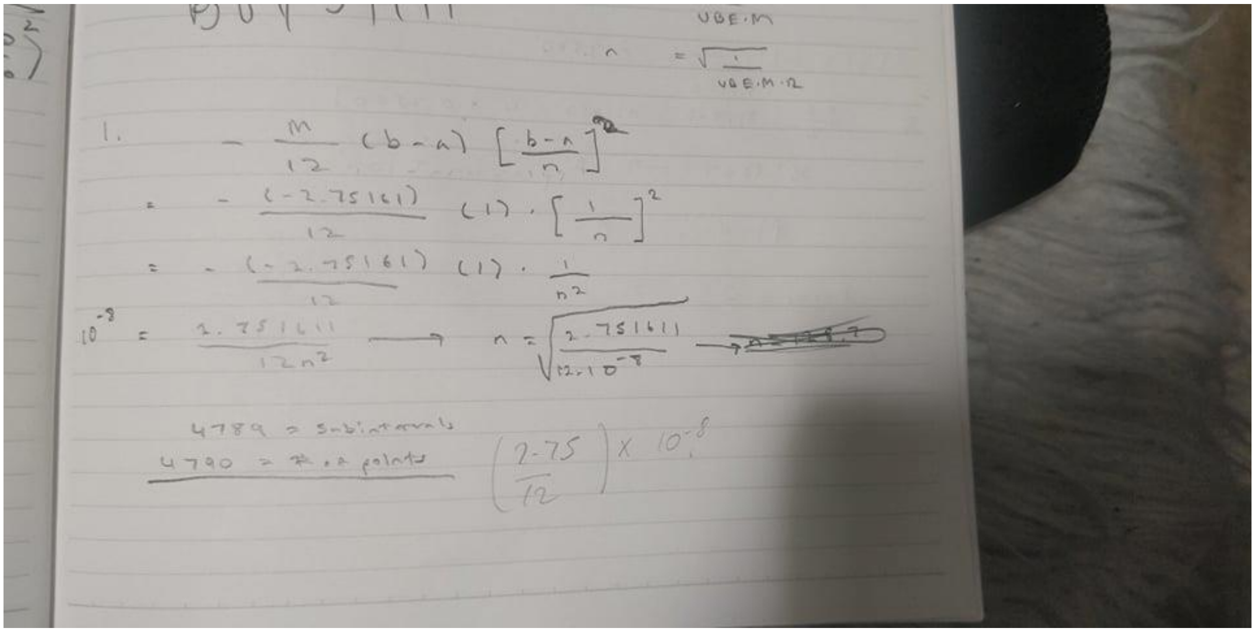
Assignment 3

Jame Tran, tranj52, 400144141

Question 1



$n = \sqrt{\frac{1}{UBE \cdot M \cdot 12}}$

1. $-\frac{M}{12}(b-a)\left[\frac{b-a}{n}\right]^2$

$= -\frac{(-2.751611)}{12}(1)\cdot\left[\frac{1}{n}\right]^2$

$= -\frac{(-2.751611)}{12}(1)\cdot\frac{1}{n^2}$

$10^{-8} = \frac{2.751611}{12n^2} \longrightarrow n = \sqrt{\frac{2.751611}{12 \cdot 10^{-8}}} \longrightarrow$

4789 = subintervals

4790 = # of points

$\left(\frac{2.75}{12}\right) \times 10^{-8}$

Question 2

2. $\int_{-1}^{1} (x-0.5)^2 \, dx \longrightarrow$

5 points $\longrightarrow$ 4 subintervals $\qquad h = \dfrac{b-a}{n}$

$a = -1 \qquad b = 1$

$= \dfrac{1-(-1)}{4}$

$t = (-1, -0.5, 0, 0.5, 1)$

$\quad\; 0 \quad\; 1 \quad\; 2 \quad 3 \quad 4$

$= 2/4 \rightarrow 0.5$

$$\approx \frac{h}{3}\left[f(a) + 2\sum_{i=1}^{n/2-1} f(t_{2i}) + 4\sum_{i=1}^{n/2} f(t_{2i-1}) + f(b) \right]$$

$f(x) = (x-0.5)^2 \qquad f(b) = f(1)$

$f(-1) = (-1 - 0.5)^2 \qquad = (1-0.5)^2$

$= 2.2500 \qquad = 0.2500$

$2\sum_{i=1}^{(4/2)-1=1} f(t_{2i}) = f(0) \longrightarrow 0.2500 \cdot (2)$

$= 0.5000$

$4\sum_{i=1}^{4/2=2} f(t_{2i-1}) \longrightarrow 4 \cdot [f(1) + f(3)]$

$\rightarrow 4[1 + 0]$

$= 4$

$f(b) = f(1) \longrightarrow 0.2500$

$\approx \dfrac{0.5}{3}[\,2.2500 + 0.5000 + 4 + 0.2500\,]$

$\approx 1.1667 \longrightarrow$ Approximation

$M = f^{(4)}(x) = 0$

Error $= 0$

Question 3

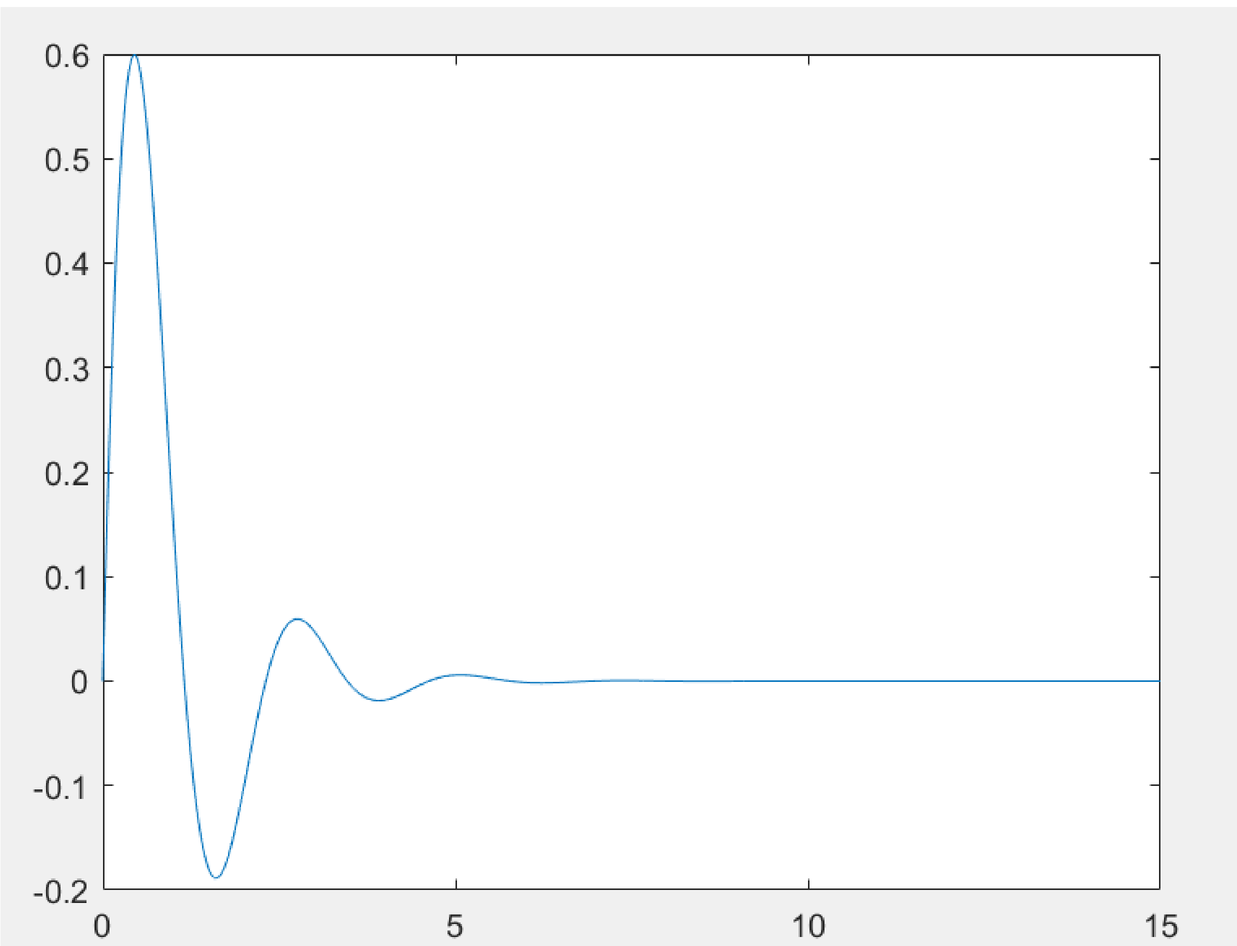
```
ans = 4×1
      2.9600
      1.7460
     -1.4600
      1.3140

x1_e = 0.0100
x2_e = 0.0060
x3_e = 0.0100
x4_e = 0.0060
```

Discussion

Our computed values have an error compared to the actual values. This is because this is an overdetermined system, or that there are more equations than there are variables. Such a system is inconsistent, or there are no solutions to the system of equations. We use a least squares method in Matlab (\) method to determine an approximate value with the lowest value for the sum of the squares of the residuals (variance). The error in the approximation is shown above, e.g. the absolute error (absolute value of measured value minus the approximation) of x1 is 0.0100.

Question 4

Function = $\int_{0.5}^{1} e^{-x}\sin(ex)\,dx$

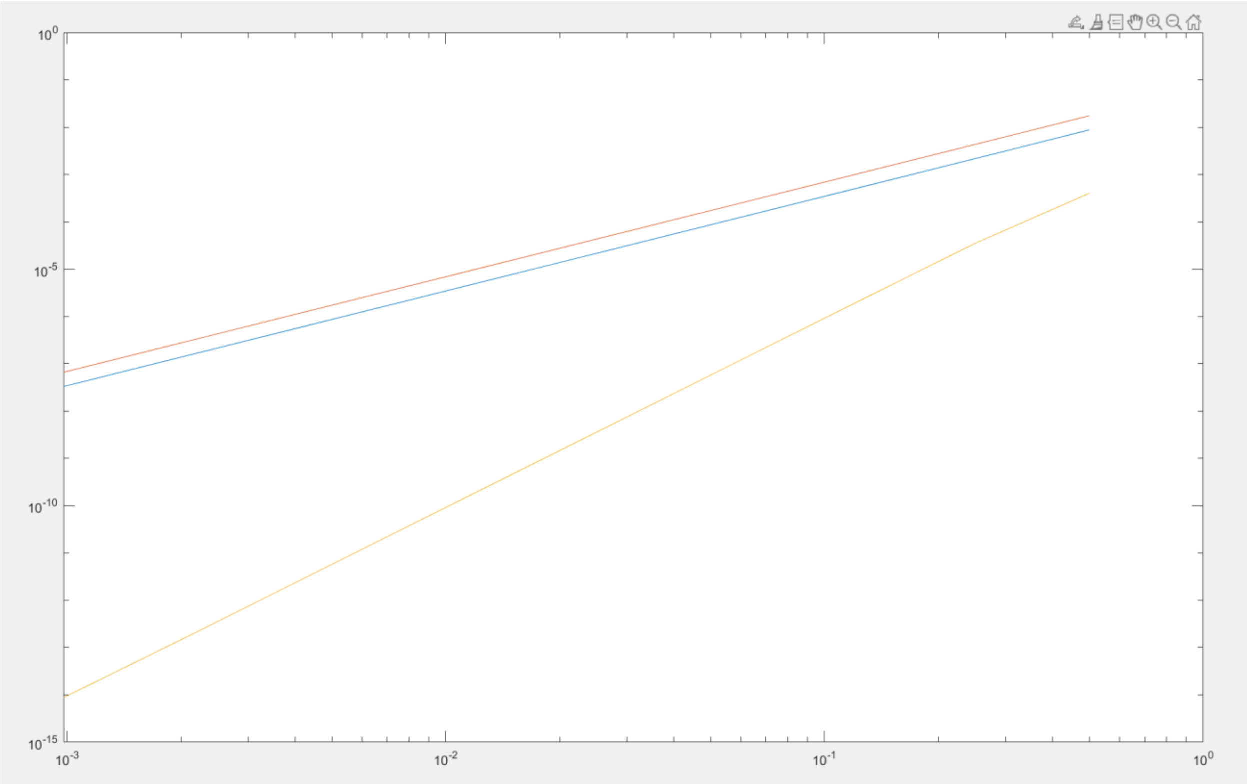


```
tol =

   0.009000000000000


accurate_value =

   0.202641716446010


adaptive =

0.202642646400613748077409089394898


adaptive_count =

     1


composite =

   0.212052991441950


composite_count =

     1000000


comp_error =

   0.009411274995940


adap_error =

0.00000092995460393615244261885284989451
```

```
function main_simpson
format long

fun = @(x) (exp(-x).*sin(exp(1).*x));

tol = 9e-3
accurate_value = quad(fun, 0.5, 1, 9e-3)

[adaptive,adaptive_count] = quadSimpson(fun, 0.5, 1, 9e-3)

composite = simpson(fun, 0.5, 1, 1000000)
composite_count = 1000000

comp_error = abs(accurate_value - composite)
adap_error = abs(accurate_value - adaptive)

x_vals = 0:1/100:15;

plot(x_vals, fun(x_vals))



function [c, count] = quadSimpson(fun, a, b, tol)
h = b - a;
c = (a + b)/2; %midpoint
count = 1;

S1 = vpa((h/6)*(fun(a) + 4*fun((a+b)/2) + fun(b)));
S2 = vpa((h/12)*(fun(a) + 4*fun((a+c)/2) + 2*fun(c) + 4*fun((c+b)/2) + fun(b)));
E2 = vpa((1/15)*(S2 - S1));

if abs(E2) <= tol
    c = S2 + E2;

else
    [Q1, Q1_c] = quadSimpson(fun, a, c, tol/2);
    [Q2, Q2_c] = quadSimpson(fun, c, b, tol/2);
    c = Q1(1) + Q2(1);
    count = count + Q1_c + Q2_c;

end
end

function c = simpson(fun, a, b, n)
h = (b - a)/n;
t_values = zeros(1, n+1);
    for i = 0:n
        t = 0 + i*h;
        t_values(i+1) = t;
    end
sum_even = 0;
    for i = 1:((n/2) - 1)
        c_even = t_values(2*i + 1);
        sum_even = sum_even + fun(c_even);
    end
```

```
sum_odd = 0;
    for i = 1:(n/2)
        c_odd = t_values(2*i);
        sum_odd = sum_odd + fun(c_odd);
    end
c = (h/3) * (fun(0) + 2*sum_even + 4*sum_odd + fun(1));
end
end
```

Question 5



Midpoint Method 3.48e-02*h^2.00

Trapezoid Method 6.95e-02*h^2.00

Simpson's Method 7.84e-03*h^3.97

```
function main_integration_error()

format long
h_values = zeros(1,10);
midpoint_errors = zeros(1, 10);
trap_errors = zeros(1, 10);
simp_errors = zeros (1, 10);

for i = 1:10
    midpoint_errors(i) = abs(midpoint(i) - erf(1));
    trap_errors(i) = abs(trapezoid(i) - erf(1));
    simp_errors(i) = abs(simpson(i) - erf(1));
    h_values(i) = 1/(2^(i));
end

loglog(h_values, midpoint_errors, h_values, trap_errors, ...
    h_values,simp_errors )

temp_matrix = ones(1, 10);

a = horzcat(temp_matrix', log(h_values)');
b_midpoint = log(midpoint_errors)';
temp_m = a\b_midpoint;
c_m = exp(temp_m(1));
k_m = temp_m(2);

fprintf( "Midpoint Method %.2e*h^%.2f\n" , c_m, k_m);
b_trap = log(trap_errors)';
temp_t = a\b_trap;
c_t = exp(temp_t(1));
k_t = temp_t(2);
fprintf( "Trapezoid Method %.2e*h^%.2f\n" , c_t, k_t);

b_simp = log(simp_errors)';
temp_s = a\b_simp;
c_s = exp(temp_s(1));
k_s = temp_s(2);
fprintf( "Simpson's Method %.2e*h^%.2f\n" , c_s, k_s);

function c = midpoint(i)
h = 1/(2^(i));
n = (1 - 0)/h;
sum = 0;
    for j = 1:n
        sum = sum + uwu((j - (1/2))*h);
    end
c = sum*h;
```

```
end
function c = trapezoid(i)
h = 1/(2^(i));
n = (1 - 0)/h;
t_values = zeros(1, n+1);
    for i = 0:n
        t = 0 + i*h;
        t_values(i+1) = t;
    end
sum = 0;
    for i = 2:n
        sum = sum + uwu(t_values(i));
    end

c = h/2 * (uwu(0) + uwu(1)) + h*sum;
end
function c = simpson(i)
h = 1/(2^(i));
n = (1 - 0)/h;
t_values = zeros(1, n+1);
    for i = 0:n
        t = 0 + i*h;
        t_values(i+1) = t;
    end
sum_even = 0;
    for i = 1:((n/2) - 1)
        c_even = t_values(2*i + 1);
        sum_even = sum_even + uwu(c_even);
    end
sum_odd = 0;
    for i = 1:(n/2)
        c_odd = t_values(2*i);
        sum_odd = sum_odd + uwu(c_odd);
    end
c = (h/3) * (uwu(0) + 2*sum_even + 4*sum_odd + uwu(1));
end
function c = uwu(x)
c = (2/(sqrt(pi)))*exp(-x^(2));
end
end
```

Question 6

planets	a	b	c	d	e
"Jupiter"	-1.18539671052994	0.0220291343599878	-0.495038694703956	-0.145054095760587	26.9822159472944
"Saturn"	-1.16674489055553	0.0359631921458413	0.11672948526852	-1.08985206866567	90.38160186136
"Uranus"	-1.19413366541963	0.0116273541038122	1.82705145910166	-0.259256316734578	367.268143728224
"Neptune"	-1.16712825105863	0.0207038823855469	-0.392686521552642	-0.423157876391122	903.80867108896
"Pluto"	-1.00333670919347	0.238832889904975	11.8470978717456	12.7170633447915	1290.67992773477

```
function main_orbit()
```

```
format long
planet_data = table2array(readtable("nbody.dat"));
data_size = size(planet_data, 1);


jupiter_x = planet_data(:, 2);
jupiter_y = planet_data(:, 3);

saturn_x = planet_data(:, 5);
saturn_y = planet_data(:, 6);

uranus_x = planet_data(:, 8);
uranus_y = planet_data(:, 9);

neptune_x = planet_data(:, 11);
neptune_y = planet_data(:, 12);

pluto_x = planet_data(:, 14);
pluto_y = planet_data(:, 15);

temp = ones(1, data_size)';

jupiter_a = horzcat(jupiter_y.^2, jupiter_x.*jupiter_y, jupiter_x, jupiter_y ...
    ,temp);

jupiter_coeffs = jupiter_a\jupiter_x.^2;

saturn_a = horzcat(saturn_y.^2, saturn_x.*saturn_y, saturn_x, saturn_y ...
    ,temp);
saturn_coeffs = saturn_a\saturn_x.^2;

uranus_a = horzcat(uranus_y.^2, uranus_x.*uranus_y, uranus_x, uranus_y ...
    ,temp);
uranus_coeffs = uranus_a\uranus_x.^2;

neptune_a = horzcat(neptune_y.^2, neptune_x.*neptune_y, neptune_x, neptune_y ...
    ,temp);
neptune_coeffs = neptune_a\neptune_x.^2;

pluto_a = horzcat(pluto_y.^2, pluto_x.*pluto_y, pluto_x, pluto_y ...
    ,temp);
pluto_coeffs = pluto_a\pluto_x.^2;

planets = ["Jupiter", "Saturn", "Uranus", "Neptune", "Pluto"]';
a = [jupiter_coeffs(1), saturn_coeffs(1), uranus_coeffs(1), ...
    neptune_coeffs(1), pluto_coeffs(1)]';

b = [jupiter_coeffs(2), saturn_coeffs(2), uranus_coeffs(2), ...
    neptune_coeffs(2), pluto_coeffs(2)]';

c = [jupiter_coeffs(3), saturn_coeffs(3), uranus_coeffs(3), ...
    neptune_coeffs(3), pluto_coeffs(3)]';

d = [jupiter_coeffs(4), saturn_coeffs(4), uranus_coeffs(4), ...
    neptune_coeffs(4), pluto_coeffs(4)]';
```

```
e = [jupiter_coeffs(5), saturn_coeffs(5), uranus_coeffs(5), ...
    neptune_coeffs(5), pluto_coeffs(5)]';

fun = table(planets, a, b, c, d, e)


% x = jupiter_x;
% y = jupiter_y;
%
% a = jupiter_coeffs(1);
% b = jupiter_coeffs(2);
% c = jupiter_coeffs(3);
% d = jupiter_coeffs(4);
% e = jupiter_coeffs(5);
%
% plot(x,y);
% hold on;
% [xs, ys] = meshgrid(min(x)-1:0.1:max(x)+1, min(y)-1:0.1:max(y)+1);
% contour(xs, ys, a*ys.^2+b*xs.*ys+c*xs+d*ys+e-xs.^2, [0, 0], 'k--', 'LineWidth', 1);


end
```