ChE 310 **Problem Set 2 (15 pts)** **Due Wed 9/11/19**

Collect all m-files in a single .zip file and upload the .zip file to the course webpage by midnight on Wednesday, September 11, 2019. Please note any collaborations in the comments. Each student must upload their own individual copy of the work.

2.1 (3 pts) Evaluate,

$$e^x + 2e^x + 3e^x + 4e^x + 5e^x$$

using nested for and while loops. You can use the exponential function in MATLAB to check your answer, not to solve. Instead approximate the exponentials as infinite loops. Hint: The double summation below, equation 2.1.1, approximates the above expression (the infinite series is the Maclaurin series for e^x, see equation 2.1.2). You don't want your computer to run for an infinite time, so where applicable use a tolerance of 10^{-5} (in other words, the difference between the solution from the previous iteration and the current iteration should be less than this number…*i.e.* the solution does not change much for more iterations!). Save your solution as a script file PS2_1.m, and print only the final result of the above expression to the command line.

$$\sum_{k=1}^{5} k \sum_{n=0}^{\infty} \frac{x^n}{n!} \qquad (2.1.1)$$

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} = \sum_{n=0}^{\infty} \frac{x^n}{n!} \qquad (2.1.2)$$

Solution:

```
x = 5; %specify x as value that series is evaluated around
n_sum = 0; %Initialize the inner loop summation variable
tol = 10^-5; %set desired tolerance

for n = 1:5
    k_sum = 0;
    err = tol + 1;
    k = 0;
    while err > tol
        k_sum_last = k_sum;
        k_sum = k_sum + x^k/factorial(k);
        k = k + 1;
        err = abs(k_sum - k_sum_last)/k_sum;
    end
    n_sum = n_sum + n*k_sum;
end
disp(n_sum);
```

Output:

```
>> PS2_1_ARC

  2.2262e+03
```

2.2 (3 pts) A *piecewise function* is useful when the relationship between dependent and independent variables cannot be adequately represented by a single equation. The velocity $v(t)$ of a rocket might be described by:

$$v(t) = \begin{cases} 10t^2 - 5t & 0 \le t \le 8 \\ 624 - 3t & 8 < t \le 16 \\ 36t + 12(t-16)^2 & 16 < t \le 26 \\ 2136e^{-0.1(t-26)} & t > 26 \\ 0 & \text{otherwise} \end{cases}$$

where t is time in seconds and v is velocity in m/s. Develop an M-file function to compute v as a function of t, and then create a script file PS2_2.m to plot v versus t for $t = -5$ to 50. Suppress all output from the code, except for the final plot. Ensure your plot is properly labeled.

Solution:

rocketvel.m

```
function v = rocketvel(t)
if t <0
    v = 0;
elseif t<= 8
    v = 10*t.^2-5*t;
elseif t <= 16
    v = 624-3*t;
elseif t <= 26
    v = 36*t + 12*(t-16).^2;
else
    v = 2136*exp(-0.1*(t-26));
end
```

PS2_2.m

```
% Coded by Nigel F. Reuel on 9.2.2016
% This code solves problem 2
t = linspace(-5,50,1000);
v = zeros(1000,1);
% for loop to generate values for plotting.
for i = 1:1000
    v(i,1) = rocketvel(t(i));
end
plot(t,v);
xlabel('t, s')
ylabel('v, m/s')
```

Output:

```
>> PS2_2
```

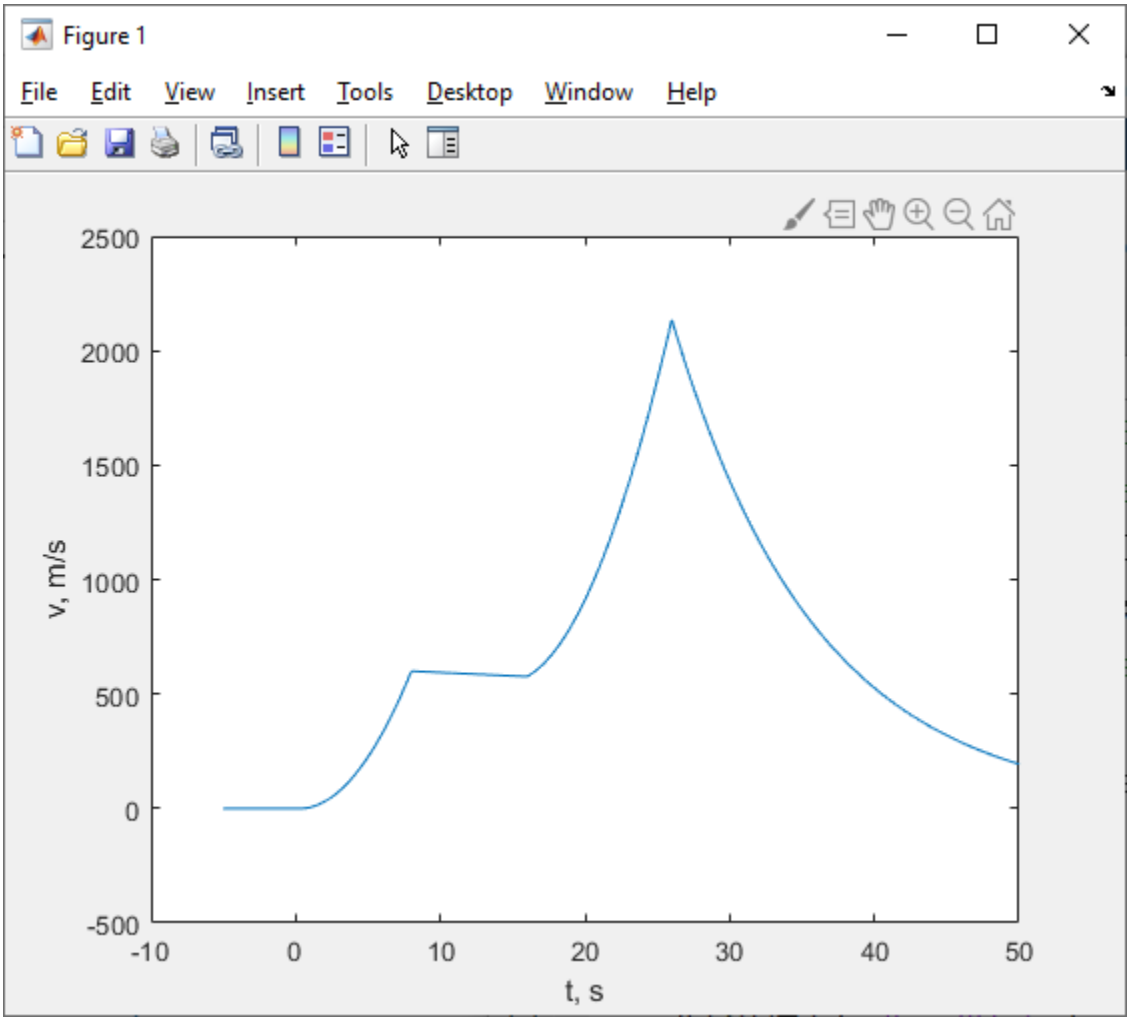


Figure 1
File Edit View Insert Tools Desktop Window Help
2500
2000
1500
1000
500
0
-500
v, m/s
-10
0
10
20
30
40
50
t, s

2.3 (4 pts)

(A) - You have opened a consulting firm that specializes in chemical engineering insight for agriculture applications. A client comes to you seeking help in determining flow rates through an open, rectangular, passive irrigation channel to his field. Thankfully you kept your ChE 310 Textbook and notice that Problem 3.9 details Manning's equation which can be used for this solution. You visit the client's field and make the following measurements:

- slope = 0.01 (rise over run)
- width = 2m
- depth = 0.5m
- roughness coefficient = 0.03

Create a MATLAB *function* that calculates the velocity of water through this channel (output) given the above inputs (slope, width, depth, roughness).

Solution:

culvert_vel.m

```
function U = culvert_vel(S,B,H,n)
U = S^(1/2)/n*(B*H/(B+2*H))^(2/3);
end
```

Pset2_3_parta.m

```
clear

S = 0.01;
B = 2;
H = 0.5;
n = 0.03;

U = culvert_vel(S,B,H,n);
fprintf('The velocity of water in this channel is %4.2fm/s\n',U)
```

Output

```
>> Pset2_3_parta

The velocity of water in this channel is 1.60m/s
```

(B) - News quickly spreads of your firm's expertise in culvert calculations. You open your email one day to find an invitation from the UAE to design all of their state funded culverts. They attach the following Excel file with all their measurements – 'culverts.xls' (Column 1-4 are roughness, slope, width, and depth respectively). They are offering a nice payment if you can get calculations back to their engineers by this afternoon. Make a new M-file PS2_3bc.m that imports the data from the Excel sheet, uses the function in part (A) to help calculate the

volumetric flow rate (NOTE: not velocity), and then exports the calculated flow rate vector as a new Excel file.

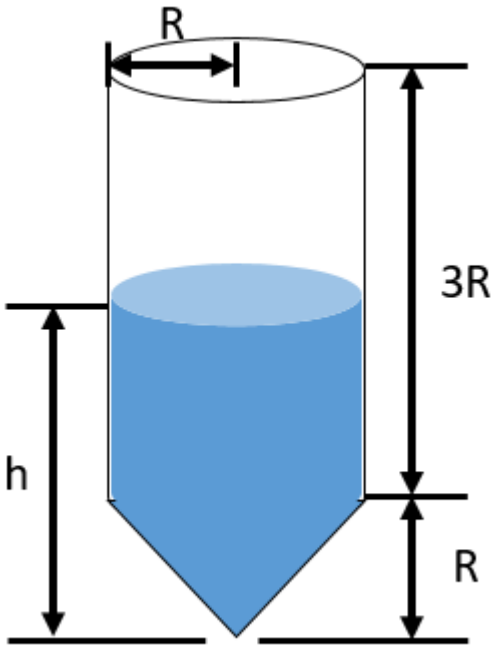
(C) - Adding on to PS2_3bc.m, use MATLAB to count the number of channels with flow rates greater than 100 m^3/s, which we consider to be poorly designed. What percentage of the designs fall in this poor category?

```
% Coded by Nigel F. Reuel on 9.1.2016
% This function solves problem 4 on the second PSet.
%
clear
Data = xlsread('culverts.xls');

%
% Determine how many data points there are in the file
N = length(Data);
% Preallocate memory for the solution vector
Q_vec = zeros(N,1);
for i = 1:N
    n = Data(i,1);
    S = Data(i,2);
    B = Data(i,3);
    H = Data(i,4);
    Velocity = culvert_vel(S,B,H,n);
    % Convert flow velocity to flow rate (Q - m^3/s)
    Q = Velocity*B*H;
    Q_vec(i,1) = Q;
end
% Save the data to an excel file
xlswrite('FlowRates.xls',Q_vec)
% End of PART B
% -------------
% Start of Part C
BinaryVec = Q_vec>100;
Count = sum(BinaryVec);
PercentPoor = Count/length(Q_vec)*100;
fprintf('There are %.1f percent of the culverts in the poorly designed
category.\n',PercentPoor)
```

Output:

```
>> Pset2_3_partb_and_c

There are 46.1 percent of the culverts in the poorly designed
category.
```

2.4 (5 pts) A cylindrical storage tank has a conical bottom to facilitate drainage. One particular tank design specifies a radius of R, a total height of $4R$, and the height of the conical bottom to be R. Write an m-file function `tank_volume.m` that calculates the volume of liquid in such a tank if the liquid level is h.



***Function*:** `tank_volume.m`

Inputs:

- `R`: an *array* of R values representing the radius of the tank.
- `h`: an *array* of liquid height values measured from the bottom of the tank

Outputs:

- `V`: an array of tank volumes, of the same shape as the input data `R` and `d`

Your function should have two checks:

- Issue an error statement if the arrays R and h are not the same shape
- Issue an error if, for any value of h, $h > 4R$ (tank overflow!). Hint: typing "doc any" at the command line will show a simple suggestion for testing all values in an array simultaneously. There are other ways to test this, as well.

Driver Script: PS2_4.m

Create a simple script that uses your tank_volume function to generate a plot (on a single axis) of the liquid volume vs. liquid level for $R = 1$ m and $R = 2$ m. Your plot should be properly labeled (axes, title, legend). No output from the function file or the driver file should be displayed other than the plot when the driver file is executed.

Hint: The volume of a cone with radius r and length l is $\frac{\pi r^2 l}{3}$.

Solution

tank_volume.m

```
function V = tank_volume(R, h)

%Ensure that the requested heights are within the tank's limits
if any(h > 4.*R )
    error('All h values must be less than 4R, or tank will overflow!');
end

%Check that R and d have the same shape
%Technically, this would be caught by the "any" statement above, if that is
%used. Otherwise, the following check will suffice assuming R and h
%are 1D or 2D arrays.
szR = size(R); szh = size(h);
if szR(1) ~= szh(1) || szR(2) ~= szh(2)
    error('R and h must have the same shape');
end

%Initialize the output: Should have same size as input
V = zeros(size(R));

%Assuming R/h are a 1D array, a single loop will work just fine.
%If R/h are 2D, then we would need nested loops.
for ii = 1:length(R)
    %If the level h is higher than the cone, then the level
    %in the cone is R. That must be corrected if d is less than R.
    cone_height = R(ii);
    if(h(ii) < cone_height )
        cone_height = h(ii);
    end
    %calculate volume of liquid in the cone
    V_cone = pi .* cone_height.^3./3;
    %Next, calculate the level in the cylindrical portion of the tank
    cylinder_height = h(ii) - cone_height;
    %calculate volume of liquid in the cylinder, if any
    if cylinder_height > 0
        V_cylinder = pi .* cylinder_height .* R(ii).^2 ;
    else
        V_cylinder = 0.;
    end
    V(ii) = V_cylinder + V_cone;
end
```

PS2_4.m

```
%PS2_4.m
clear
%option 1: use fplot and function handles to make plots
%This is nice, though some subtle aspects of how we've
%defined our R variable in tank_volume will cause MATLAB
%to throw some warnings as this runs.
%Note that we can plot heights up to 4 * R.
fplot( @(h) tank_volume(1,h),[0 4],'r');
hold on ;
fplot( @(h) tank_volume(2,h),[0 8],'k');

%(reset for option 2)
clear
hold off

%option 2 Store values in arrays (more straightforward)
h1 = linspace(0,4); R1 = ones(size(h1)); %Input h and R arrays
V1 = tank_volume(R1,h1); %Calculate output volume
plot(h1,V1,'r');
hold on
h2 = linspace(0,8); R2 = 2*ones(size(h2));
V2 = tank_volume(R2,h2);
plot(h2,V2,'k');

%Regardless of data generation method, need to label plot:
xlabel('Liquid height, m');
ylabel('Liquid volume, m^3');
title('Retained volume in tanks with conical bottoms');
legend('R=1m','R=2m');
```