance (convergence criterion)

tol = 1e-8; %Converged when x changes by less than 10^-6 percent

%Initialise iteration counter

niter = 0;

relerr = 1

while (relerr > tol) %Loop until the residual error is <= the tolerance

niter = niter + 1; %Keep track of number of iterations

xold = xnew; %Store the existing value in xold

%Evaluate f(xold) and J(xold)

fold = myf(xold);

J = myJ(xold);

%Get updated estimate

%from part a

xnew = xold-inv(J)/fold

%Calculate relative error in convergence (estimated by the L2 norm of

% the change in x over the L2 norm of x)

relerr = norm(xnew - xold) / norm(xnew);

% %Debugging messages

% fprintf('Iter %6i: xnew = [%10.2e; %10.2e], f = [%10.2e; %10.2e]\n',...

% niter,xnew,fold)

% %End debugging messages

end

%Print results to the screen

fprintf('Newton-Raphson example\n') %\n jumps to the next line

fprintf('Solution converged in %i iterations.\n',niter)

fprintf('Root found:\n')

disp(xnew)

res = myf(xnew); %Calculate residual

fprintf('Function (residual):\n')

disp(res)

%===========================================================

function [f] = myf(x)

%Function myf(x)

% Returns the vector f to solve for pipe dimensions x = [L; D]

%

% Problem statement: Find pipe dimensions x = [L; D] such that:

% 913 kPa will cause glycerin at 20 deg C to flow at 4.67 L/s

% Darcy-Weisbach equation, laminar flow:

% (0.2835 N-m)\*L/D^4 - 913000 Pa = 0

% The surface area of the pipe is 3 m^2

% pi\*L\*D - 3 m^2 = 0

f = [0.2835 \* x(1) / (x(2)^4) - 913000;

pi\*x(1)\*x(2) - 3];

end %function myf

%===========================================================

function [J] = myJ(x)

%Function myJ(x)

% Returns the Jacobian associated with the function myf(x)

% x(1)=L and x(2)=D

%Pre-allocation not necessary, but saves time and avoids warning messages

J = zeros(2,2); %Pre-allocate memory for 2x2 matrix

J(1,1) = 0.2854 / (x(2)^4); %df1/dx1

J(1,2) = -4\*0.2854\*x(1) / (x(2)^5); %df1/dx2

J(2,1) = pi\*x(2); %df2/dx1

J(2,2) = pi\*x(1); %df2/dx2

end %function myJ

%= = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

%DEBUGGING TOOLS:

% F12 or click on "-" Set/clear breakpoint

% Code will pause at these locations

% F10 or Step Run next line

% F11 or Step In Jump into function code

% Ctrl+C Stop execution

% Useful for stopping infinite loops

% Hover over variable name Show current value of variable

**Part B)** Now use MATLAB command fsolve with the same initial conditions you picked for previous part and solve for dimension of the pipe. Please note that the answer necessarily should be similar as different set of length and diameter may cause laminar flow!

**Hint:** the function for set of nonlinear equations is already given to you, “myf”! check the first example in MATLAB documentation for fsolve…

(2) Given (30 pts): Interpolation (Fall 20)

An IR sensor is an electronic device that detects and measures infrared radiation in its environment, emitted by all objects with a temperature above absolute zero. The sensor utilizes different technologies like thermopiles, pyroelectric detectors, and bolometers for detecting IR radiation. IR sensors have various applications, such as measuring temperature, detecting motion or objects, and communication. They are used in devices like thermometers, thermal imaging cameras, home security systems, remote controls, and communication systems for short-range data transmission.

A light-emitting diode (LED) is a semiconductor device that emits optical radiation when an electric current passes through it. Now if we attach a LED to a mobile robot and install the IR sensor to a wall, that means I can measure the intensity of the optical radiation with the sensor and map voltage reading to distance values.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V\_Output (mV) | 0 | 42.6187 | 57.5646 | 66.8537 | 73.6110 | 78.9250 | 83.3051 | 87.0310 |
| Distance (cm) | 1 | 5.5 | 10 | 14.5 | 19 | 23.5 | 28 | 32.5 |
| V\_Output (mV) | 90.2729 | 93.1423 |  |  |  |  |  |  |
| Distance (cm) | 37 | 41.5 |  |  |  |  |  |  |

**Find:**

What is the relative distance between wall and mobile robot if the voltage readings of the sensor are 81.2 and 59.74.

|  |  |  |
| --- | --- | --- |
| V\_Output (mV) | 81.2000 | 59.7400 |
| Distance (cm) | 25.7419 | 10.9160 |

% creating matrices for output and distance

V\_out = [0 42.6187 57.5646 66.8537 73.6610 78.9250 83.3051 87.0310 90.2729 93.1423]

D = [1 5.5 10 14.5 19 23.5 28 32.5 37 41.5]

%interpolation funcion

interp1(V\_out,D,[81.2 59.74], 'spline')

(3) Given (30 pts): Numeric Integration (Fall 20)

An IMU is an electronic instrument that combines accelerometers, gyroscopes, and, in some cases, magnetometers to calculate and relay information on a body's specific force, angular rate, and occasionally its orientation. IMUs play a crucial role in guiding and controlling unmanned systems like UAVs, UGVs, and UUVs. They are indispensable components in this process. In a navigation system, the data reported by the IMU is fed into a processor which calculates altitude, velocity and position. Suppose a drone fly vertically only and the IMU records acceleration of the drone as below:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | 2 | 1.9950 | 1.9801 | 1.9553 | 1.9211 | 1.8776 | 1.8253 | 1.7648 | 1.6967 |
| T (s) | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 |
| a | 1.6216 | 1.5403 | 1.4536 | 1.3624 | 1.2675 | 1.1700 | 1.0707 | 0.9708 | 0.8712 |
| T (s) | 0.9 | 1 | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 1.6 | 1.7 |
| a | 0.7728 |  |  |  |  |  |  |  |  |
| T (s) | 1.8 |  |  |  |  |  |  |  |  |

**Find:**

**Part a)** find the vertical velocity of the drone using MATLAB. The Drone was stationary initially!

**Hint:** check MATLAB documentation for command “trapz(X,Y)” or check out the numerical methods we discussed during the class. To find the velocity of the drone at each time instant you need to integrate the acceleration from beginning to that specific time instant. For example, if you are trying to find the velocity at time you should calculate .

**MATLAB Script:**

%creating matrices of acceleration and time

a = [2 1.995 1.9801 1.9553 1.9211 1.8776 1.82523 1.7648 1.6967 1.6216 1.5403 1.4536 1.3624 1.2675 1.1700 1.0707 0.9708 0.8712 0.7728];

T = 0:.1:1.8;

% trapz(time,acceleration) = velocity approximation

% creating a for loop to create a matrix V that displays the velocity at

% every time

for i=1:1:18

V(i)=trapz(T(1:i),a(1:i));

end

V

**Part b)** Plot the numerical results vs analytical solution. Note that the plot should contain a legend and title, X- and Y- axis should have label. Suppose the analytical expression for the velocity is .

f=@(x) sin(x)+x

fplot(f,[0 1.8],'Color','b')

hold on

plot(V,T(2:end),'Color','k')

legend('f(x)','trapezoidal integration')

**Part c)** In one line discuss the reason behind the error and how to minimize it.

% this error is possibly due to the large difference (0.1) between values.

% It can be minimzed through taking readings every 0.01 seconds instead