In [1]:

```
import numpy as np
import sympy as sp
```

Problem 1:

In [2]:

```
def trapezoid(a,b,f):
    return((b-a)*(f(a)+f(b))/2)
def simpson(a,b,f):
    return((b-a)*(f(a)+4*f((a+b)/2)+f(b))/6)
def midpoint(a,b,f):
    return((b-a)*(f((a+b)/2)))
```

a) $f(x) = 5x^2 + 3$

In [3]:

```
def f(x):
    return(5*x**2+3)
a=0; b=5;
i=simpson(a,b,f)
print('The approximate value of integral using simpsons rule is '
      +str(i)+'\n')
```

```
The approximate value of integral using simpsons rule is 223.3333333333333
4
```

The exact integral of f(x) is $\frac{5}{3}x^3 + 3x$ in the limits 0,5

In [4]:

```
def If(x):
    return((5/3)*x**3+3*x)
v = If(b)-If(a)
print('The analytical solution is '+str(v))
e = v-i
print('error is '+str(e))
```

```
The analytical solution is 223.33333333333334
error is 0.0
```

As, the integrand is a quadratic function. Use of simpsons method is appropriate as simpsons method is exact for polynomials upto degree 3.

b) $f(x) = 2x + 1$

In [5]:

```
def f(x):
    return(2*x+1)
a=1; b=4;
i=simpson(a,b,f)
j=trapezoid(a,b,f)
k=midpoint(a,b,f)
print('The approximate value of integral using simpsons rule is '+str(i)+'\n')
print('The approximate value of integral using Trapeziod rule is '+str(j)+'\n')
print('The approximate value of integral using midpoint rule is '+str(k)+'\n')
```

```
The approximate value of integral using simpsons rule is 18.0

The approximate value of integral using Trapeziod rule is 18.0

The approximate value of integral using midpoint rule is 18.0
```

The exact integral of f(x) is $x^2 + x$ in the limits 1,4

In [6]:

```
def If(x):
    return(x**2+x)
v = If(b)-If(a)
print('The analytical solution is '+str(v))
e = v-i
print('error is '+str(e))
```

```
The analytical solution is 18
error is 0.0
```

As the integrand is a linear function. We can use either of the three rules to find the exact value as all of them will give an exact value for a polynomial of degree 1.

c) $f(x) = e^x$

In [7]:

```
def f(x):
    return((np.e)**x)
a=1; b=2;
i=simpson(a,b,f)
j=trapezoid(a,b,f)
k=midpoint(a,b,f)
print('The approximate value of integral using simpsons rule is '
      +str(i)+'\n')
#print('The approximate value of integral using Trapezoid rule is '+str(j)+'\n')
#print('The approximate value of integral using midpoint rule is '+str(k)+'\n')
```

```
The approximate value of integral using simpsons rule is 4.672349034790325
```

The exact integral of f(x) is e^x in the limits 1,2

In [8]:

```
v = f(2)-f(1)
print('The analytical solution is '+str(v))
e = v-i
print('error is '+str(e))
```

```
The analytical solution is 4.670774270471604
error is -0.0015747643187209448
```

The function e^x can be expressed as $\sum_{n=0}^{\infty} \frac{x^n}{n!}$ (using taylors series) therefore integral cannot be exactly evaluated using any of the three rules. However, as simpsons rule gives an exact value for polynomial upto degree of 3, therefore it will give the closest approximation compared to other rules.

Problem 2:

In [9]:

```
def gauss_q2(a,b,f):
    p=1/np.sqrt(3)
    return((b-a)*(f(p)+f(-p))/2)
```

a) $\int_1^6 (3x^4 + 5x^2)dx$

In [10]:

```
x = sp.symbols('x')
z = sp.symbols('z')
a=1;b=6;
#z= ((2*x)-(a+b))/(b-a)
def change_var(a,b,z):
    return(((b-a)*z+(a+b))/2)
x = change_var(a,b,z)
f = 3*x**4+5*x**2
print('x = '+str(x))
print('f(z) = '+str(f))
```

```
x = 5*z/2 + 7/2
f(z) = 3*(5*z/2 + 7/2)**4 + 5*(5*z/2 + 7/2)**2
```

In [11]:

```
def f(z):
    return(3*(5*z/2 + 7/2)**4 + 5*(5*z/2 + 7/2)**2)
i=gauss_q2(a,b,f)
print('The approximated value of integral is '+str(i))
```

```
The approximated value of integral is 4971.25
```

The exact value of the integral is $3x^5/5 + 5x^3/3$ in the limits 1,6

In [12]:

```
def If(x):
    return(3*x**5/5+5*x**3/3)
v=If(b)-If(a)
print('The analytical solution is '+str(v))
e = v-i
print('error is '+str(e))
```

```
The analytical solution is 5023.333333333334
error is 52.0833333333
```

We know that 2-point gauss quadrature can give us an exact value if the integrand is a polynomial of degree 3 or less. However here our integrand is a polynomial of degree 4, therefore the quadrature will not give an exact solution.

b) $\int_1^4 (e^{x-5}) dx$

In [13]:

```
x = sp.symbols('x')
z = sp.symbols('z')
e = sp.symbols('e')
a=1;b=4;
def change_var(a,b,z):
    return(((b-a)*z+(a+b))/2)
x = change_var(a,b,z)
f = e**(x-5)
print('x = '+str(x))
print('f(z) = '+str(f))
```

```
x = 3*z/2 + 5/2
f(z) = e**(3*z/2 - 5/2)
```

In [14]:

```
def f(z):
    return(np.e**(3*z/2 - 5/2))
i=gauss_q2(a,b,f)
print('The approximated value of integral is '+str(i))
```

```
The approximated value of integral is 0.344518459477
```

The exact integral is e^{x-5} in the limits 1,4

In [15]:

```
def F(x):
    return(np.e**(x-5))
v=F(b)-F(a)
print('The analytical solution is '+str(v))
e = v-i
print('error is '+str(e))
```

```
The analytical solution is 0.34956380228270817
error is 0.00504534280522
```

As the integrand can be written as $e^{x-5} = e^{-5} * \sum_{n=0}^{\infty} \frac{x^n}{n!}$ which is polynomial of degree infinity and therefore the value estimated using gauss quadrature will not be exact.

c) $\int_a^b f(x)dx$

In [16]:

```
x = sp.symbols('x')
z = sp.symbols('z')
a = sp.symbols('a')
b = sp.symbols('b')
x=(((b-a)*z+(a+b))/2)
print('x = '+str(x))
print('dx = ('+str(sp.diff(x,z))+')dz')
print('f(x)dx= f('+str(x)+')*('+str(sp.diff(x,z))+')dz')
```

```
x = a/2 + b/2 + z*(-a + b)/2
dx = (-a/2 + b/2)dz
f(x)dx= f(a/2 + b/2 + z*(-a + b)/2)*(-a/2 + b/2)dz
```

$\int_a^b f(x)dx = \int_{-1}^1 f(a/2 + b/2 + z * (-a + b)/2) * (-a/2 + b/2)dz$

and the generalised 2-point gauss quadrature can be given as
$I = (-a/2 + b/2) * \left[f(a/2 + b/2 + (\frac{-1}{\sqrt{3}}) * (-a + b)/2) + f(a/2 + b/2 + (\frac{1}{\sqrt{3}}) * (-a + b)/2\right]$

Problem 3:

Lets assume the Hermite quadrature as $\int_{-1}^1 f(t)dx = w * f(t_1) + w * f(t_2) + w * f(t_3)$

for $f(x) = H_0(t) = 1$ we have $\int_{-1}^1 1dt = 3w$ which implies $w = 2/3$

for $f(x) = H_1(t) = 2t$ we have $\int_{-1}^1 2tdt = 0 = \frac{2}{3} * 2(t_1 + t_2 + t_3)$ which implies $t_1 + t_2 + t_3 = 0$

for $f(x) = H_2(t) = 4t^2 - 2$ we have $\int_{-1}^1 (4t^2 - 2)dt = \frac{-4}{3} = \frac{2}{3}(4(t_1^2 + t_2^2 + t_3^2) + 3 * (-2))$

$\Rightarrow t_1^2 + t_2^2 + t_3^2 = 1$

for $f(x) = H_3(t) = 8t^3 - 12t$ we have
$\int_{-1}^1 (8t^3 - 12t)dt = 0 = \frac{2}{3}(8(t_1^3 + t_2^3 + t_3^3) - 12 * (t_1 + t_2 + t_3))$

$\Rightarrow t_1^3 + t_2^3 + t_3^3 = 0$

We can use the identity that if $a + b + c = 0$ then $a^3 + b^3 + c^3 = 3abc$

$\Rightarrow t_1^3 + t_2^3 + t_3^3 = 3t_1t_2t_3 = 0$

let $t_2 = 0 \Rightarrow t_1 + t_3 = 0 \Rightarrow t_1 = -t_3$

$\Rightarrow 2 * t_1^2 = 1 \Rightarrow t_1 = \pm\frac{1}{\sqrt{2}}$ and $t_3 = \mp\frac{1}{\sqrt{2}}$

Therefore the 3 point hermite quadrature can be written as:

$$I = \frac{2}{3}\left(f(\frac{-1}{\sqrt{2}}) + f(0) + f(\frac{1}{\sqrt{2}})\right)$$

Further,the hermite quadrature will give an exact value for polynomials of degree 3 or less

In [17]:

```
def hermite_q3(f):
    p=1/np.sqrt(2)
    return((2/3)*(f(p)+f(0)+f(-p)))
#def If(x):
#    return(x**2+x)
#i = gauss_q2(-1,1,If)
#print(i)
#I = hermite_q3(If)
#print(I)
```

Problem 4:

a)$\int_0^6 \int_0^2 (-(x+3)^2 + y^2)dxdy$

In [18]:

```
x = sp.symbols('x')
y = sp.symbols('y')
u = sp.symbols('u')
v = sp.symbols('v')
x = change_var(0,2,u)
y = change_var(0,6,v)
print('x='+str(x))
print('y='+str(y))
print('dx='+str(sp.diff(x,u))+'du')
print('dy='+str(sp.diff(y,v))+'dv')
f = -(x+3)**2+y**2
print('f(u,v)='+str(f))
grad=sp.Matrix([[sp.diff(x,u),sp.diff(x,v)],[sp.diff(y,u),sp.diff(y,v)]])
J = sp.det(grad)
print('J= '+str(J))
print('f(u,v)*J(u,v)='+str(J*f))
```

```
x=u + 1
y=3*v + 3
dx=1du
dy=3dv
f(u,v)=-(u + 4)**2 + (3*v + 3)**2
J= 3
f(u,v)*J(u,v)=-3*(u + 4)**2 + 3*(3*v + 3)**2
```

In [19]:

```
def gauss_q2_2d(f):
    p = 1/np.sqrt(3);
    return(f(p,p)+f(-p,p)+f(-p,-p)+f(p,-p))
def g(u,v):
    return(-3*(u + 4)**2 + 3*(3*v + 3)**2)
i = gauss_q2_2d(g)
print('The approximated value of integral is '+str(i))
```

```
The approximated value of integral is -52.0
```

The exact integral is

$$\int_2^6 \big(- \frac{(x+3)^3}{3}|_{x=0}^{x=2} + (x * y^2)|_{x=0}^{x=2}\big)dy$$
$$= \int_2^6 \big(- (5^3 - 3^3)/3 + 2y^2\big)dy$$
$$= \frac{-98}{3}y|_{y=0}^{y=6} + \frac{2}{3}y^3|_{y=0}^{y=6}$$

In [20]:

```
v = (-98/3)*(6)+(2/3)*(6**3)
print('The analytical solution is '+str(v))
e = v-i
print('error is '+str(e))
```

```
The analytical solution is -52.0
error is 0.0
```

The gauss quadrature gives an exact solution as the integrand can be expressed as a product of two polynomials both of which have a degree 3 or less

b) $\int_0^6 \int_0^2 e^{-(x^2+y^2)}dxdy$

In [21]:

```
x = sp.symbols('x')
y = sp.symbols('y')
u = sp.symbols('u')
v = sp.symbols('v')
e = sp.symbols('e')
x = change_var(0,2,u)
y = change_var(0,6,v)
print('x='+str(x))
print('y='+str(y))
print('dx='+str(sp.diff(x,u))+'du')
print('dy='+str(sp.diff(y,v))+'dv')
f = e**(-(x**2+y**2))
print('f(u,v)='+str(f))
grad=sp.Matrix([[sp.diff(x,u),sp.diff(x,v)],[sp.diff(y,u),sp.diff(y,v)]])
J = sp.det(grad)
print('J= '+str(J))
print('f(u,v)*J(u,v)='+str(J*f))
```

```
x=u + 1
y=3*v + 3
dx=1du
dy=3dv
f(u,v)=e**(-(u + 1)**2 - (3*v + 3)**2)
J= 3
f(u,v)*J(u,v)=3*e**(-(u + 1)**2 - (3*v + 3)**2)
```

In [22]:

```
def g(u,v):
    return(3*np.e**(-(u + 1)**2 - (3*v + 3)**2))
i = gauss_q2_2d(g)
print('The approximated value of integral is '+str(i))
```

```
The approximated value of integral is 0.552653973779
```

The integrand can be expressed as

$$e^{-(x^2+y^2)} = e^{-x^2}e^{-y^2} = (\sum_{n=0}^{n=\infty} \frac{x^n}{n!}) * (\sum_{n=0}^{n=\infty} \frac{y^n}{n!})$$

which is a product of two infite degree polynomials. Thus the gaussian quadrature will not give an exact solution as gaussian quadrature gives an exact solution for product of two polynomials each of degree 3 or less

Problem 5:

In [23]:

```
def N(a,b):
    u = sp.symbols('u')
    v = sp.symbols('v')
    i = np.copy(a)
    j = np.copy(b)
    if i==1:
        if j==1:
            return((1-u)*(1-v)/4)
        elif j==2:
            return((1-u)*(1+v)/4)
    elif i==2:
        if j==2:
            return((1+u)*(1+v)/4)
        elif j==1:
            return((1+u)*(1-v)/4)
def Map(p):
    xs=np.copy(p[:,0])
    ys=np.copy(p[:,1])
    x = sp.symbols('x')
    y = sp.symbols('y')
    x = N(1,1)*(xs[0])+(xs[1])*N(1,2)+N(2,1)*(xs[2])+N(2,2)*(xs[3])
    y = N(1,1)*(ys[0])+N(1,2)*(ys[1])+N(2,1)*(ys[2])+N(2,2)*(ys[3])
    J = sp.det(sp.Matrix([[sp.diff(x,u),sp.diff(x,v)],[sp.diff(y,u),sp.diff(y,v)]]))
    return(x,y,J)
p = np.array([[0,0],[0,4],[2,0],[4,4]])
m = Map(p)
```

In [24]:

```
print('x= '+str(sp.expand(m[0])))
print('y= '+str(sp.expand(m[1])))
print('J= '+str(sp.expand(m[2])))
x = m[0];y=m[1]
f = x*y
J=m[2]
print('f(u,v)=x*y='+str(sp.expand(f)))
print('f(u,v)*J(u,v)='+str(sp.expand(f*J)))
```

```
x= 0.5*u*v + 1.5*u + 0.5*v + 1.5
y= 2.0*v + 2.0
J= 1.0*v + 3.0
f(u,v)=x*y=1.0*u*v**2 + 4.0*u*v + 3.0*u + 1.0*v**2 + 4.0*v + 3.0
f(u,v)*J(u,v)=1.0*u*v**3 + 7.0*u*v**2 + 15.0*u*v + 9.0*u + 1.0*v**3 + 7.0*
v**2 + 15.0*v + 9.0
```

Now we have to Analytically integrate the function $F(u,v) = f(u,v) * J(u,v)$ over a symmetric domain $u \in [-1,1] and v \in [-1,1]$ therefore the odd parts (i.e. of the form $u^a * v^b$ where a **or** b is of the form 2n+1) of the function can be neglected (as they will integrate to the value of 0) and only even parts can be considered. From the above function we can identify the even part of the function F(u,v) as $7.0 * v^2 + 9.0$

The integral becomes $\int_{-1}^{1}\int_{-1}^{1}(7 * v^2 + 9)dudv$

In [25]:

```
g = 7.0*v**2+9.0
Gv = sp.integrate(g,v)
#print(Gv)
def gv(a):
    return(2.33333333333333*a**3 + 9.0*a)
I = gv(1)-gv(-1) #evaluating inner derivative with limits on v
I = I*2 #integrating wrt u and then evaluating using the limits
print('The analytical solution is '+str(I))
```

```
The analytical solution is 45.33333333333332
```

In [26]:

```
def F(u,v):
    return(1.0*u*v**3 + 7.0*u*v**2 + 15.0*u*v + 9.0*u + 1.0*v**3 + 7.0*v**2 + 15.0*v +
9.0)
I = gauss_q2_2d(F)
print('The approximated value of integral is '+str(I))
```

```
The approximated value of integral is 45.3333333333
```

Problem 6:

a) $\int \int_A (x+3)ydxdy$ over the region given in problem 5

As the domain is not changing for this problem we can use the values of Jacobian and X,Y obtained in the previous problem

In [27]:

```
x = m[0]
y=m[1]
J=m[2]
f = ((x+3)*y)*J
print('f(u,v)*J(u,v) = '+str(sp.expand(f)))
```

```
f(u,v)*J(u,v) = 1.0*u*v**3 + 7.0*u*v**2 + 15.0*u*v + 9.0*u + 1.0*v**3 + 1
3.0*v**2 + 39.0*v + 27.0
```

In [28]:

```
def F(u,v):
    return(1.0*u*v**3 + 7.0*u*v**2 + 15.0*u*v + 9.0*u + 1.0*v**3 + 13.0*v**2 + 39.0*v +
 27.0)
i = gauss_q2_2d(F)
print('The approximated value of integral is '+str(i))
```

```
The approximated value of integral is 125.333333333
```

b)$\int \int_A e^{xy}dxdy$

In [29]:

```
p = np.array([[-3,-2],[-1,2],[1,-2],[3,2]])
m = Map(p)
print('x= '+str(sp.expand(m[0])))
print('y= '+str(sp.expand(m[1])))
print('J= '+str(sp.expand(m[2])))
f = e**(m[0]*m[1])
print('f(u,v)=e^(x*y)='+str(sp.expand(f)))
print('f(u,v)*J(u,v)='+str(sp.expand(f*m[2])))
```

```
x= 2.0*u + 1.0*v
y= 2.0*v
J= 4.00000000000000
f(u,v)=e^(x*y)=e**(2.0*v**2)*e**(4.0*u*v)
f(u,v)*J(u,v)=4.0*e**(2.0*v**2)*e**(4.0*u*v)
```

In [30]:

```
def F(u,v):
    return(4.0*np.e**(2.0*v**2)*np.e**(4.0*u*v))
i = gauss_q2_2d(F)
print('The approximated value of integral is '+str(i))
```

```
The approximated value of integral is 63.2197857437
```