Q1

Problem 1 [5 points] Consider the integral $\int_0^1 \sin(\pi x^2/3)dx$. Suppose that we wish to integrate it numerically with an error of magnitude less than 10^{-8} and using equally spaced points and the trapezoidal rule. Derive how many points are needed to achieve this accuracy.

$f(x) = \sin(\frac{\pi x^2}{3})$ $\quad f'(x) = \frac{2\pi x}{3}\cos(\frac{\pi x^2}{3})$ $\quad f''(x) = \frac{2\pi}{3}\cos(\frac{\pi x^2}{3}) - (\frac{2\pi x}{3})^2\sin(\frac{\pi x^2}{3})$

error $= \left|-\frac{f''(u)}{12}(b-a)h^2\right| \le \left|-\frac{\max(f''(u))}{12}(b-a)h^2\right|$

$f''(x)$ is monotomicly decreasing $|f''(0)| = \frac{2\pi}{3} \approx 2.09$ $\quad |f''(1)| = \frac{4\pi^2}{9}\frac{\sqrt{3}}{2} - \frac{\pi}{3} \approx 2.7516$

so $f''(1)$ is $\max|f''(x)|$ in $[0,1]$

$f''(1) = 2.7516$

error $\le \left|-\frac{2.7516}{12}(b-a)h^2\right| = \left|\frac{2.7516}{12}h^2\right|$

$\frac{2.7516}{12}h^2 \le 10^{-8}$

$h^2 \le \frac{18\times10^{-8}}{\pi}$

$h \le 5.307e^{-4}$

number of points $\ge \frac{a-b}{h} \ge \frac{1}{h} \approx 1884.4$ points

Q2

Problem 2 [3 points] Approximate $\int_{-1}^{1}(x-0.5)^2\,dx$ using the Simpson's rule with 5 equally spaced points and calculate the error in this approximation.

$h = \frac{2}{5} = 0.4$

X	-1	-0.6	-0.2	0.2	1
Y	2.25	1.21	0.44	0.04 0.09	0.25

$$I \approx \frac{h}{3}\left[f(a) + 2\sum_{i=1}^{\frac{5}{2}-1} f(x_{2i}) + 4\sum_{i=1}^{\frac{5}{2}} f(x_{2i-1}) + f(b)\right]$$

$$= \frac{0.4}{3}\left[f(0) + 2(f(2)+f(4)) + 4(f(1)+f(3)) + f(5)\right]$$

$$= \frac{0.4}{3}\left[2.25 + 4\times(1.21+0.04) + 2\times(0.44+0.09) + 0.25\right]$$

$$= 1.16$$

$$\int_{-1}^{1}(x-0.5)^2\,dx = \int_{-1}^{1}(x^2 - x + 0.25)\,dx = \frac{x^3}{3} - \frac{x^2}{2} + 0.25x\Big|_{-1}^{1}$$

$$= 1.16667$$

error $= 1.16667 - 1.16 = 0.00667$

Q3

```
>>  A=[1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1;1 -1 0 0;1 0 -1 0;1 0 0 -1;0 1 -1 0;0 1 0 -1;0 0 1 -1];
b=[2.95;1.74;-1.45;1.32;1.23;4.45;1.61;3.21;0.45;-2.75];
X=(inv(A'*A)*A'*b)

X =

    2.9600
    1.7460
   -1.4600
    1.3140
```

Problem 3 [3 points] A common problem in surveying is to determine the altitudes of a series of points with respect to some reference point. The measurements are subject to error, so more observations are taken than are necessary to determine the altitudes, and the resulting overdetermined system is solved in the least square sense to smooth out the error. Suppose that there are four points whose altitudes x_1, x_2, x_3, x_4 are to be determined. In addition to direct measurements of each x_i, with respect to a reference point, measurements are taken of each point with respect to all of the others. The resulting measurements are:

$$
\begin{aligned}
x_1 &= 2.95 & x_2 &= 1.74 \\
x_3 &= -1.45 & x_4 &= 1.32 \\
x_1 - x_2 &= 1.23 & x_1 - x_3 &= 4.45 \\
x_1 - x_4 &= 1.61 & x_2 - x_3 &= 3.21 \\
x_2 - x_4 &= 0.45 & x_3 - x_4 &= -2.75
\end{aligned}
$$

From these data, find the best values for the altitudes. How do your computed values compare with the direct measurements?

for an overdetermined system, we can derive $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$ by the normal equation of least squres method

$$x = (A^T A)^{-1} A^T b = A^{\dagger} b$$

where A, b are as following:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad b = \begin{bmatrix} 2.95 \\ 1.74 \\ -1.45 \\ 1.32 \\ 1.23 \\ 4.45 \\ 1.61 \\ 3.21 \\ 0.45 \\ -2.75 \end{bmatrix}$$

to solve $[x_1, x_2, x_3, x_4]^T$ from matlab

we can get $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2.96 \\ 1.746 \\ -1.46 \\ 1.314 \end{bmatrix}$

Q4

f(x)=e^(-2366936*|x^3|+54321*x^4)

C1>100*C2

Tol=1e-3

a=-1111

b=1111

```
ERROR1 =

   9.9997e-04


ERROR2 =

   4.9299e-06
```

```
C2 =

        2695
```

```
C1 =

      369113
```

Matlabcode

Note that the function recursion_simpson will be used to construct adaptive_simpson

```
function [S,C2]=recursion_simpson(f,S,nodes,y,C2,tol);
    h=(nodes(1,end)-nodes(1))/2;
    nnodes=[nodes(1)+h/2 nodes(2)+h/2];
    ny=feval(f,nnodes);
    S1=(y(1)+4*ny(1)+y(2))*h/6;
    S2=(y(2)+4*ny(2)+y(3))*h/6;
    temp=S1+S2;
    C2=C2+2;
    if abs(temp-S)<tol
        S=16/15*(S1+S2)-S/15;
    else
        tol=tol/2;

[S1,C21]=recursion_simpson(f,S1,[nodes(1),nodes(1)+h/2,nodes(2)],[y(1),ny(1),y(2)],C2,tol);

[S2,C22]=recursion_simpson(f,S2,[nodes(2),nodes(2)+h/2,nodes(end)],[y(2),ny(2),y(3)],C2,tol);
        S=S1+S2;
        C2=C21+C22;
    end
end
function [S,C2]=adaptive_simpson(f,a,b,tol)
    C2=0;
    h=b-a;
    nodes=[a,a+h/2,b];
    y=f(nodes);
    S=(y(1,1)+4*y(1,2)+y(1,3))*h/6;
    tol=tol*15;
```

```
    [S,C2]=recursion_simpson(f,S,nodes,y,C2,tol);
    C2=C2+3
end


function [SC,C1]=composite_simpson(f,a,b,tol,I);
    n=0;
    error=88888;
    while error>tol
        n=n+2;
        x=linspace(a,b,2*n+1);
        y=f(x);
        SC=((b-a)/n/6)*(y(1)+y(2*n+1)+2*sum(y(3:2:2*n-
1))+4*sum(y(2:2:2*n)));
        error=abs(SC-I);
    end
    C1=2*n+1;
end
tol=1e-3;
f = @(x)exp(-2366936.*(abs(x.^3)+54321.*x.^4));
I = quad(f,-1111,1111,1e-5);
[A2 C2]=adaptive_simpson(f,-1111,1111,tol)
[A1 C1]=composite_simpson(f,-1111,1111,tol,I)
ERROR1=abs(A1-I)
ERROR2=abs(A2-I)
```

Q5

(a)

Erf1 of midpoint rule

errorm

1x10 double

	1	2	3	4	5	6	7	8	9	10
1	0.0088	0.0022	5.4100e-04	1.3516e-04	3.3783e-05	8.4455e-06	2.1114e-06	5.2784e-07	1.3196e-07	3.2990e-08

Erf1 simpson rule

errors

1x10 double

1	2	3	4	5	6	7	8	9	10
3.5258e-05	2.2429e-06	1.4062e-07	8.7952e-09	5.4980e-10	3.4364e-11	2.1475e-12	1.3467e-13	8.9928e-15	1.1102e-16

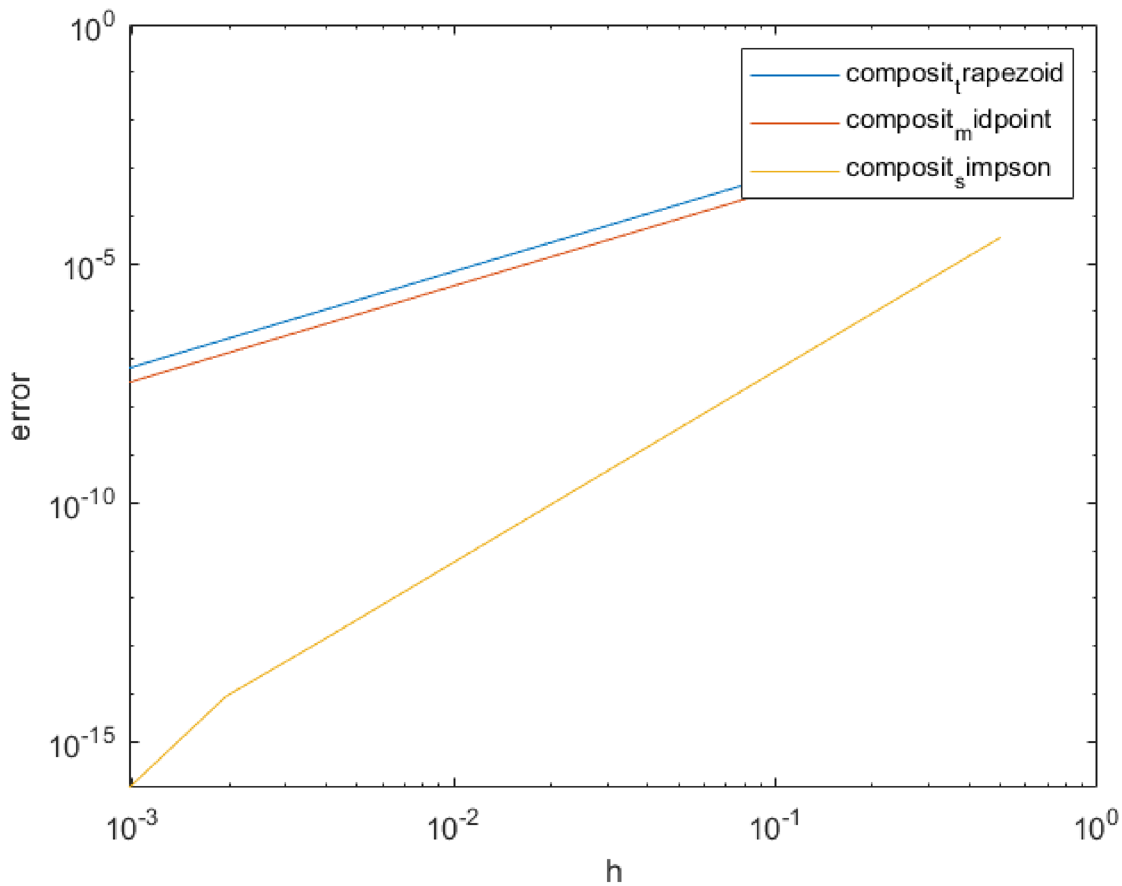
Erf1 of trapezoid rule

errort

1x10 double

	1	2	3	4	5	6	7	8	9	10
1	0.0174	0.0043	0.0011	2.7029e-04	6.7565e-05	1.6891e-05	4.2227e-06	1.0557e-06	2.6392e-07	6.5980e-08

Error plot



(b)

```
trapezoid 6.95e-02*h^2.00
midpoint 3.48e-02*h^2.00
simpson 7.71e-04*h^4.12
```

```
ct =

    0.0695


kt =

    2.0008


cm =

    0.0348


km =

    2.0014


cs =

   7.7100e-04


ks =

    4.1158
```

Matlab code

```
function erft=composit_trapezoid(f,b,h)
    n=b/h;
    i=1:n;
    erft=sum(f(i.*h)).*h+f(0)*h/2-f(b)*h/2;
end
function erfm=composit_midpoint(f,b,h)
    n=b/h;
    i = 1:n;
```

```
    erfm=sum(f((i-0.5).*h))*h;
end
function erfs=composit_simpson(f,b,h)
    n=b/h;
    ti=linspace(0,b,2*n+1);
    y=f(ti);
    erfs=(h/6)*(y(1)+y(2*n+1)+2*sum(y(3:2:2*n-
1))+4*sum(y(2:2:2*n)));
end


f = @(t)exp(-t.^2)*2/sqrt(pi);
I=integral(f,0,1);
for i = 1:10
   h(i)=1/2^i;
   errort(i) = abs(composit_trapezoid(f,1,h(i))-I);
   errorm(i) = abs(composit_midpoint(f,1,h(i))-I);
   errors(i) = abs(composit_simpson(f,1,h(i))-I);
end

loglog(h,errort);
hold on
loglog(h,errorm);
loglog(h,errors);
legend('composit_trapezoid','composit_midpoint','composit_s
impson');
xlabel('h');
ylabel('error');
hold off
t=[ones(10,1),log(h)']\(log(errort))';
ct=exp(t(1));
kt=t(2);
m=[ones(10,1),log(h)']\(log(errorm))';
cm=exp(m(1));
km=m(2);
s=[ones(10,1),log(h)']\(log(errors))';
cs=exp(s(1));
ks=s(2);
fprintf('trapezoid %.2e*h^%.2f\n', ct, kt);
fprintf('midpoint %.2e*h^%.2f\n', cm, km);
fprintf('simpson %.2e*h^%.2f\n', cs, ks);
```

Q6

	a	b	c	d	e
Jupiter	-1.185397	0.022029	-0.495039	-0.145054	26.982216
Saturn	-1.166745	0.035963	0.116729	-1.089852	90.381602
Uranus	-1.194134	0.011627	1.827051	-0.259256	367.268144
Neptune	-1.167128	0.020704	-0.392687	-0.423158	903.808671
Pluto	-1.003337	0.238833	11.847098	12.717063	1290.679928

```
nbodydata=importdata('nbody.dat');
l=length(nbodydata);
c=zeros(5,5);
x = zeros(l,5);
y = zeros(l,5);
fprintf('                 a            b             c             d
e\n')
for i=0:4
    x(:,i+1)=nbodydata(:,2+3*i);
    y(:,i+1)=nbodydata(:,3+3*i);

c(:,i+1)=[y(:,i+1).^2,y(:,i+1).*x(:,i+1),x(:,i+1),y(:,i+1),
ones(l,1)]\x(:,i+1).^2;
end
fprintf('Jupiter  %f  %f  %f  %f  %f\nSaturn  %f  %f  %f  %
f  %f\nUranus  %f  %f  %f  %f  %f\nNeptune  %f  %f  %f  %f
%f\nPluto  %f  %f  %f  %f  %f\n',c(1,1),c(2,1),c(3,1),c(4,1
),c(5,1),c(1,2),c(2,2),c(3,2),c(4,2),c(5,2),c(1,3),c(2,3),c
(3,3),c(4,3),c(5,3),c(1,4),c(2,4),c(3,4),c(4,4),c(5,4),c(1,
5),c(2,5),c(3,5),c(4,5),c(5,5));
```