SymPy (Mr. P Solver)

Video Link: https://youtu.be/1yBPEPhq54M (https://youtu.be/1yBPEPhq54M)

Codes: https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbFpXWXFuczlGOVZ4RFIlbHV0eko2T29jUEw1UXxBQ3
(https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbFpXWXFuczlGOVZ4RFIlbHV0eko2T29jUEw1UXxBQ3

SymPy means SYMBOLIC PYTHON.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import sympy as smp
```

Introduction

Symbols can be defined as follows

In [2]:
```
x = smp.symbols('x')
```

In [3]:
```
x**2 - 5*x + 6
```

Out[3]: $x^2 - 5x + 6$

In [4]:
```
y = smp.sin(x)
y**2 + (smp.cos(x))**2
```

Out[4]: $\sin^2(x) + \cos^2(x)$

In [5]:
```
y1 = smp.log(x,10)  # input the base
y1
```

Out[5]: $\frac{\log(x)}{\log(10)}$

type **'smp.'** and press **'Tab'** to see all the SymPy functions

In [6]:
```
z = (x**2 - 5*x + 6)**2
display(z.factor(), z.expand())
display(smp.solve(z,x))
```

$(x-3)^2(x-2)^2$

$x^4 - 10x^3 + 37x^2 - 60x + 36$

```
[2, 3]
```

In [7]:
```
z.as_poly()
```

Out[7]: $\operatorname{Poly}\left(x^4 - 10x^3 + 37x^2 - 60x + 36, x, domain = \mathbb{Z}\right)$

type **'z.'** and press **'Tab'** to see all the operations that can be done on z

In [8]:
```
z.fourier_series([-smp.pi, smp.pi])
```

Out[8]:

$$-20\pi^2 \sin(x) + \frac{(45\pi + 10\pi^3)\sin(2x)}{\pi} + \frac{(-100\pi - 8\pi^3)\cos(x)}{\pi} + \frac{(2\pi^3 + 34\pi)\cos(2x)}{\pi}$$

In [9]:
```
z1 = smp.exp(x) + smp.exp(-x)
display(z1)
display(smp.solve(z1))
```

$e^x + e^{-x}$

```
[-I*pi/2, I*pi/2]
```

In [10]:
```
x = smp.symbols('x', real=True, positive=True)
smp.solve(x**4 - 16, x)
```

Out[10]:
```
[2]
```

Can define many variables at once

In [11]:
```
x, y, z = smp.symbols('x y z')
f = x**3*smp.sin(z) - y**2*smp.exp(z)
f
```

Out[11]: $x^3 \sin(z) - y^2 e^z$

In [12]:
```
y_soln = smp.solve(f, y)
y_soln
```

Out[12]:
```
[-sqrt(x**3*exp(-z)*sin(z)), sqrt(x**3*exp(-z)*sin(z))]
```

In [13]:
```
smp.solve(f,x)
```

Out[13]:
```
[(y**2*exp(z)/sin(z))**(1/3),
 -(y**2*exp(z)/sin(z))**(1/3)/2 - sqrt(3)*I*(y**2*exp(z)/sin(z))**(1/3)/2,
 -(y**2*exp(z)/sin(z))**(1/3)/2 + sqrt(3)*I*(y**2*exp(z)/sin(z))**(1/3)/2]
```

For multivariable expressions, can also substitute values in

In [14]:
```
f.subs([(y,smp.sqrt(x)), (z,smp.pi/2)])
```

Out[14]: $x^3 - xe^{\frac{\pi}{2}}$

SymPy to NumPy conversion so that the function can be plotted.

In [15]:
```
f1sym = y_soln[1]
f1sym
```

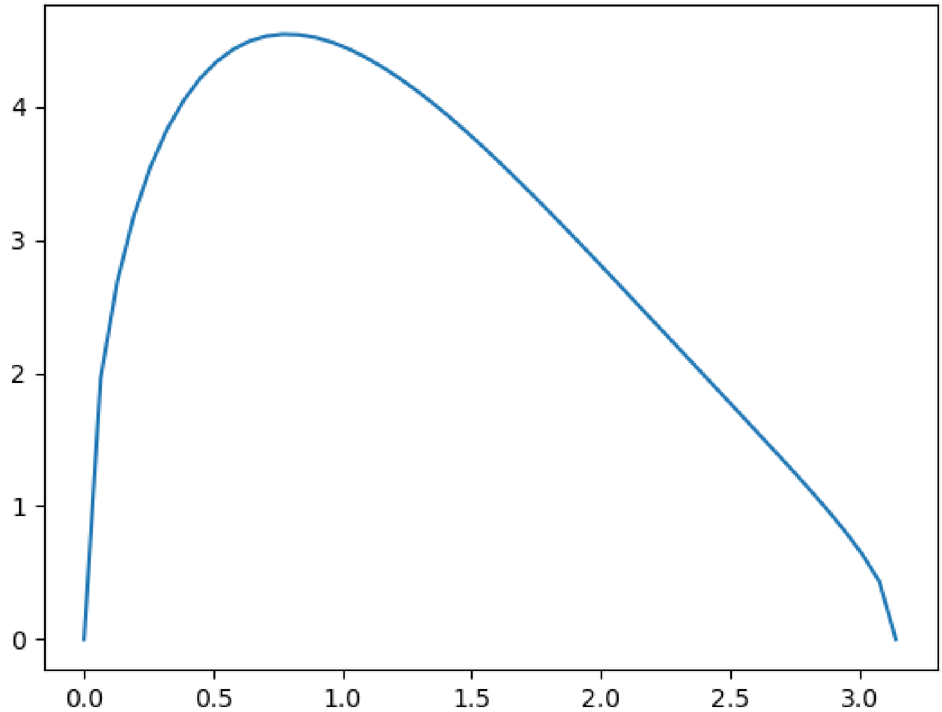
Out[15]: $\sqrt{x^3 e^{-z} \sin(z)}$

In [16]:
```
f1num = smp.lambdify([x,z], f1sym)
f1num(4,np.pi/2)
```

Out[16]: 3.64750502212797

In [17]:
```
xnum = 4
ynum = np.linspace(0,np.pi,50)
plt.plot(ynum, f1num(xnum, ynum))
```

Out[17]: [<matplotlib.lines.Line2D at 0x19fc34ea250>]



Example

A falling object encounters a moving platform accelerating upwards:

1. Object $h_o(t) = h_0 - v_o t - \frac{1}{2}gt^2$
2. Platform $h_p(t) = v_p t + \frac{1}{2}qt^2$

Find the initial velocity v_0 such that when the object and platform collide, they are moving at the same speed.

Soln: We need to solve for v_0 and t in the 2 eqns:

1. $h_0(t) = h_p(t)$ and
2. $\frac{dh_0}{dt}(t) = -\frac{dh_p}{dt}(t)$

from these 2 equations we can define 2 functions eq1 and eq2

In [18]:
```python
t, h0, v0, g, vp, q = smp.symbols('t h_0 v_0 g v_p q',
                                  real=True, positive=True)

h0t = h0 - v0*t - smp.Rational(1,2)*g*t**2
hpt = vp*t + smp.Rational(1,2)*q*t**2
dh0dt = -v0 + g*t
dhpdt = vp + q*t

eq1 = h0t - hpt
eq2 = dh0dt + dhpdt

sol = smp.solve([eq1,eq2], [v0,t])
display(sol)
```

```
[(v_p + (g + q)*(-2*v_p/(3*(g + q)) + sqrt(2)*sqrt(3*g*h_0 + 3*h_0*q + 2*v_p**2)/(3
*(g + q))),
  -2*v_p/(3*(g + q)) + sqrt(2)*sqrt(3*g*h_0 + 3*h_0*q + 2*v_p**2)/(3*(g + q)))]
```

In [19]:
```python
v_initial, t_collision = sol[0]
v_initial
```

Out[19]:

$$v_p + (g + q)\left(-\frac{2v_p}{3(g+q)} + \frac{\sqrt{2}\sqrt{3gh_0 + 3h_0q + 2v_p^2}}{3(g+q)}\right)$$

velocities at the time of collision

In [20]:
```python
dh0dt.subs([(v0, v_initial), (t,t_collision)]).simplify()
```

Out[20]:

$$\frac{-gv_p - \frac{qv_p}{3} - \frac{q\sqrt{6gh_0+6h_0q+4v_p^2}}{3}}{g+q}$$

In [21]:
```python
dhpdt.subs([(v0, v_initial), (t,t_collision)]).simplify()
```

Out[21]:

$$\frac{gv_p + \frac{qv_p}{3} + \frac{q\sqrt{6gh_0+6h_0q+4v_p^2}}{3}}{g+q}$$

Calculus (1st year)

More depth discussion here: https://www.youtube.com/watch?v=-SdIZHPuW9o (https://www.youtube.com/watch?v=-SdIZHPuW9o)

In [22]:
```python
x = smp.symbols('x')
```

Limits

$$\lim_{x\to\pi} \sin(x/2 + \sin(x))$$

In [23]:
```python
smp.limit(smp.sin(x/2 + smp.sin(x)), x, smp.pi)
```

Out[23]: 1

Derivatives

$$\frac{d}{dx}\left(\frac{1+sinx}{1-cosx}\right)^2$$

In [24]:
```
smp.diff(((1+smp.sin(x))/(1-smp.cos(x)))**2, x)
```

Out[24]:
$$\frac{2\left(\sin\left(x\right)+1\right)\cos\left(x\right)}{\left(1-\cos\left(x\right)\right)^{2}}-\frac{2\left(\sin\left(x\right)+1\right)^{2}\sin\left(x\right)}{\left(1-\cos\left(x\right)\right)^{3}}$$

$$\frac{d}{dx}f(x+g(x))$$

In [25]:
```
f,g = smp.symbols('f g', cls=smp.Function)

g = g(x)
f = f(x+g)
display(f)

dfdx = smp.diff(f,x)
dfdx
```

$$f(x+g(x))$$

Out[25]:
$$\left(\frac{d}{dx}g(x)+1\right)\left.\frac{d}{d\xi_1}f(\xi_1)\right|_{\xi_1=x+g(x)}$$

In [26]:
```
dfdx.subs(g, smp.sin(x)).doit()  # doit() performs the derivative
```

Out[26]:
$$\left(\cos\left(x\right)+1\right)\left.\frac{d}{d\xi_1}f(\xi_1)\right|_{\xi_1=x+\sin\left(x\right)}$$

Integrations

Indefinite Integrals (Integration constant is not added)

$$\int\csc(x)\cot(x)dx$$

In [27]:
```
smp.integrate(smp.csc(x)*smp.cot(x), x)
```

Out[27]:
$$-\frac{1}{\sin\left(x\right)}$$

Definite Integrals

$$\int_0^{\ln(4)}\frac{e^x\,dt}{\sqrt{e^{2x}+9}}$$

In [28]:
```
smp.integrate(smp.exp(x)/smp.sqrt(smp.exp(2*x) + 9), (x, 0, smp.log(4)))
```

Out[28]:
$$-\operatorname{asinh}\left(\frac{1}{3}\right)+\operatorname{asinh}\left(\frac{4}{3}\right)$$

$$\int_1^t x^{10} e^x \, dx$$

In [29]:
```
t = smp.symbols('t')
smp.integrate(x**10*smp.exp(x), (x,1,t))
```

Out[29]: $(t^{10} - 10t^9 + 90t^8 - 720t^7 + 5040t^6 - 30240t^5 + 151200t^4 - 604800t^3 + 1814400t^2 - 3628800$

Examples

The hydrogen wave function is given by

$$\psi_{nlm} = R_{nl}(r) Y_l^m(\theta, \phi)$$

where

$$R_{nl}(r) = \sqrt{\left(\frac{2}{na}\right)^3 \frac{(n-l-1)!}{2n[(n+l)!]}} e^{-r/na} \left(\frac{2r}{na}\right)^l \left[L_{n-l-1}^{2l+1}(2r/na)\right]$$

Calculate:

The mean distance from the nucleus of the electron:

$$\langle r \rangle = \int R_{nl}^2 r^3 dr$$

The standard deviation in the distance from the nucleus of the electron:

$$\sigma = \sqrt{\langle r^2 \rangle - \langle r \rangle^2} = \sqrt{\left(\int_0^\infty R_{nl}^2 r^4 dr\right) - \left(\int_0^\infty R_{nl}^2 r^3 dr\right)^2}$$

Solution:

In [30]:
```
from sympy import assoc_laguerre
```

Define variables

In [31]:
```
r, a = smp.symbols('r a', real=True, positive=True)
n, l = smp.symbols('n l', integer=True, positive=True)
```

define $R_{nl}(r)$

In [32]:
```
R = smp.sqrt((2/(n*a))**3 * smp.factorial(n-l-1) / (2*n*smp.factorial(n+l))) * smp.ex
*(2*r/(n*a))**l * assoc_laguerre((n-l-1), (2*l+1), (2*r/(n*a)))
R
```

Out[32]:

$$\frac{2\left(\frac{2r}{an}\right)^l e^{-\frac{r}{an}} L_{-l+n-1}^{(2l+1)}\left(\frac{2r}{an}\right)\sqrt{(-l+n-1)!}}{a^{\frac{3}{2}} n^2 \sqrt{(l+n)!}}$$

ground state function

In [33]:
```
R_10 = R.subs([(n,1),(l,0)])
R_10
```

Out[33]:

$$\frac{2e^{-\frac{r}{a}}}{a^{\frac{3}{2}}}$$

Function to compute $\int_0^{\infty} R_{nl}^2 r^k dr$ for particular values of n, l and k,

In [34]:
```
def comp_int(n1,l1,k):
    Rn1l1 = R.subs([(n,n1),(l,l1)])
    return smp.integrate(Rn1l1**2 * r**k, (r,0, smp.oo))
```

Average r

In [35]:
```
r10avg = comp_int(1,0,3)  # ground state
r10avg
```

Out[35]:

$$\frac{3a}{2}$$

In [36]:
```
r53avg = comp_int(n1=5,l1=3,k=3)
r53avg
```

Out[36]:

$$\frac{63a}{2}$$

In [37]:
```
sigma10 = smp.sqrt(comp_int(1,0,4) - (comp_int(1,0,3))**2)
sigma10   # ground state
```

Out[37]:

$$\frac{\sqrt{3}a}{2}$$

In [38]:
```
sigma64 = smp.sqrt(comp_int(6,4,4) - (comp_int(6,4,3))**2)
sigma64
```

Out[38]:

$$11\sqrt{2}a$$

Graphs

In [39]:
```
def meandist(n1,l1=0):
    expr = comp_int(n1,l1,k=3)
    expr_f = smp.lambdify([a], expr)
    return expr_f(1)  # taking a=1
```

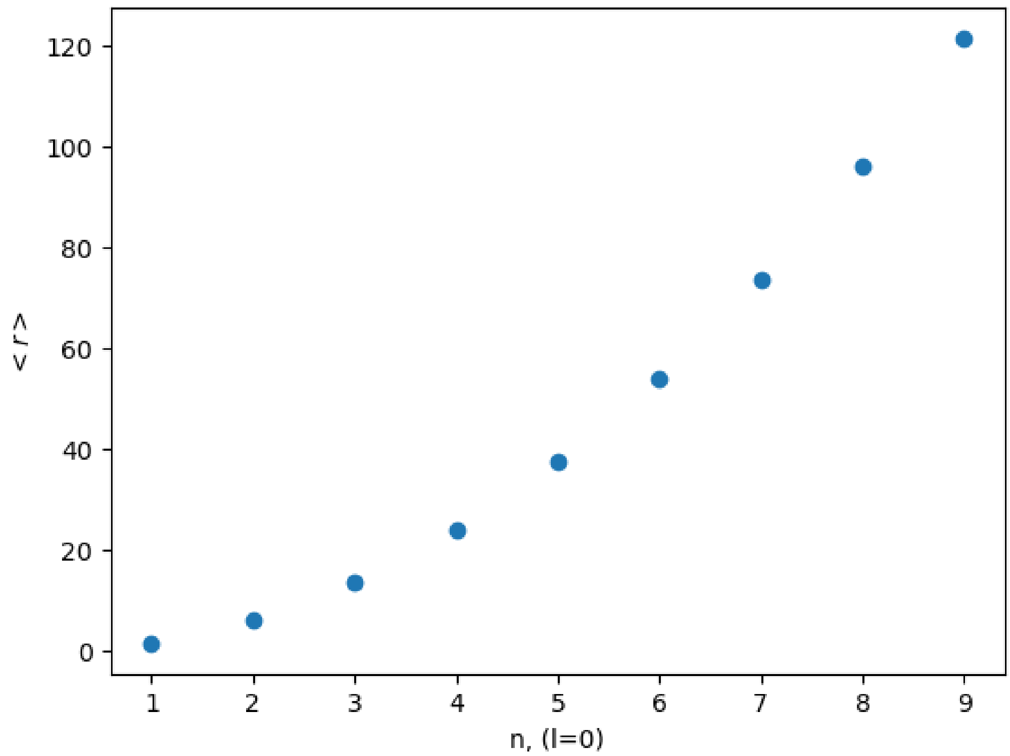
In [40]:
```
meandist(1)
```

Out[40]: 1.5

In [41]:
```
n2 = np.arange(1,10)
dist = [meandist(ni) for ni in n2]
```

In [42]:
```
plt.scatter(n2,dist)
plt.xlabel('n, (l=0)')
plt.ylabel('$< r >$')
```

Out[42]: `Text(0, 0.5, '$< r >$')`



Multivariable Calculus

In [43]:
```
x,y,z,t,u1,u2,u3,v1,v2,v3 = smp.symbols('x y z t u_1 u_2 u_3 v_1 v_2 v_3')
```

Vectors and Geometry

In [44]:
```
u = smp.Matrix([u1,u2,u3])
v = smp.Matrix([v1,v2,v3])
```

Matrix Operations

In [45]:
```
5*u - 2*v
```

Out[45]:

$$\begin{bmatrix} 5u_1 - 2v_1 \\ 5u_2 - 2v_2 \\ 5u_3 - 2v_3 \end{bmatrix}$$

In [46]:
```
u.dot(v)
```

Out[46]: $u_1 v_1 + u_2 v_2 + u_3 v_3$

In [47]:
```
u.cross(v)
```

Out[47]:
$$\begin{bmatrix} u_2 v_3 - u_3 v_2 \\ -u_1 v_3 + u_3 v_1 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

In [48]:
```
u.norm()
```

Out[48]:
$$\sqrt{|u_1|^2 + |u_2|^2 + |u_3|^2}$$

Projection - (See the 2nd year calculus notes)

Lines: $\vec{r}(t) = \vec{r}_0 + t\vec{v}$

In [49]:
```
r0 = smp.Matrix([1,1,1])
v = smp.Matrix([4,3,4])
r = r0 + t*v
r
```

Out[49]:
$$\begin{bmatrix} 4t + 1 \\ 3t + 1 \\ 4t + 1 \end{bmatrix}$$

Planes: $\vec{n} \cdot (P_0 - \langle x, y, z \rangle) = 0$

In [50]:
```
n = smp.Matrix([3,2,3])
P0 = smp.Matrix([2.2,3,2])
r = smp.Matrix([x,y,z])
n.dot(P0 - r)
```

Out[50]: $-3x - 2y - 3z + 18.6$

Vector Calculus

Vector derivatives

In [51]:
```
r = smp.Matrix([4*t,6*smp.cos(5*t),t**3])
smp.diff(r,t)
```

Out[51]:
$$\begin{bmatrix} 4 \\ -30\sin(5t) \\ 3t^2 \end{bmatrix}$$

Example: Find the angle between the velocity and acceleration as a function of time $\theta(t)$ and also find the angle at $t = 8s$.

In [52]:
```
v = smp.diff(r,t)
a = smp.diff(v,t)
theta = smp.acos(v.dot(a)/(v.norm()*a.norm()))
theta.simplify()
```

Out[52]:

$$\operatorname{acos}\left(\frac{3\left(t^{3}+125\sin\left(10t\right)\right)}{\sqrt{|t|^{2}+625|\cos\left(5t\right)|^{2}}\sqrt{9|t^{2}|^{2}+900|\sin\left(5t\right)|^{2}+16}}\right)$$

In [53]:
```
theta.subs(t,8).evalf()  # eval() evaluates a float value
```

Out[53]: 1.23941092042577

Vector Integrals

In [54]:
```
r = smp.Matrix([smp.exp(-t**3), smp.sin(t), 5*t**3 + 4*t])
I = smp.Integral(r,t)
I
```

Out[54]:

$$\int\left[\begin{matrix}e^{-t^{3}}\\ \sin\left(t\right)\\ 5t^{3}+4t\end{matrix}\right]dt$$

In [55]:
```
I.doit()  # performs the integration
```

Out[55]:

$$\left[\begin{matrix}\frac{\Gamma\left(\frac{1}{3}\right)\gamma\left(\frac{1}{3},t^{3}\right)}{9\Gamma\left(\frac{4}{3}\right)}\\ -\cos\left(t\right)\\ \frac{5t^{4}}{4}+2t^{2}\end{matrix}\right]$$

Some cases integrals can't be solved analytically. So we need to solve them numerically.

In [56]:
```
from scipy.integrate import quad_vec
```

In [57]:
```
r1 = smp.Matrix([smp.exp(-t**2)*smp.cos(t)**3, smp.exp(-t**4), 1/(3+t**2)])
I1 = smp.Integral(r1, (t,0,1))
I1
```

Out[57]:

$$\int_{0}^{1}\left[\begin{matrix}e^{-t^{2}}\cos^{3}\left(t\right)\\ e^{-t^{4}}\\ \frac{1}{t^{2}+3}\end{matrix}\right]dt$$

In [58]:
```
rf = smp.lambdify([t],r1)
rf(1)
```

Out[58]:
```
array([[0.05802511],
       [0.36787944],
       [0.25      ]])
```

In [59]:
```
quad_vec(rf, 0,1)  # integration and error
```

Out[59]:
```
(array([[0.53525785],
        [0.84483859],
        [0.30229989]]),
 3.5151979041265046e-14)
```

```
smp.integrate(r1, (t,0,1))
```

Result: (high processing time)

$$\begin{bmatrix} \int e^{-t^2}\cos^3(t)\,dt \\ \frac{\Gamma(\frac{1}{4})\gamma(\frac{1}{4},t^4)}{16\Gamma(\frac{5}{4})} \\ \frac{\sqrt{3}\operatorname{atan}\left(\frac{\sqrt{3}t}{3}\right)}{3} \end{bmatrix}$$

Arclength

$$L = \int_a^b \sqrt{dx^2 + dy^2 + dz^2} = \int_a^b \sqrt{(dx/dt)^2 + (dy/dt)^2 + (dz/dt)^2}dt$$

Find arclength of $\langle 0, 3t, 2t^2 \rangle$ from $t = 0$ to $t = 1$.

In [60]:
```
r2= smp.Matrix([0, 3*t, 2*t**2])
display(r2)

f1 = smp.diff(r2,t).norm()
L= smp.integrate(f1, (t,0,1))
display(L)
```

$$\begin{bmatrix} 0 \\ 3t \\ 2t^2 \end{bmatrix}$$

$$\frac{9\operatorname{asinh}\left(\frac{4}{3}\right)}{8} + \frac{5}{2}$$

Biot-Savart Law

The magnetic field at a point $\vec{r}$ of a current configuration is

$$\vec{B}(\vec{r}) = \frac{\mu_0}{4\pi}\int_t \frac{I\frac{d\vec{\ell}}{dt} \times (\vec{r} - \vec{\ell})}{|\vec{r} - \vec{\ell}|^3}dt$$

Where $\vec{r} = (x, y, z)$ and $\vec{\ell} = (f(t), g(t), h(t))$ is a 1D curve in space that gives location of the wire. (Here, t is a parameter, not time.)

Writing General Formulae

In [61]:
```
x,y,z,t,I, mu0 = smp.symbols('x y z t I \mu_0', real= True)
f,g,h = smp.symbols('f g h', cls= smp.Function)
f = f(t)
g = g(t)
h = h(t)
```

In [62]:
```
r = smp.Matrix([x, y, z])
l = smp.Matrix([f, g, h])
dldt = smp.diff(l)
```

In [63]:
```
l
```

Out[63]:

$$\begin{bmatrix} f(t) \\ g(t) \\ h(t) \end{bmatrix}$$

In [64]:
```
dBdt = (mu0* I/(4*smp.pi))* dldt.cross(r-l)/ ((r-l).norm())**3
dBdt
```

Out[64]:

$$\begin{bmatrix} \frac{I\mu_0\left(-(y-g(t))\frac{d}{dt}h(t)+(z-h(t))\frac{d}{dt}g(t)\right)}{4\pi\left(|x-f(t)|^2+|y-g(t)|^2+|z-h(t)|^2\right)^{\frac{3}{2}}} \\ \frac{I\mu_0\left((x-f(t))\frac{d}{dt}h(t)-(z-h(t))\frac{d}{dt}f(t)\right)}{4\pi\left(|x-f(t)|^2+|y-g(t)|^2+|z-h(t)|^2\right)^{\frac{3}{2}}} \\ \frac{I\mu_0\left(-(x-f(t))\frac{d}{dt}g(t)+(y-g(t))\frac{d}{dt}f(t)\right)}{4\pi\left(|x-f(t)|^2+|y-g(t)|^2+|z-h(t)|^2\right)^{\frac{3}{2}}} \end{bmatrix}$$

Question: Find magnetic field at a distance H above a ring of radius R flowing clockwise.

In [65]:
```
H, R = smp.symbols('H R', real = True)
```

In [66]:
```
dBdt1 = dBdt.subs([(f, R*smp.cos(t)), (g, R*smp.sin(t)), (h, 0),
                  (x,0), (y,0), (z,H)]).doit()
dBdt1.simplify()
dBdt1
```

Out[66]:

$$\begin{bmatrix} \frac{HIR\mu_0\cos(t)}{4\pi\left(H^2+R^2\right)^{\frac{3}{2}}} \\ \frac{HIR\mu_0\sin(t)}{4\pi\left(H^2+R^2\right)^{\frac{3}{2}}} \\ \frac{IR^2\mu_0}{4\pi\left(H^2+R^2\right)^{\frac{3}{2}}} \end{bmatrix}$$

In [67]:
```
B1 = smp.integrate(dBdt1, [t, 0, 2*smp.pi])
B1
```

Out[67]:

$$\begin{bmatrix} 0 \\ 0 \\ \frac{IR^2\mu_0}{2\left(H^2+R^2\right)^{\frac{3}{2}}} \end{bmatrix}$$

Question: Find magnetic field at a distance ρ from a wire of length L kept at the z axis.

In [68]:
```
L, rho, th = smp.symbols('L \\rho \\theta', real = True)
```

In [69]:
```
dBdt2 = dBdt.subs([(f, 0), (g, 0), (h, t),
                   (x, rho* smp.cos(th)), (y, rho* smp.sin(th)), (z,0)]).doit()
dBdt2.simplify()
dBdt2
```

Out[69]:

$$\begin{bmatrix} -\frac{I\mu_0\rho\sin(\theta)}{4\pi(\rho^2+t^2)^{\frac{3}{2}}} \\ \frac{I\mu_0\rho\cos(\theta)}{4\pi(\rho^2+t^2)^{\frac{3}{2}}} \\ 0 \end{bmatrix}$$

In [70]:
```
B2 = smp.integrate(dBdt2, [t, -L/2, L/2])
B2
```

Out[70]:

$$\begin{bmatrix} -\frac{IL\mu_0\sin(\theta)}{4\pi\rho^2\sqrt{\frac{L^2}{4\rho^2}+1}} \\ \frac{IL\mu_0\cos(\theta)}{4\pi\rho^2\sqrt{\frac{L^2}{4\rho^2}+1}} \\ 0 \end{bmatrix}$$

Partial/Directional Derivatives

In [71]:
```
x, y, z = smp.symbols('x y z')
```

Partial derivatives $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$ and $\frac{\partial^3 f}{\partial x y^2}$ of $f(x, y) = y^2 \sin(x + y)$

In [72]:
```
fxy = y**2 * smp.sin(x+y)
```

In [73]:
```
smp.diff(fxy, x)
```

Out[73]: $y^2 \cos(x + y)$

In [74]:
```
smp.diff(fxy, y)
```

Out[74]: $y^2 \cos(x + y) + 2y \sin(x + y)$

In [75]:
```
smp.diff(fxy, y, y, x)
```

Out[75]: $-y^2 \cos(x + y) - 4y \sin(x + y) + 2\cos(x + y)$

The Chain Rule

Suppose x,y and z are functions of t and $w = w(x, y, z)$. Find dw/dt.

In [76]:
```
t = smp.symbols('t')
x, y, z, w = smp.symbols('x y z w', cls = smp.Function)
x = x(t)
y = y(t)
z = z(t)
w = w(x,y,z)
w
```

Out[76]: $w(x(t), y(t), z(t))$

In [77]:
```
smp.diff(w,t)
```

Out[77]: $$\frac{d}{dx(t)}w(x(t), y(t), z(t))\frac{d}{dt}x(t) + \frac{d}{dy(t)}w(x(t), y(t), z(t))\frac{d}{dt}y(t) + \frac{d}{dz(t)}w(x(t), y(t), z(t))\frac{d}{dt}z(t)$$

In [78]:
```
w1 = x* smp.sin(y)* smp.exp(-z**2)
smp.diff(w1,t)
```

Out[78]: $$-2x(t)z(t)e^{-z^2(t)}\sin(y(t))\frac{d}{dt}z(t) + x(t)e^{-z^2(t)}\cos(y(t))\frac{d}{dt}y(t) + e^{-z^2(t)}\sin(y(t))\frac{d}{dt}x(t)$$

In [79]:
```
smp.diff(w1,t).subs([(x, 1/t**2), (y,14*t), (z, 2*t)]).doit()
```

Out[79]: $$-\frac{8e^{-4t^2}\sin(14t)}{t} + \frac{14e^{-4t^2}\cos(14t)}{t^2} - \frac{2e^{-4t^2}\sin(14t)}{t^3}$$

Multiple Integrals

In rare cases it can be solved symbolically.

$$\int_0^1 \int_0^{1-x^2} \int_3^{4-x^2-y^2} x^3 \, dz\,dy\,dx$$

In [80]:
```
x, y, z = smp.symbols('x y z')
f1 = x**3
smp.integrate(f1, (z,3, 4-x**2-y**2), (y,0,1-x**2), (x,0,1))
```

Out[80]: $$\frac{1}{30}$$

Lagrangian Mechanics

In [ ]:
```

```