

## CIVE 1331 Computing for Engineers

Purpose: MatLab and FORTRAN Practice Examination

Problem –1 FORTRAN .....	2
Problem –2 Linear Systems (use MatLab).....	6
Problem 3 Integration (MatLab) .....	9
Problem 4 Ordinary Differential Equations – Use MatLab .....	11

## CIVE 1331 Computing for Engineers

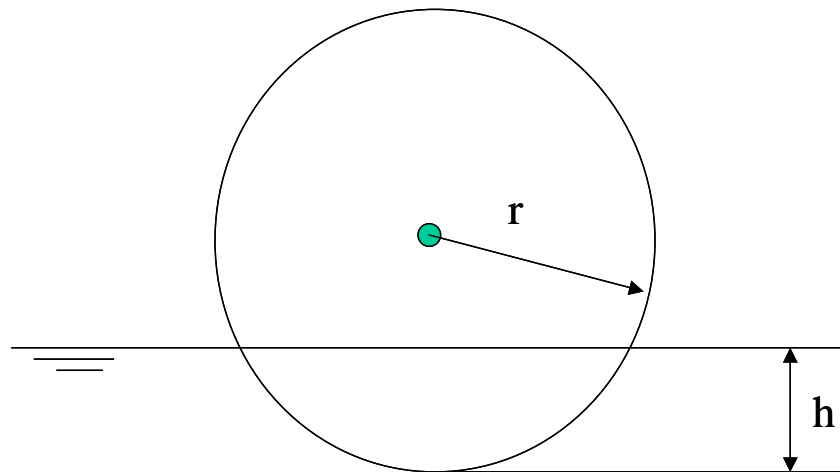
**Problem –1 FORTRAN**

Use your FORTRAN root finding program to determine the depth a sphere of specific weight 0.6 (specific weight of water is 1) sinks into water as a function of its radius. The

weight of the sphere as a function of its radius is  $weight = \gamma \frac{4}{3} \pi r^3$  where the specific

weight is denoted by gamma. The volume of a spherical segment (see figure) is

$$volume = \frac{1}{3} \pi (3rh^2 - h^3)$$



**Figure 1. Bouyant Sphere in Water**

The force balance is that the weight of water displaced is equal to the weight of the floating object. To compute the forces, multiply the specific weight of water and the volume displaced and set this value equal to the weight of the sphere, then solve for h.

In your program, you want to rewrite your function as

$$f(h) = \gamma \frac{4}{3} \pi r^3 - \frac{1}{3} \pi (3rh^2 - h^3)$$

and solve for  $f(h) = 0$ , using a realistic initial guess. Verify your numerical result using the MatLab symbolic solver on the actual equation, then substitute the numerical values and see how they compare with your FORTRAN result.

## CIVE 1331 Computing for Engineers

```

      program findroot
c
c allocate program memory
c
      real*8 guess,stepsize,root
c
c allocate function names and tell compiler they are external
c to the findroot module
c note: necessary on most compilers, appears to be optional on the
c       visual fortran
c
      real*8 func,dfdx
      external func,dfdx
c
c prompt user for an initial guess and ths stepsize value
c
      write(*,*)'root finder program '
      write(*,*)'enter initial guess'
      read(*,*)guess
      write(*,*)'enter a stepsize, if in doubt, enter 1e-6'
      read(*,*)stepsize
c
c we will limit our program to 20 updates
c
      do 1001 it=1,20,1
          guess=guess - func(guess)/dfdx(guess,stepsize)
c
c test for rootness
c
          if (abs(func(guess)) .le. 1.e-06) then
              write(*,*)'root found within tolerance'
              root=guess
              write(*,*)'      root      function value'
              write(*,9001)root,func(root)
              go to 1002
          else
              end if
      1001 continue
c
c get to next lines only if we fail to find a root
c
      write(*,*)'cannot find root after 20 trials -- last guess is'
      write(*,*)' guess = ',guess,' function value = ',func(guess)
c
c next line ends execution - get here from either the rootness test or above
c
      1002 stop
      9001 format(1x,g12.5,2x,g12.5)
      end

      function dfdx(x,dx)
c
c finite-difference approximation for derivative of function
c named "func"
c
      real*8 x,dx,func,dfdx
      external func
      dfdx= ( func(x+dx) - func(x) ) / dx
      return
      end

c function for sphere submerge ONLY THIS MODULE CHANGES!!!

      function func(h)
c

```

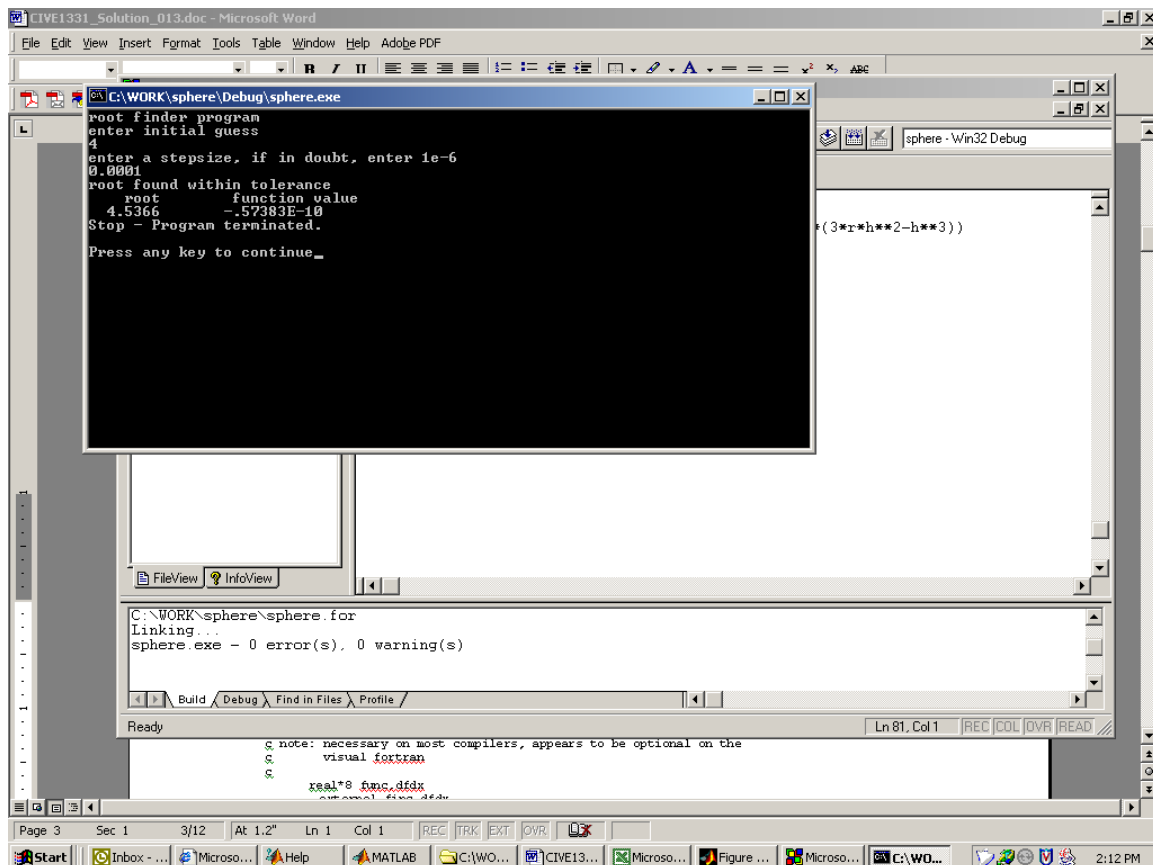
## CIVE 1331 Computing for Engineers

```

c we will hard-code the sphere sp. gravity and radius, change and
c recompile as needed. Since we have to change the function anyway,
c this unstructured coding is acceptable
c
      real*8 func,pie,h,r,gamma
      pie=acos(-1.)
      gamma=0.6
      r=4.0

c
c pick meaningful r
c
      func=gamma*(4./3.)*pie*r**3 - ((1./3.)*pie*(3*r*h**2-h**3))
      return
      end

```

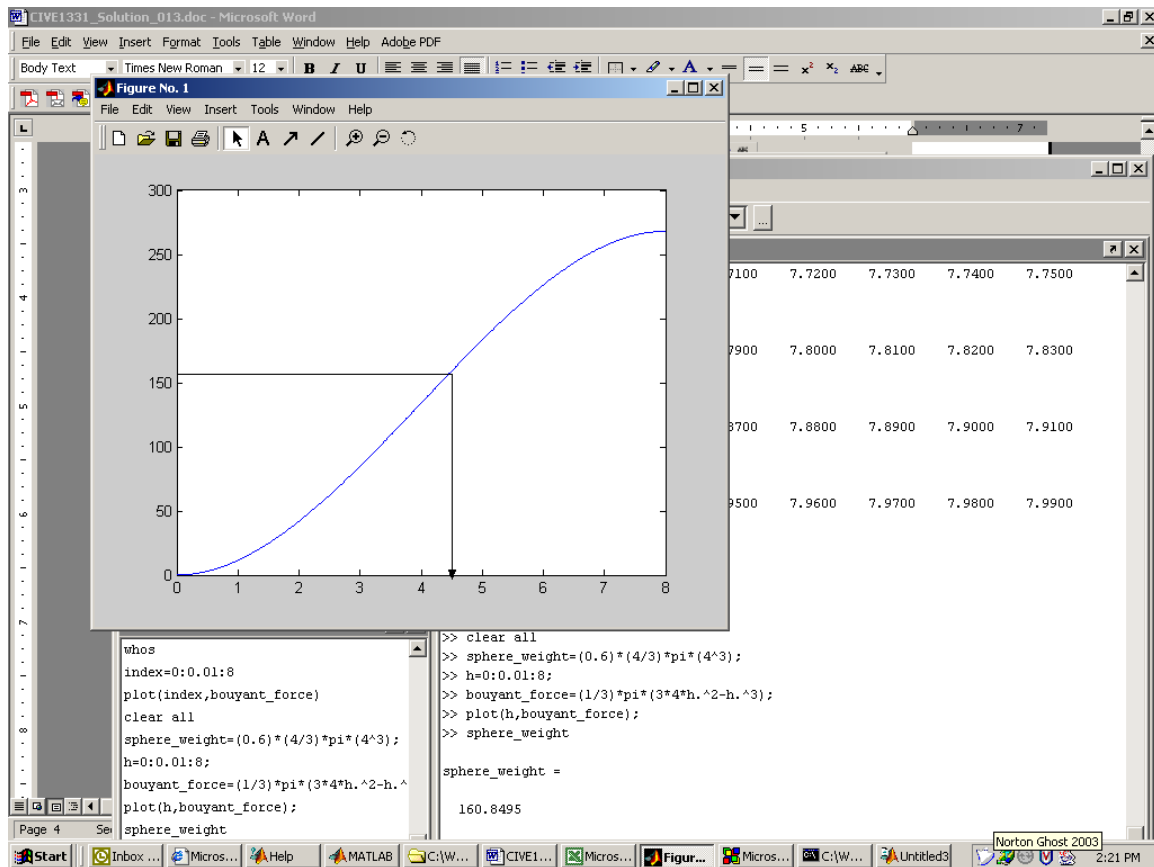


That was the fortran solution. To verify in MatLab easiest way is to:

- Step 1: Compute the sphere weight for a given radius (we used 4) in the example.
- Step 2: compute buoyant force for a lot of h values, pick value closest to sphere weight.
- Step 3: See if results are consistent with FORTRAN solution.

H:\Courses\CIVE\_1331\CE1331F2003\CIVE\_1331\CIVE1331\_ABET\_Format\LabExercises\Lab013\CIVE1331\_Solution\_013.doc

## CIVE 1331 Computing for Engineers



Mat lab shows that  $h$  is somewhere between 4 and 5, FORTRAN picked 4.53, so results are reasonable.

## CIVE 1331 Computing for Engineers

**Problem –2 Linear Systems (use MatLab)**

Use the linear algebra method and the polyval() function to analyze the following experimental results of fuel mileage versus vehicle weight.

Weight (lbs)	Mileage (mpg)	Weight (lbs)	Mileage (mpg)
2775	33	3325	20
2495	27	3200	21
2405	29	3450	19
2545	28	3515	21
2270	34	3495	19
2560	24	4010	19
3050	23	4205	17
3710	24	2900	24
3085	23	2555	28
2940	21	2790	21
2395	26	2190	34

From these data determine the linear equation that relates fuel mileage to vehicle weight.

Based on your analysis how well are the data represented by a straight line?

Step 1: Capture the table and save as two column ASCII file (use Excel)

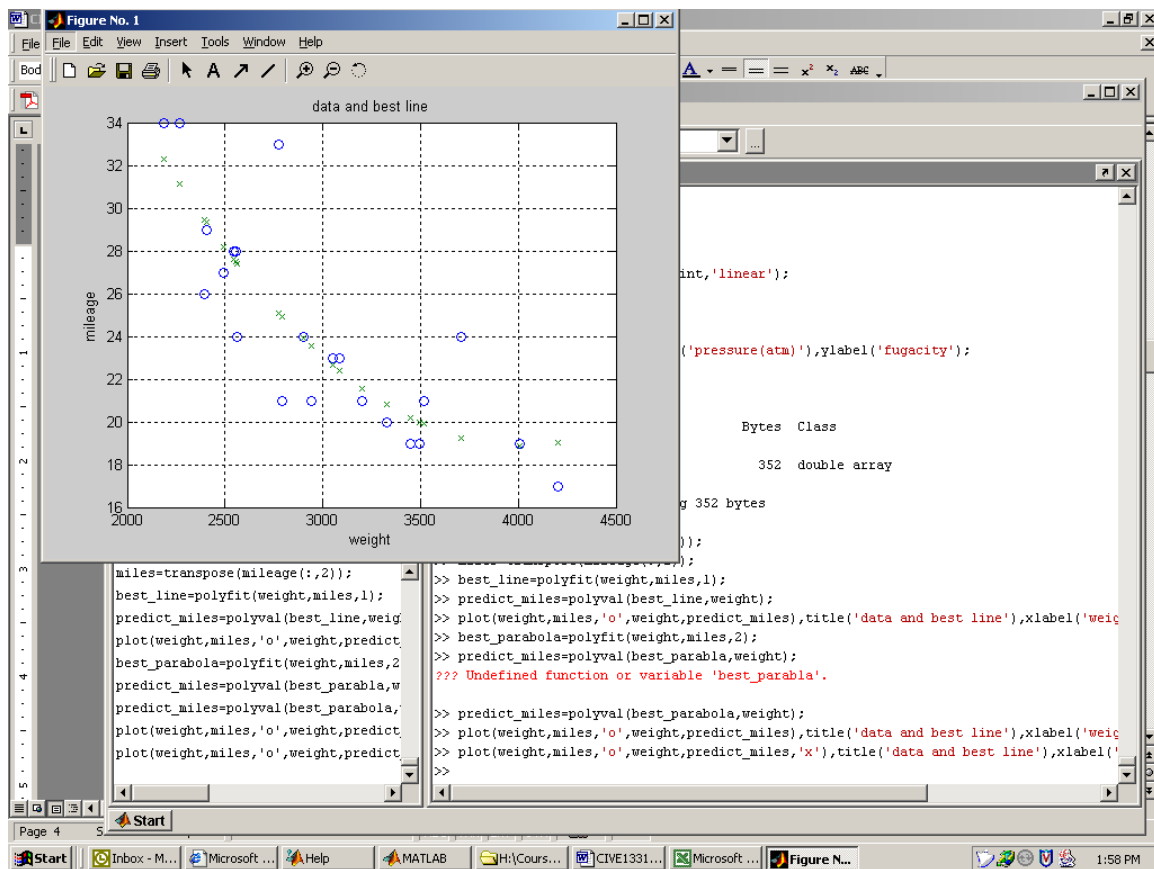
Step 2: Load in the file, extract the columns and convert into rows

Step 3: Use the curve fitting tools and find the equation.

Step 4: Try polynomial other than linear, see which is better.

H:\Courses\CIVE\_1331\CE1331F2003\CIVE\_1331\CIVE1331\_ABET\_Format\LabExercises\Lab013\CIVE1331\_Solution\_013.doc

## CIVE 1331 Computing for Engineers





## CIVE 1331 Computing for Engineers

**Problem 3 Integration (MatLab)**

Fugacity is a term used in engineering to describe the available work from an isothermal process. For an ideal gas, the fugacity  $f$  is equal to its pressure  $P$ , but for real gasses

$$\ln\left(\frac{f}{P}\right) = \int_0^P \frac{C-1}{P} dP$$

Where  $C$  is the experimentally determined compressibility factor. For methane, values of  $C$  are tabulated as

P(atm)	C
1	0.9940
10	0.9370
20	0.8683
40	0.7043
60	0.4515
80	0.3429
120	0.4259
160	0.5252
250	0.7468
400	1.0980

Develop a script that calculates  $f$  given the tabulation for all values of  $P$  between 1 and 400 atmospheres in 2 atm increments. Plot your result of  $f$  versus pressure. Assume that the value  $C$  varies linearly between the tabulated values.

Step 1 Produce two vectors  $P$ , and  $C$  that interpolate the tabulated values.

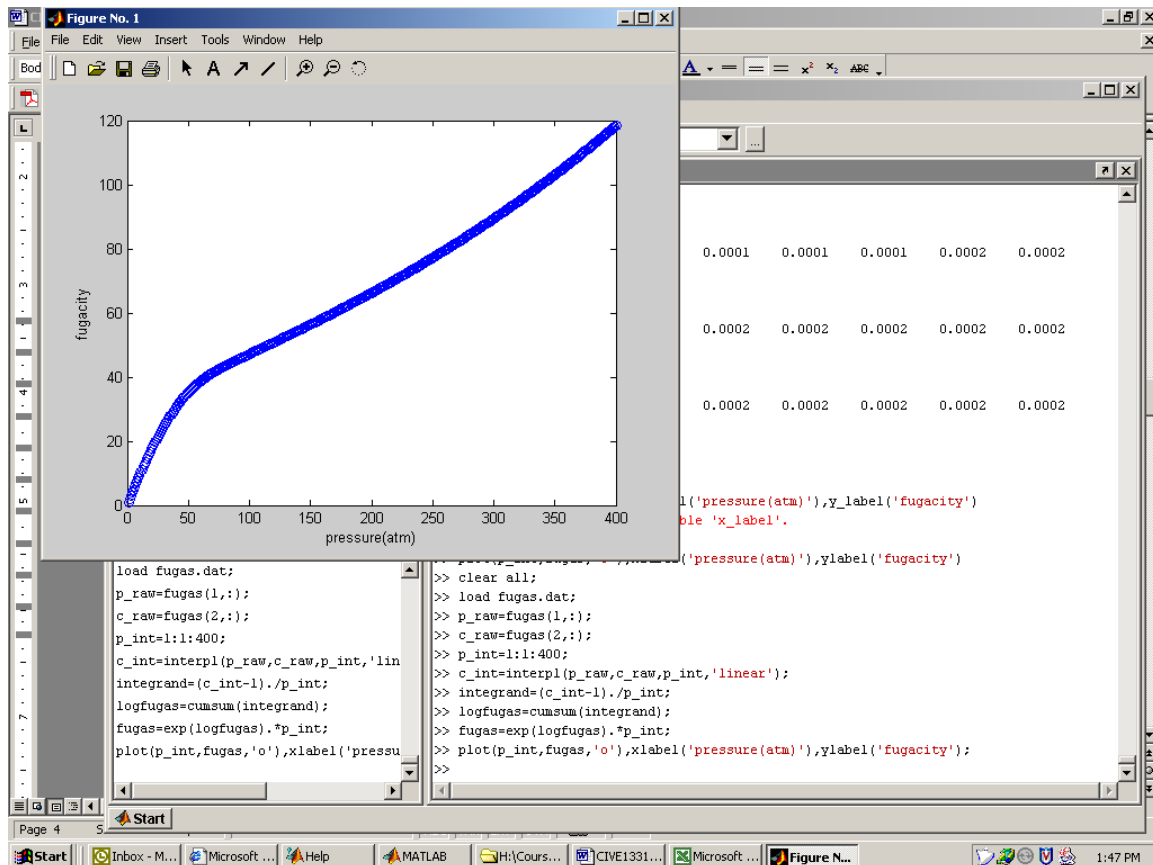
Step 2 Formulate the integrand using these two vectors .

H:\Courses\CIVE\_1331\CE1331F2003\CIVE\_1331\CIVE1331\_ABET\_Format\LabExercises\Lab013\CIVE1331\_Solution\_013.doc

## CIVE 1331 Computing for Engineers

Step 3. Cumulate the integrand and this will represent the integral with the variable limit.

Step 4. Perform required arithmetic on the integral to recover fugacity.



## CIVE 1331 Computing for Engineers

**Problem 4 Ordinary Differential Equations – Use MatLab**

Use the Euler method to solve for y at x=0.1 from

$$\frac{dy}{dx} = x + y + xy$$

With the initial condition that y=1.0 at x=0.0. Use a stepsize of  $\Delta x = 0.01$  and again using 0.001. Compare the results.

Step 1: Construct the finite difference equation for updating:

$$y(x + \Delta x) = y(x) + \Delta x(x + y + xy)$$

Step 2: Starting with the initial values  $(x,y) = (0,1)$  step forward by increments of  $\Delta x$  until we get to the desired  $(x,y)$  pair.

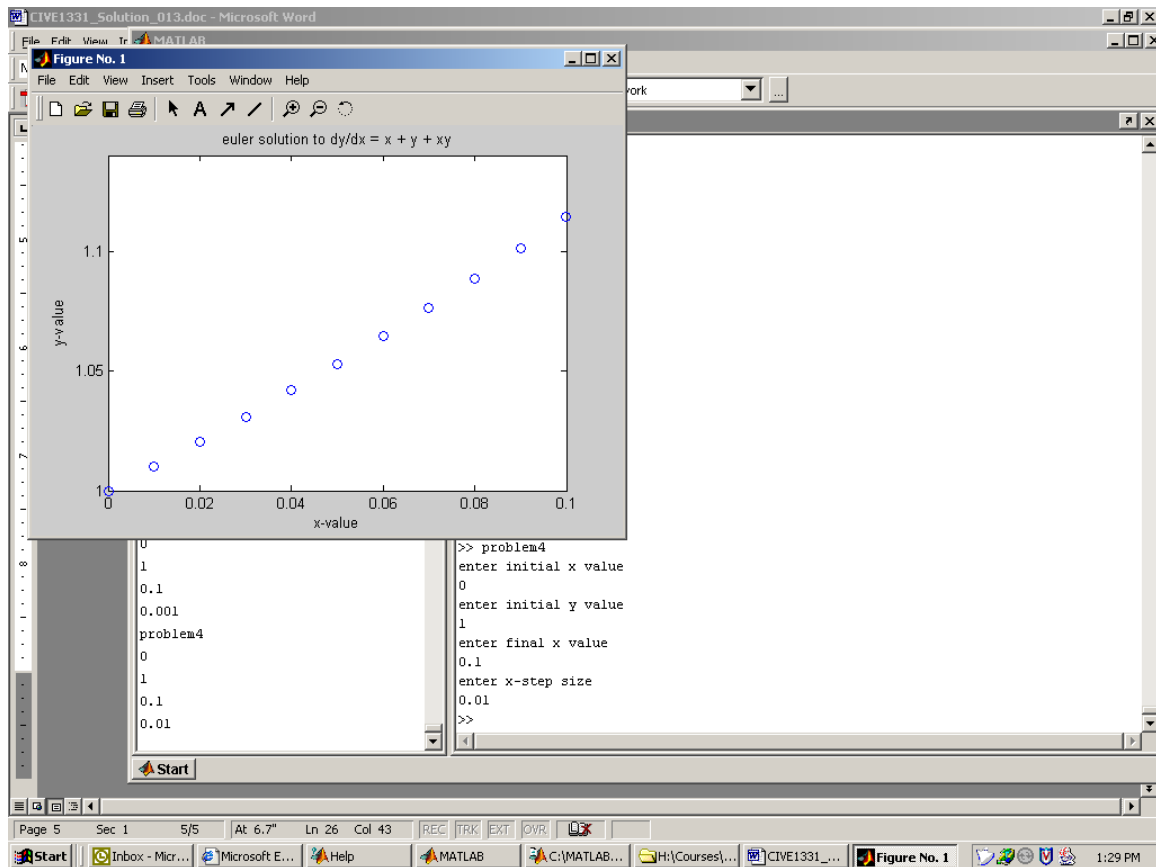
Step 3: The scripts below should perform the required computations. Also two screen captures of the two cases requested.

```
% function to integrate
function euler_out=euler_func(x_input,y_input);
euler_out=x_input+y_input+(x_input*y_input);
% end function name is 'euler_func'; file name is 'euler_func.m'

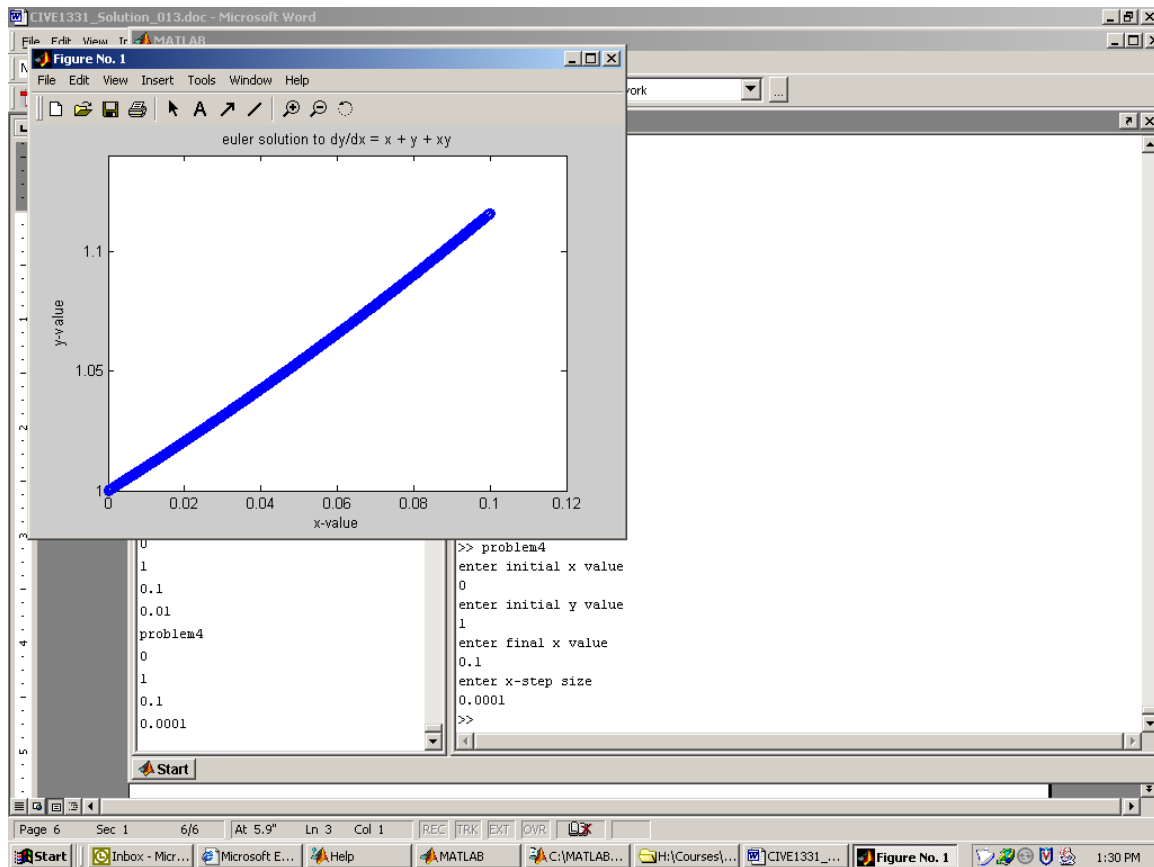
% euler's method
clear all
x_start=input('enter initial x value \n');
y_start=input('enter initial y value \n');
x_end=input('enter final x value \n');
x_step=input('enter x-step size \n');

% figure number of update steps to take
nsteps=(x_end-x_start)/x_step;
% start marching
x_now(1)=x_start;
y_now(1)=y_start;
for i=2:1:nsteps+1;
    y_now(i)=y_now(i-1)+x_step *(euler_func(x_now(i-1),y_now(i-1)));
    x_now(i)=x_now(i-1)+x_step;
end
plot(x_now,y_now,'o'),xlabel('x-value'),ylabel('y-value'),title('euler solution to dy/dx
= x + y + xy')
```

## CIVE 1331 Computing for Engineers



## CIVE 1331 Computing for Engineers



Naturally you need to extract the result at (0.1,y?) but that is left to you, just extract the proper part of the result array.