Производная

Ввод [208]:

```
from sympy import Symbol, limit, oo, sin, sqrt, solve, factorial, symbols, cos, exp, asin
from sympy import *
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

Ввод [267]:

```
def tangent(y, x0):
    y0 = y.subs(x, x0)
    x1 = x0 + 1
    k = diff(y,x).subs(x,x0)
    y1 = y0 + k
    return Line((x0,y0), (x1,y1))
```

Пример 1

Ввод [6]:

```
x = symbols('x')
y = x*cos(x)
diff(x*cos(x), x)
```

Out[6]:

$$-x\sin(x) + \cos(x)$$

Пример 2

Ввод [9]:

```
diff(log(x), x, 3)
```

Out[9]:

$$\frac{2}{x^3}$$

Пример 3

Ввод [11]:

```
y = log(x**3,10)**3
diff(y,x,2).subs(x,10)
```

Out[11]:

$$-\frac{9(-6+\log(1000))\log(1000)}{100\log(10)^3}$$

Ввод [12]:

```
diff(y,x,2).subs(x,10).simplify()
```

Out[12]:

$$\frac{81(2-\log(10))}{100\log(10)^2}$$

Пример 4

Ввод [15]:

```
y = (x**2+x-6)/(x**2-10*x+25)
z = diff(y,x)
z
```

Out[15]:

$$\frac{9\log\left(x^3\right)^2}{x\log(10)^3}$$

Ввод [16]:

```
solve(z, x)
```

Out[16]:

```
[1, -1/2 - sqrt(3)*I/2, -1/2 + sqrt(3)*I/2]
```

Пример 5

Ввод [21]:

```
x = symbols('x')
y = symbols('y')
f = x**2 + y**2 - 4
idiff(f, y, x)
```

Out[21]:

$$-\frac{x}{y}$$

Ввод [23]:

```
f = x**2 + y**2 - 4
idiff(f, y, x, 2)
```

Out[23]:

$$\frac{-\frac{x^2}{y} - y}{y^2}$$

Ввод [25]:

```
idiff(f, y, x, 2).simplify()
```

Out[25]:

$$-\frac{x^2 + y^2}{y^3}$$

Пример 6

Ввод [29]:

```
t = symbols('t')
x = t - sin(t)
y = 1 - cos(t)
y_diff = diff(y,t)/diff(x,t)
y_diff
```

Out[29]:

$$\frac{\sin(t)}{1 - \cos(t)}$$

Ввод [30]:

```
y_2diff = diff(y_diff,t)/diff(x,t)
y_2diff.simplify()
```

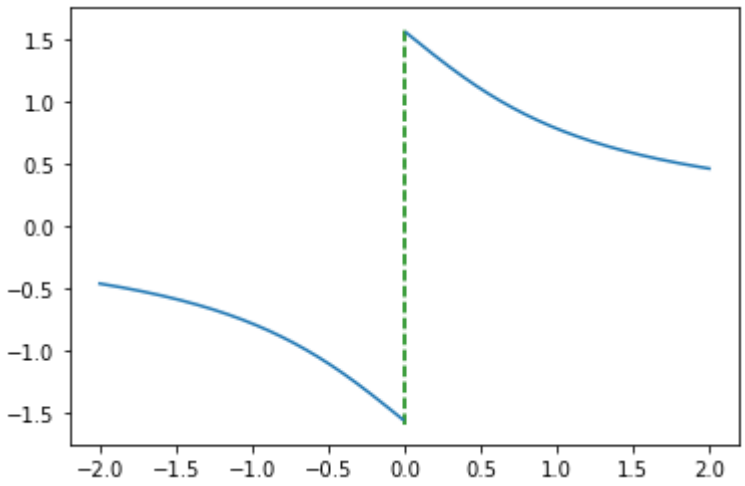
Out[30]:

$$-\frac{1}{(\cos(t) - 1)^2}$$

Пример 7

Ввод [33]:

```
x = np.linspace(-2,2,500)
x[(x>-0.01) & (x < 0.01)] = np.nan
y = np.arctan(1/x)
plt.plot(x,y)
plt.vlines(0, -1.6, 1.6, color='g', linestyles='dashed')
plt.show()
```



Ввод [37]:

```
x = symbols('x')
y = atan(1/x)
z = diff(y,x)
limit(z, x, 0, dir='+')
```

Out[37]:

$$-1$$

Ввод [38]:

```
limit(z, x, 0, dir='-')
```

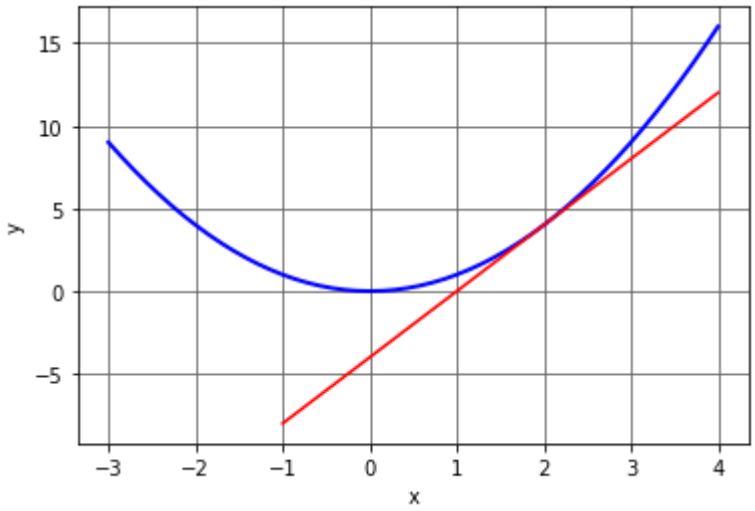
Out[38]:

$$-1$$

Пример 8

Ввод [55]:

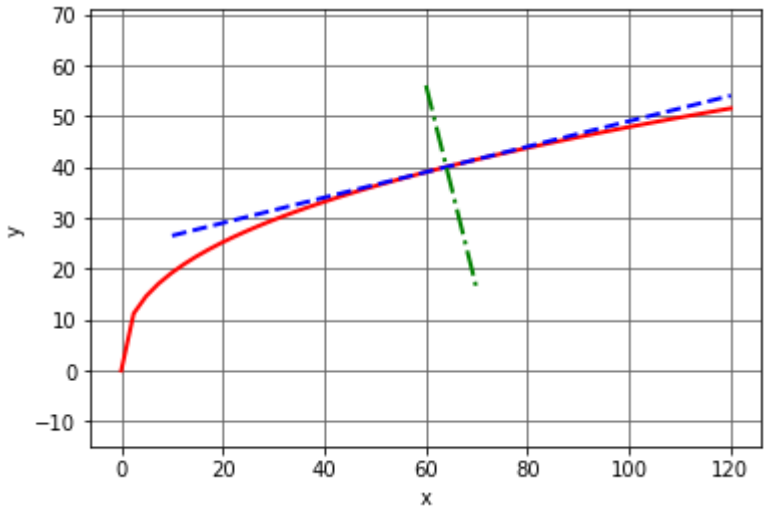
```
x = np.linspace(-3,4,50)
y1 = x**2
plt.plot(x,y1,lw=2,c='b')
x = np.linspace(-1,4,50)
y2 = 4*x - 4
plt.plot(x,y2,c='r')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Пример 9

Ввод [65]:

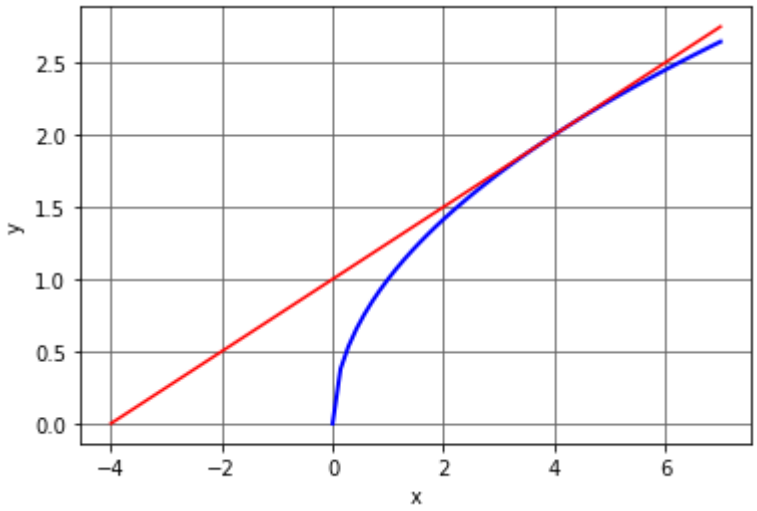
```
x = np.linspace(0,120,50)
y1 = 6*x**(1/3) + 2*x**(1/2)
plt.plot(x,y1,lw=2,c='r')
x = np.linspace(10,120,50)
y2 = x/4 + 24
plt.plot(x,y2,'--',lw=2,c='b')
x = np.linspace(60,70,50)
y3 = 296 - 4*x
plt.plot(x,y3,'-.',lw=2,c='g')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True, linestyle='-', color='0.4')
plt.axis('equal')
plt.show()
```



Пример 10

Ввод [72]:

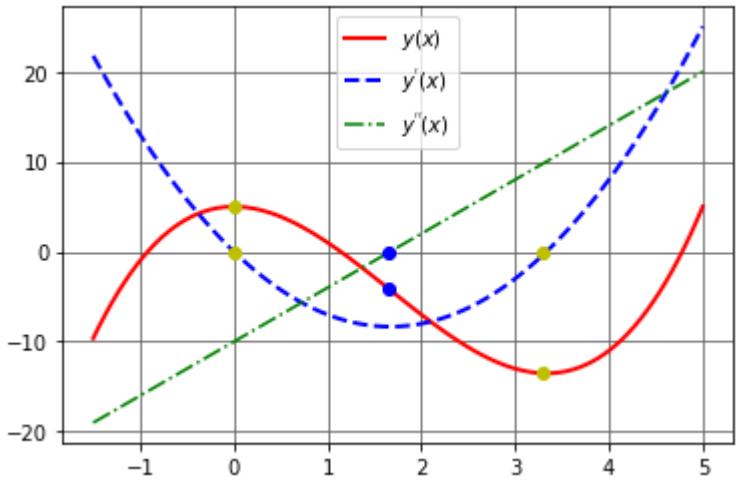
```
x = np.linspace(0,7,50)
y1 = np.sqrt(x)
plt.plot(x,y1,lw=2,c='b')
x = np.linspace(-4,7,50)
y2 = x/4 + 1
plt.plot(x,y2,c='r')
plt.xlabel('x')
plt.ylabel('y')
plt.grid (True, linestyle='-', color='0.4')
plt.show()
```



Исследование функции

Ввод [76]:

```
t = np.linspace(-1.5, 5, 100)
f = t**3 - 5*t**2 + 5
fd = 3*t**2 - 10*t
fdd = 6*t - 10
plt.plot(t,f,lw=2,color='red',label = "$y(x)$")
plt.plot(t,fd,'--',lw=2,color='b',label = "$y^{'}(x)$")
plt.plot(t,fdd, '-.', color='g',label = "$y^{''}(x)$")
plt.plot([0], [0], 'o', color='y')
plt.plot([0], [5], 'o', color='y')
plt.plot([3.3], [0], 'o', color='y')
plt.plot([3.3], [-13.4], 'o', color='y')
plt.plot([1.65], [0], 'o', color='b')
plt.plot([1.65], [-4], 'o', color='b')
plt.grid(True, linestyle='-', color='0.4')
plt.legend()
plt.show()
```



Пример 11

Ввод [80]:

```
x, y = symbols('x y')
solve(x**2 < 3)
```

Out[80]:

$$-\sqrt{3} < x \wedge x < \sqrt{3}$$

Пример 12

Ввод [81]:

```
solve(x**2 - y**2, x)
```

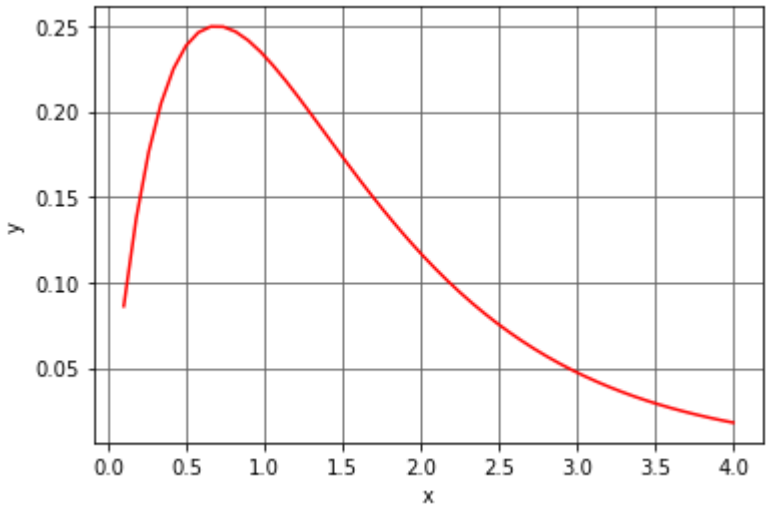
Out[81]:

```
[-y, y]
```

Пример 13

Ввод [83]:

```
f = lambda x: np.exp(-x) - np.exp(-2*x)
x = np.linspace(0.1,4,50)
plt.plot(x, f(x), 'r')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [91]:

```
f_max = lambda x: -(np.exp(-x) - np.exp(-2*x))
res = minimize(f_max, -2)
res
#print('x_max: %.3f f_max: %.3f' % (res.x, f(res.x)))
```

Out[91]:

```
      fun: -0.24999999999945666
 hess_inv: array([[1.98553383]])
      jac: array([-7.26431608e-07])
  message: 'Optimization terminated successfully.'
     nfev: 26
      nit: 12
     njev: 13
   status: 0
  success: True
        x: array([0.69314571])
```

Пример 14

Ввод [95]:

```
x = symbols('x')
y = x**3
x0 = solve(diff(y,x))[0]
print('x0: %.3f y(x0): %.3f' % (x0, y.subs(x, x0)))
```

```
x0: 0.000 y(x0): 0.000
```

Ввод [96]:

```
diff(y,x,2).subs(x,x0)
```

Out[96]:

$$0$$

Ввод [97]:

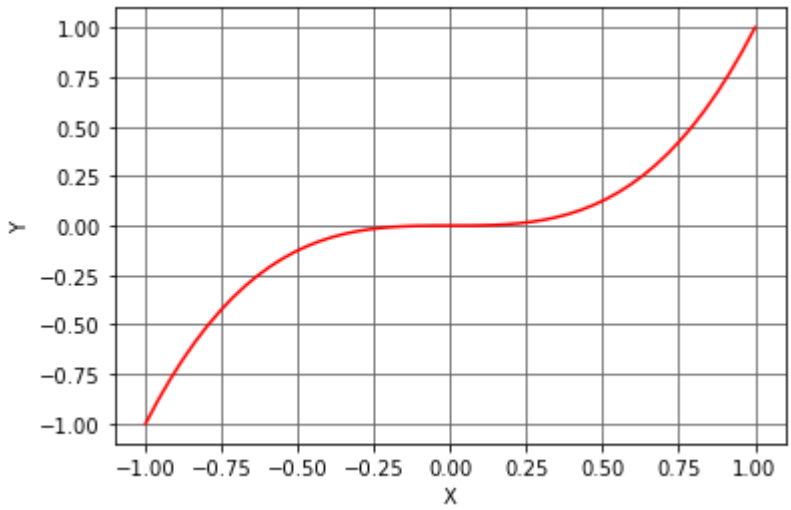
```
diff(y,x,3).subs(x,x0)
```

Out[97]:

$$6$$

Ввод [100]:

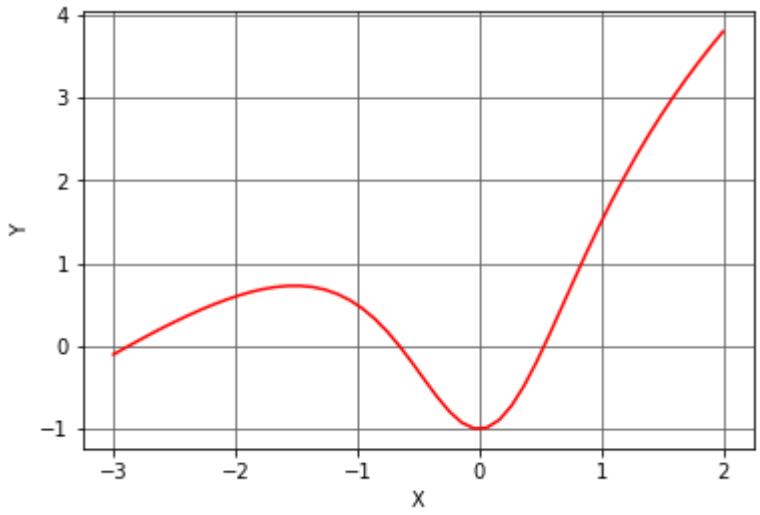
```
x = np.linspace(-1,1,50)
plt.plot(x, x**3, 'r')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Пример 15

Ввод [102]:

```
f = lambda x: (x**3+3*x**2-1) / (x**2+1)
x = np.linspace(-3,2,50)
plt.plot(x, f(x), 'r')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [103]:

```
res = minimize(f, 1)
print('x_min: %.3f fmin: %.3f' % (res.x, f(res.x)))
```

```
x_min: 0.000 fmin: -1.000
```

Ввод [105]:

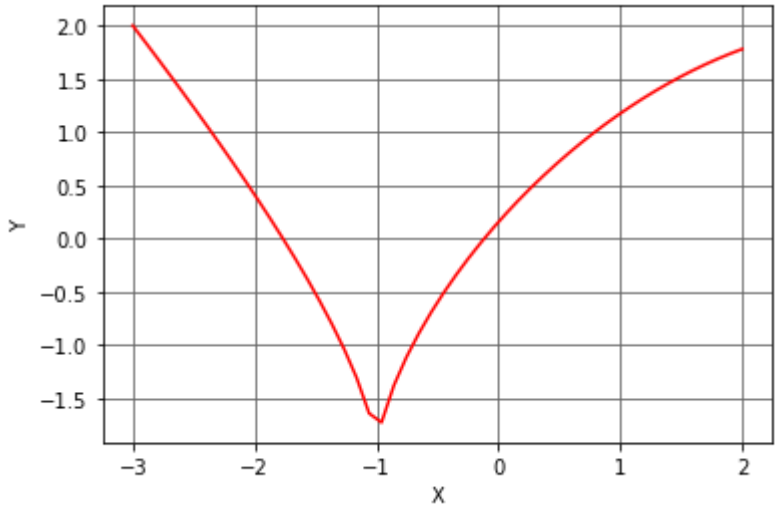
```
f_max = lambda x: -(x**3+3*x**2-1) / (x**2+1)
res = minimize(f_max, -2)
print('x_max: %.3f f max: %.3f' % (res.x, f(res.x)))
```

```
x_max: -1.513 f max: 0.731
```

Пример 16

Ввод [109]:

```
fun = lambda x: np.cbrt(2*(x+1)**2*(5-x)) - 2
x = np.linspace(-3, 3, 100)
plt.xlabel('X')
plt.ylabel('Y')
plt.plot(x, fun(x), 'r')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [111]:

```
res = minimize(fun, -1.5)
print('x_min: %.3f' % res.x)
```

```
x_min: -0.490
```

Ввод [112]:

```
res = minimize(fun, -1.001)
print('xmin: %.3f' % res.x)
```

```
xmin: -1.001
```

Ввод [113]:

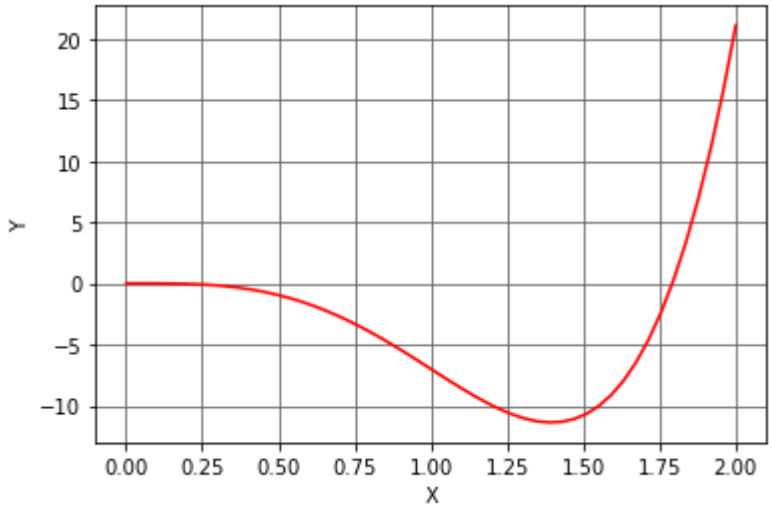
```
print('y(-3): %.3f y(3): %.3f' % (fun(-3), fun(3)))
```

```
y(-3): 2.000 y(3): 2.000
```

Пример 17

Ввод [116]:

```
f = lambda x: x**4 * (12*np.log(x) - 7)
x = np.linspace(0.001,2,50)
plt.plot(x, f(x), 'r')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [118]:

```
x = symbols('x')
y = x**4 * (12*log(x) - 7)
y_2deriv = diff(y,x,2)
y_2deriv
```

Out[118]:

$$144x^2\log(x)$$

Ввод [120]:

```
x_inflex = solve(y_2deriv, x)
x_inflex
```

Out[120]:

```
[0, 1]
```

Ввод [121]:

```
diff(y,x,3).subs(x, 1)
```

Out[121]:

$$144$$

Ввод [123]:

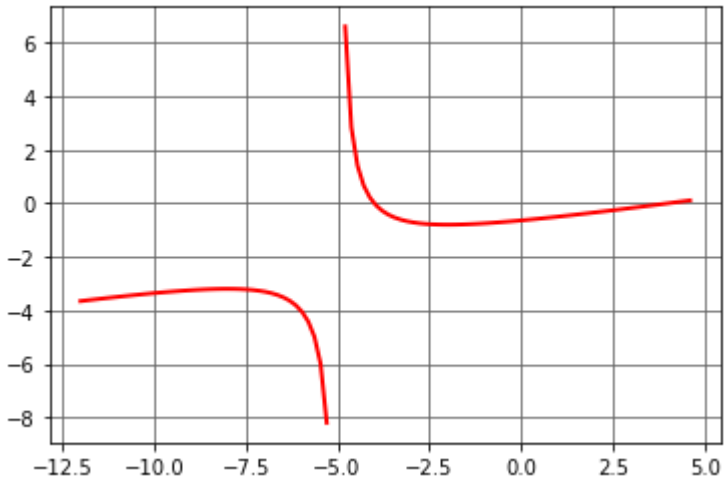
```
print('слева: %.1f справа: %.1f' % \
(y_2deriv.subs(x,0.9), y_2deriv.subs(x,1.1)))
```

```
слева: -12.3 справа: 16.6
```

Пример 18

Ввод [126]:

```
x = symbols('x')
y = (x**2-16)/(5*(x+5))
f = lambda x: (x**2-16)/(5*(x+5))
x = np.linspace(-12,4.6,100)
x[(x>-5.2) & (x < -4.8)] = np.nan
y = f(x)
plt.plot(x,y,lw=2,color='red')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [137]:

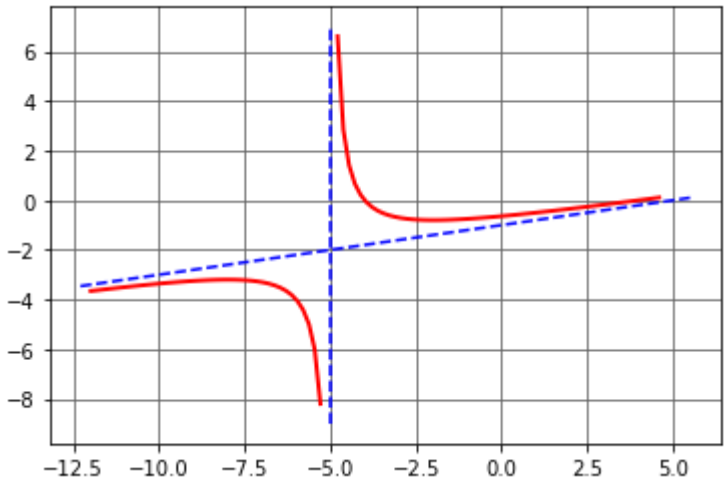
```
k = limit(y/x, x, oo)
k
```

Out[137]:

$$\left[.304761904761905 \quad 0.306779839677906 \quad 0.308883012171189 \quad 0.311077290143904 \quad 0.3133691\right.$$

Ввод [142]:

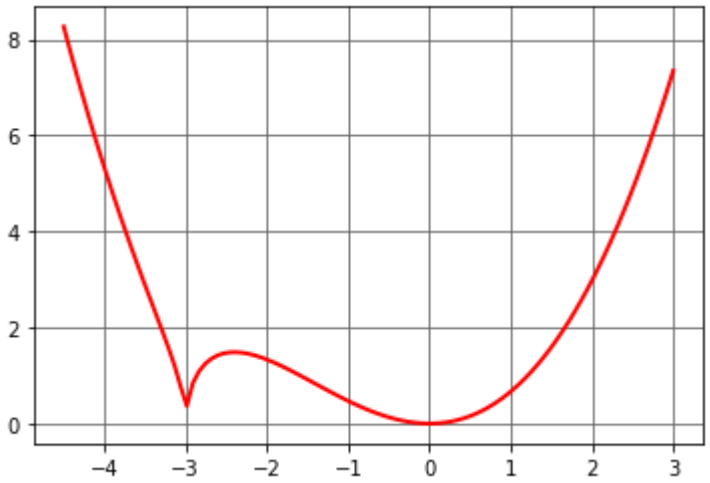
```
f = lambda x: (x**2-16)/(5*(x+5))
x = np.linspace(-12,4.6,100)
x[(x>-5.2) & (x < -4.8)] = np.nan
y = f(x)
plt.plot(x,y,lw=2,color='red')
x = np.linspace(-12.3,5.5,100)
y = x/5 - 1
plt.plot(x,y,'--',color='b')
plt.plot([-5,-5],[-9,7],'--',color='b')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Пример 19

Ввод [151]:

```
x = symbols('x')
y = x**2*sqrt(abs(x+3))/3
f = lambda x: x**2*np.sqrt(abs(x+3))/3
x = np.linspace(-4.5,3,100)
f_x = f(x)
plt.plot(x,f_x,lw=2,color='red')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Ввод [152]:

```
y.subs(x,0)
```

Out[152]:

$$\frac{x^{2}\sqrt{|x+3|}}{3}$$

Ввод [154]:

```
limit(y, x, oo)
```

Out[154]:

$$\infty$$

Ввод [156]:

```
k = limit(y/x, x, oo)
k
```

Out[156]:

$$\left[0.0740740740740741\,x^{2}\sqrt{|x+3|} - 0.0753424657534247\,x^{2}\sqrt{|x+3|} - 0.0766550522648084\,x\right.$$

Ввод [161]:

```
x = symbols('x')
y1 = x**2*sqrt(-x-3)/3
y1_ = diff(y1,x). simplify()
y1_
```

Out[161]:

$$-\frac{x\,(5x+12)}{6\sqrt{-x-3}}$$

Ввод [172]:

```
y2 = x**2*sqrt(x+3)/3
y12_ = diff(y1,x,2).simplify()
y12_
```

Out[172]:

$$\frac{\sqrt{-x-3}\left(5x^{2}+24x+24\right)}{4\left(x^{2}+6x+9\right)}$$

Ввод [166]:

```
y22_ = diff(y2,x,2).simplify()
y22_
```

Out[166]:

$$\frac{5\,x^{2} + 24\,x + 24}{4(x + 3)^{\frac{3}{2}}}$$

Ввод [173]:

```
xp1 = solve(y12_)
xp1
```

Out[173]:

```
[-12/5 - 2*sqrt(6)/5, -12/5 + 2*sqrt(6)/5]
```

Ввод [176]:

```
diff(y1, x, 3).subs(x,xp1[0]).evalf(5)
```

Out[176]:

-10.465

Ввод [179]:

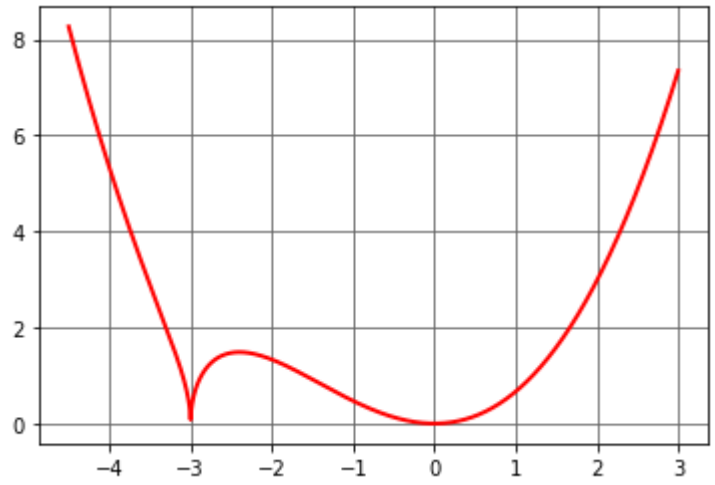
```
diff(y1, x, 3).subs(x,xp1[1]).evalf(4)
```

Out[179]:

$1.234i$

Ввод [181]:

```
f = lambda x: x**2*np.sqrt(abs(x+3))/3
x = np.linspace(-4.5,3,2000)
f_x = f(x)
plt.plot(x,f_x,lw=2,color='red')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Пример 20

Ввод [182]:

```
x, y = symbols("x y")
z = x*y**2 + exp(-x)
```

Ввод [184]:

```
diff(z, x, 2)
```

Out[184]:

$$e^{-x}$$

Ввод [185]:

```
diff(z, y, 2)
```

Out[185]:

$$2x$$

Пример 21

Ввод [186]:

```
x, y = symbols('x y')
z = sin(x)*cos(y)
diff(z, x, 2, y)
```

Out[186]:

$$\sin(y)\sin(x)$$

Пример 22

Ввод [187]:

```
x,y = symbols('x y')
z = 5*log(x**2 + y**2)
z_x = diff(z,x).subs({x:1, y:2})
z_y = diff(z,y).subs({x:1, y:2})
grad_f = (z_x, z_y)
grad_f
```

Out[187]:

```
(2, 4)
```

Пример 23

Ввод [191]:

```
x,y = symbols('x y')
z = x**2 + x*y +7
z_x = diff(z,x).subs({x:1, y:-1})
z_y = diff(z,y).subs({x:1, y:-1})
grad_f = (z_x, z_y)
grad_f
```

Out[191]:

(1, 1)

Пример 24

Ввод [192]:

```
l = Point(3,4)
l_n = l.distance(Point(0,0))
cos_a = l.x/l_n
cos_b = l.y/l_n
x,y = symbols('x y')
z = x**2 + y**2
z_x = diff(z,x).subs({x:1, y:1})
z_y = diff(z,y).subs({x:1, y:1})
z_l = z_x*cos_a + z_y*cos_b
z_l
```

Out[192]:

$$\frac{14}{5}$$

Пример 25

Ввод [197]:

```
def tangent_plane(F,M):
    F_diff_x = diff(F,x).subs({x:M.x,y:M.y,z:M.z})
    F_diff_y = diff(F,y).subs({x:M.x,y:M.y,z:M.z})
    F_diff_z = diff(F,z).subs({x:M.x,y:M.y,z:M.z})

    n = Point(F_diff_x, F_diff_y, F_diff_z)

    p = Plane(M, normal_vector=n).equation()

    K = Point(M.x+n.x, M.y+n.y, M.z+n.z)
    l_n = Line(M, K).arbitrary_point()
    return p, ln
```

Ввод [202]:

```
x, y, z = symbols('x y z')
F = x**2 + y**2 + z**2 - 9
M = Point(1,1,1)
p, l_n = tangent_plane(F,M)
```

Ввод [203]:

```
p
```

Out[203]:

$2x + 2y + 2z - 6$

Пример 26

Ввод [205]:

```
z = lambda w: (w[0]-1)**2 + (w[1]-3)**4
res = minimize(z, (0, 0))
res.x
```

Out[205]:

```
array([0.99999999, 2.98725136])
```

Ввод [206]:

```
z((1,3)) < z((0.999,3.001))
```

Out[206]:

```
True
```

Ввод [207]:

```
z((1,3))
```

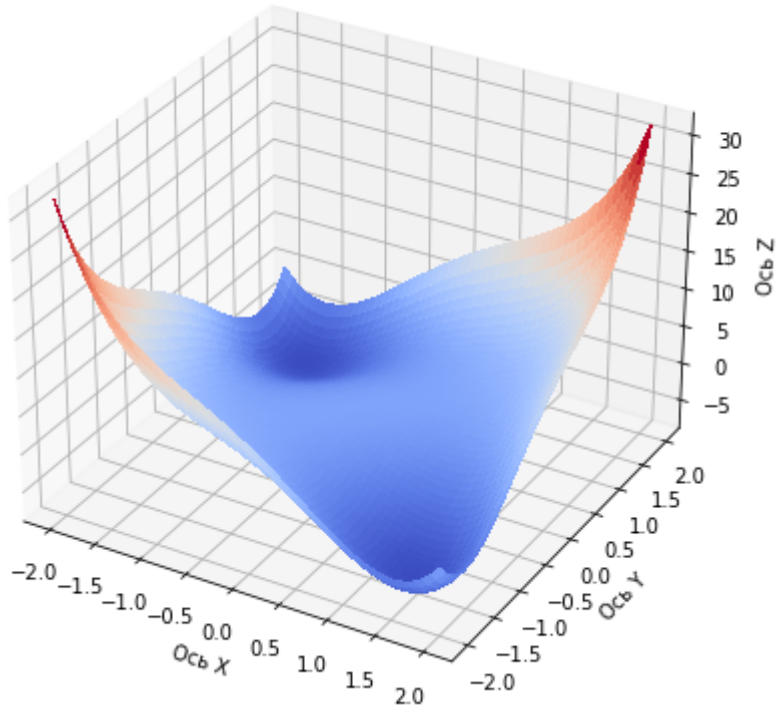
Out[207]:

```
0
```

Пример 27

Ввод [211]:

```
z = lambda w: w[0]**4 + w[1]**4 - 2*w[0]**2 +4*w[0]*w[1] - 2*w[1]**2

fig = plt.figure(figsize=(7,7))
axes = fig.gca(projection='3d')
y = x = np.linspace(-2, 2, 50)
x, y = np.meshgrid(x, y)
Z = z((x,y))
surf = axes.plot_surface(x, y, Z, cmap='coolwarm',linewidth=0, antialiased=False)

axes.set_xlabel('Ось X')
axes.set_ylabel('Ось Y')
axes.set_zlabel('Ось Z')
plt.show()
```



Ввод [212]:

```
res = minimize(z, (1, -1))
res.x
```

Out[212]:

```
array([ 1.41421356, -1.41421356])
```

Ввод [213]:

```
z(res.x)
```

Out[213]:

```
-8.0
```

Ввод [214]:

```
res = minimize(z, (-1, 1))
res.x
```

Out[214]:

```
array([-1.41421356,  1.41421356])
```

Ввод [216]:

```
z(res.x)
```

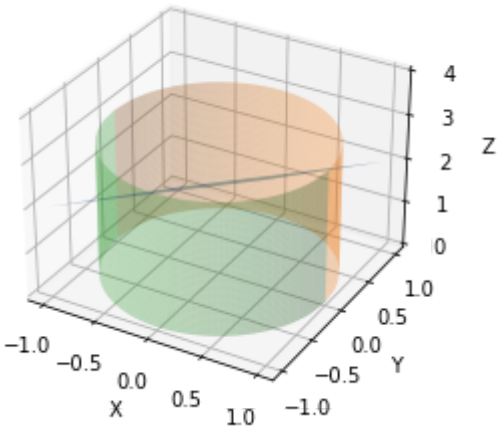
Out[216]:

```
-8.0
```

Пример 28

Ввод [219]:

```
f = lambda w: w[0] - w[1] + 2
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x = np.linspace(-1, 1, 50)
y = np.linspace(-1, 1, 50)

x, y = np.meshgrid(x, y)
z1 = f((x,y))
ax.plot_surface(x, y, z1, alpha=0.4)

x = np.linspace(-1, 1, 100)
z = np.linspace(0, 3, 100)
xc, zc = np.meshgrid(x, z)
yc = np.sqrt(1-xc**2)
ax.plot_surface(xc, yc, zc, alpha=0.3)
ax.plot_surface(xc, -yc, zc, alpha=0.3)
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
plt.show()
```



Ввод [221]:

```
cons = ({'type': 'eq', 'fun': lambda w: w[0]**2 + w[1]**2 - 1})

bnds = ((None, None), (None, None))

res = minimize(f, (-0.5, 0.5), bounds=bnds, constraints=cons)

res.x
```

Out[221]:

```
array([-0.70710679,  0.70710677])
```

Ввод [222]:

```
f_max = lambda w: -(1.5*w[0] - w[1] + 1)
cons = ({'type': 'eq', 'fun': lambda w: w[0]**2 + w[1]**2 - 1})
bnds = ((None, None), (None, None))
res = minimize(f_max, (0.5, -0.5), bounds=bnds,constraints=cons)

res.x
```

Out[222]:

```
array([ 0.83205051, -0.55469991])
```

Пример 29

Ввод [229]:

```
x,y = symbols('x y')
z = 4.5*x**(0.33) * y**(0.66)
z_x = diff(z, x)
z_y = diff(z, y)

E_x = (x/z)*z_x
E_y = (y/z)*z_y
print('E_x: %.2f E_y: %.2f' % (E_x, E_y))
```

```
E_x: 0.33 E_y: 0.66
```

Пример 30

Ввод [230]:

```
K,V0 = symbols('K V0')
V = V0*log(5+K**2)

Vprim2 = diff(V,K,2)

Vprim3 = diff(V,K,3)

s = solve(Vprim2,K)
s
```

Out[230]:

```
[-sqrt(5), sqrt(5)]
```

Ввод [233]:

```
Vprim3.subs(K,s[1])
```

Out[233]:

$$-\frac{\sqrt{5}V_0}{25}$$

Примеры решения задач

Вычислить y' для функции $xcos(lnx) + sin(lnx))$

Ввод [237]:

```
x = symbols('x')
y = x*(cos(log(x))+sin(log(x)))
diff(y,x)
```

Out[237]:

$$\mathrm{x}\left(-\frac{\sin(\log(\mathrm{x}))}{\mathrm{x}}+\frac{\cos(\log(\mathrm{x}))}{\mathrm{x}}\right)+\sin(\log(\mathrm{x}))+\cos(\log(\mathrm{x}))$$

Ввод [238]:

```
diff(y,x).simplify()
```

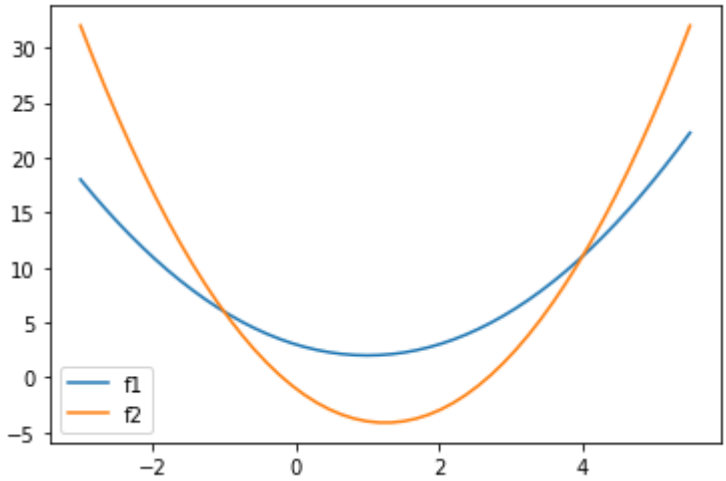
Out[238]:

$$2\cos(\log(\mathrm{x}))$$

Решить уравнение $y'(x) = 0$, где $y(x) = max\{x^2 - 2x + 3; 2x^2 - 5x - 1\}$

Ввод [248]:

```
f1 = lambda x: x**2-2*x+3
f2 = lambda x: 2*x**2-5*x-1
x = np.linspace(-3, 5.5, 50)
y1 = f1(x)
plt.plot(x,y1, label = "f1")
y2 = f2(x)
plt.plot(x,y2, label = "f2")
plt.legend()
plt.show()
```



Ввод [243]:

```
f1(-1) == f2(-1)
```

Out[243]:

```
True
```

Ввод [244]:

```
f1(4) == f2(4)
```

Out[244]:

```
True
```

Ввод [251]:

```
x = symbols('x')
f1 = x**2-2*x+3
f2 = 2*x**2-5*x-1

y_diff1 = diff(f2,x)
y_diff1
```

Out[251]:

$4x - 5$

Ввод [252]:

```
y_diff2 = diff(f1,x)
y_diff2
```

Out[252]:

$2x - 2$

Показать, что функция $y = xsinx$ удовлетворяет уравнению $\frac{y'}{cosx} - x = tgx$

Ввод [254]:

```
x, y = symbols('x y')
y = x*sin(x)
yprim = diff(y, x)
f = yprim/cos(x) - x
f.simplify()
```

Out[254]:

$\tan(x)$

Написать уравнения касательных к графику функции $y = (x^2 + 1)(x - 2)$ в точках её пересечения с осями координат.

Ввод [268]:

```
x = symbols('x', real=True)
y = (x**2 + 1)*(x - 2)
```

Ввод [269]:

```
tangent(y, 0).equation()
```

Out[269]:

$-x + y + 2$

Ввод [270]:

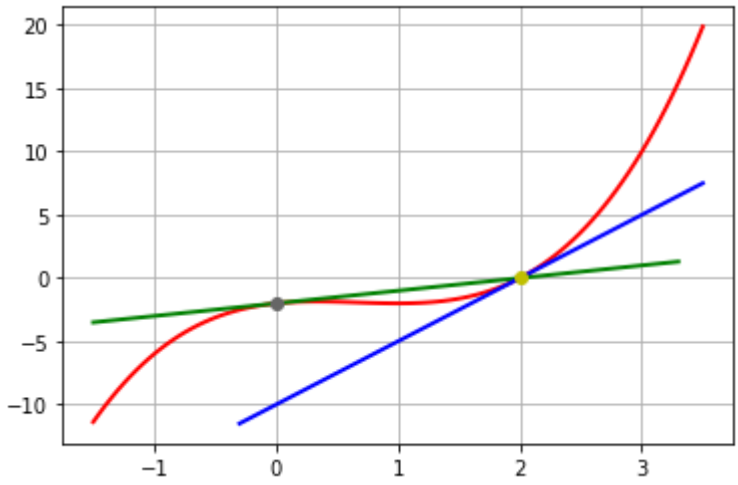
```
xp = solve(y, x)
tangent(y, xp[0]).equation()
```

Out[270]:

$-5x + y + 10$

Ввод [274]:

```
x = np.linspace(-1.5,3.5,100)
y = (x**2 + 1)*(x - 2)
plt.plot(x, y, lw=2, color='red')
x = np.linspace(-1.5,3.3,100)
y1 = x - 2
plt.plot(x, y1, lw=2, color='green')

x = np.linspace(-0.3,3.5,100)
y2 = 5*x - 10
plt.plot(x, y2, lw=2, color='blue')
plt.plot([0], [-2], 'o', color='0.4')
plt.plot([2], [0], 'o', color='y')
plt.grid(True)
plt.show()
```



При каком значении параметра a парабола $y = ax^2$ касается кривой $y = lnx$?

Ввод [277]:

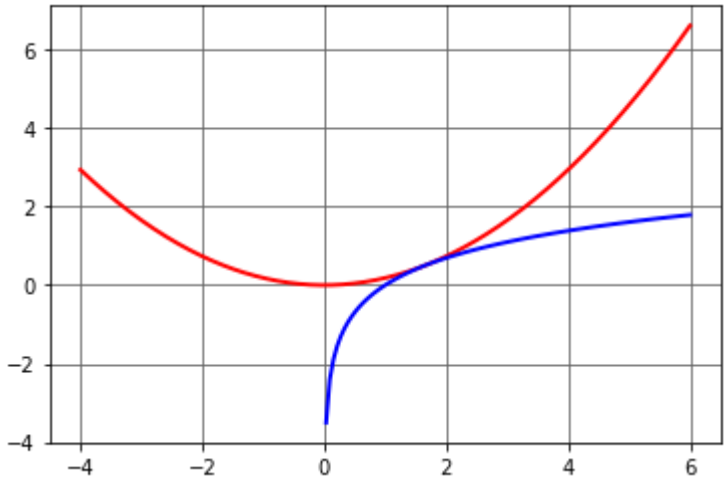
```
x, a, x0 = symbols('x a x0')

y1 = a*x**2
y2 = log(x)

y1_diff = diff(y1,x).subs(x,x0)
y2_diff = diff(y2,x).subs(x,x0)

y1_0 = y1.subs(x,x0)
y2_0 = y2.subs(x,x0)

solve([y1_0-y2_0, y1_diff-y2_diff], [x0, a])
```

Out[277]:

```
[(exp(1/2), exp(-1)/2)]
```

Ввод [280]:

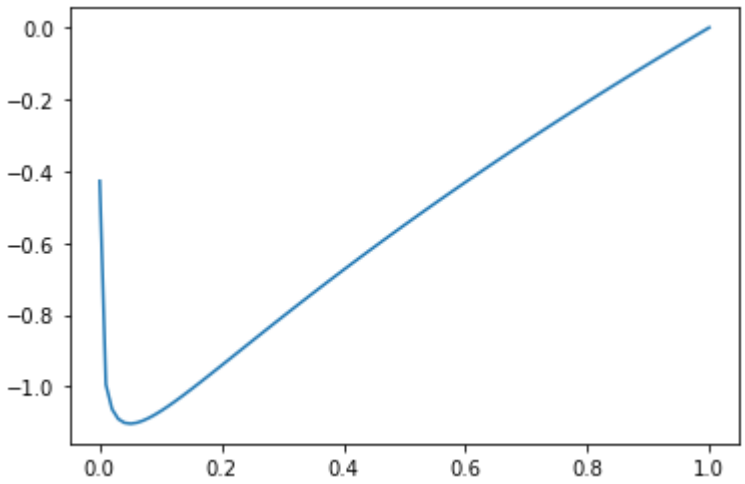
```
x = np.linspace(-4, 6, 500)
y = x**2/(2*np.exp(1))
plt.plot(x, y, lw=2, c='r')
x = np.linspace(0.03, 6, 100)
y = np.log(x)
plt.plot(x, y, lw=2, c='b')
plt.grid(True, linestyle='-', color='0.4')
plt.show()
```



Исследовать на экстремум функцию $y = \sqrt[3]{x} \times lnx.$

Ввод [281]:

```
f = lambda x: (x**(1/3))*np.log(x)

x = np.linspace(0.0001,1,100)
y = f(x)
plt.plot(x, y)
plt.show()
```



Ввод [282]:

```
res = minimize(f, 0.01)
print('xmin: %.4f y(x_min): %.3f' % (res.x, f(res.x)))
```

```
xmin: 0.0498 y(x_min): -1.104
```

Ввод [284]:

```
x = symbols('x')
y = x**(1/3) * log(x)
x_min = solve(diff(y,x))[0]
print('x_min: %.4f y(x_min) %.3f' % (x_min, y.subs(x, x_min)))
```

```
x_min: 0.0498 y(x_min) -1.104
```

Найти производную функции $w = \frac{x^2}{2} + \frac{y^2}{9} - z^2$ в точке $A(1; 2)$ по направлению радиус-вектора этой точки.

Ввод [285]:

```
l = Point(2,3,1)

l_n = l.distance(Point(0,0,0))

cos_a = l.x/l_n
cos_b = l.y/l_n
cos_c = l.z/l_n
```

Ввод [288]:

```
x,y,z = symbols('x y z')
w = x**2/2 + y**2/9 - z**2

w_x = diff(w,x).subs({x:2, y:3, z:1})
w_y = diff(w,y).subs({x:2, y:3, z:1})
w_z = diff(w,z).subs({x:2, y:3, z:1})
```

Ввод [289]:

```
w_x = diff(w,x).subs([(x,2),(y,3),(z,1)])
w_y = diff(w,y).subs([(x,2),(y,3),(z,1)])
w_z = diff(w,z).subs([(x,2),(y,3),(z,1)])

w_l = w_x*cos_a + w_y*cos_b + w_z*cos_c
w_l
```

Out[289]:

$$\frac{2\sqrt{14}}{7}$$

Найти экстремумы функции $z = x^2 - 4xy - 2y^2 + 8x$

Ввод [303]:

```
def critical_points(z):

    z_x = diff(z,x)
    z_y = diff(z,y)

    cr_point = solve([z_x, z_y], [x, y], dict=True)

    A = diff(z,x,2)
    B = diff(z,x,y)
    C = diff(z,y,2)

    D = A*C - B**2
    return cr_point, A, D
```

Ввод [305]:

```
def suff_indic(A, D, cr_point):
    A0 = A.subs(cr_point)
    D0 = D.subs(cr_point)
    return D0, A0
```

Ввод [304]:

```
x, y = symbols('x y')
z = x**2 - 4*x*y - 2*y**2 + 8*x
cr_point, A, D = critical_points(z)
cr_point
```

Out[304]:

```
[{x: -4/3, y: 4/3}]
```

Ввод [306]:

```
D0, A0 = suff_indic(A, D, cr_point[0])
D0, A0
```

Out[306]:

```
(-24, 2)
```

Зависимость между себестоимостью продукции C и объёмом её производства Q выражается формулой $C(Q) = 80 - 0,38Q$. Определить эластичность себестоимости при выпуске продукции $Q = 20$ ден. ед.

Ввод [308]:

```
Q = symbols('Q')
c = 80 - 0.38*Q
Dprim = diff(c,Q)
E = (Q*Dprim/c).subs(Q,20)
S(E).n(3)
```

Out[308]:

-0.105

Функция спроса D и предложения S от цены p имеют вид: $D(p) = 40 - 1,3p$, $S(p) = 20 + 1,2p$. Найти эластичность спроса в точке равновенсой цены.

Ввод [312]:

```
p = symbols('p')
D = 40 - 1.3*p
S = 20 + 1.2*p
p0 = solve(D-S,p)
p0[0].n(2)
```

Out[312]:

8.0

Ввод [314]:

```
Dprim = diff(D,p)
E = (p*Dprim/D).subs(p,p0[0])
E.n(3)
```

Out[314]:

-0.351

Решение собственной задачи с использованием производных

Модель Блэка-Шоулза. Уравнения Блэка-Шоулза произвели революцию в ценообразовании опционов, когда в 1973 году Мрайон Шоулз и Фишер Блэк опубликовали свою работу.

При рассмотрении приведенных ниже формул необходимо учитывать ряд важных допущений:

1. Процентная ставка известна и постоянна во времени.
2. Акции следуют случайному блужданию в непрерывном времени, дисперсия путей цены акции следует логнормальному распределению.
3. Волатильность постоянна
4. Акции не выплачивают дивиденды (однако их можно модифицировать для включения дивидендов).
5. Опцион может быть исполнен только по истечении срока действия, т.е. это европейский тип опциона.
6. Отсутствие транзакционных издержек, т.е. комиссий за продажу в короткую позицию и т.д.
7. Возможна дробная торговля, т.е. мы можем купить/продать 0,x любой акции.

Формулы Блэка-Шоулза для акций, не выплачивающих дивиденды

$$Call = S_0 N(d_1) - N(d_2) K e^{-rT}$$

$$Put = N(-d_2) K e^{-rT} - N(-d_1) S_0$$

$$d_1 = \frac{ln(\frac{S}{K}) + (r + \frac{\sigma^2}{2})T}{\sigma\sqrt{T}} d_2 = d_1 - \sigma\sqrt{T}$$

S : текущая цена актива

K: цена исполнения опциона

r: безрисковая ставка

T : время до истечения срока опциона

σ: годовая волатильность доходности актива

N(x): кумулятивная функция распределения для стандартного нормального распределения, показанного ниже.

$$N(x) = \int_{-\infty}^{x} \frac{e^{-x^2/2}}{\sqrt{2\pi}}$$

Ввод [357]:

```
pip install pandas-datareader
```

```
Collecting pandas-datareader
  Downloading pandas_datareader-0.10.0-py3-none-any.whl (109 kB)
Requirement already satisfied: requests>=2.19.0 in e:\anaconda\lib\site-pa
ckages (from pandas-datareader) (2.25.1)
Requirement already satisfied: pandas>=0.23 in e:\anaconda\lib\site-packag
es (from pandas-datareader) (1.2.4)
Requirement already satisfied: lxml in e:\anaconda\lib\site-packages (from
pandas-datareader) (4.6.3)
Requirement already satisfied: numpy>=1.16.5 in e:\anaconda\lib\site-packa
ges (from pandas>=0.23->pandas-datareader) (1.20.1)
Requirement already satisfied: pytz>=2017.3 in e:\anaconda\lib\site-packag
es (from pandas>=0.23->pandas-datareader) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in e:\anaconda\lib\s
ite-packages (from pandas>=0.23->pandas-datareader) (2.8.1)
Requirement already satisfied: six>=1.5 in e:\anaconda\lib\site-packages
(from python-dateutil>=2.7.3->pandas>=0.23->pandas-datareader) (1.15.0)
Requirement already satisfied: idna<3,>=2.5 in e:\anaconda\lib\site-packag
es (from requests>=2.19.0->pandas-datareader) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in e:\anaconda\lib\site-
packages (from requests>=2.19.0->pandas-datareader) (2022.12.7)
Requirement already satisfied: chardet<5,>=3.0.2 in e:\anaconda\lib\site-p
ackages (from requests>=2.19.0->pandas-datareader) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in e:\anaconda\lib\si
te-packages (from requests>=2.19.0->pandas-datareader) (1.26.4)
Installing collected packages: pandas-datareader
Successfully installed pandas-datareader-0.10.0
Note: you may need to restart the kernel to use updated packages.
```

Ввод [358]:

```
import numpy as np
from scipy.stats import norm
import pandas_datareader.data as web
import pandas as pd
import datetime as dt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Ввод [348]:

```
N = norm.cdf

def BS_CALL(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + sigma**2/2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return S * N(d1) - K * np.exp(-r*T)* N(d2)

def BS_PUT(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + sigma**2/2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma* np.sqrt(T)
    return K*np.exp(-r*T)*N(-d2) - S*N(-d1)
```

В этом разделе мы рассмотрим влияние изменения входных параметров на стоимость контрактов и опционов.

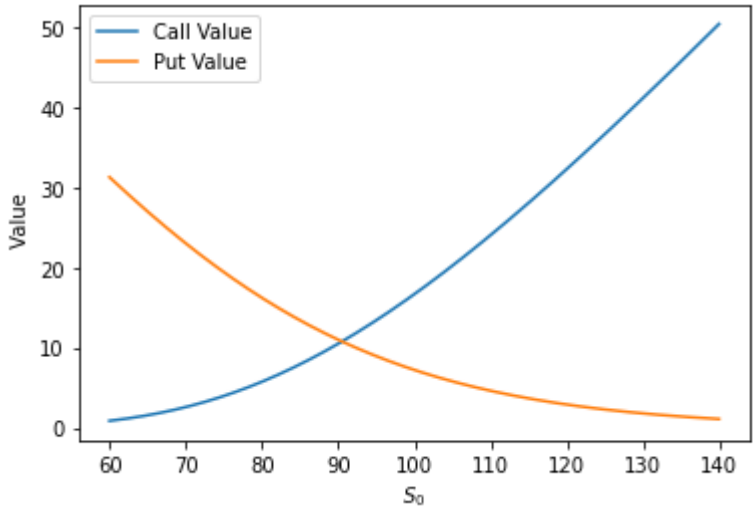
Влияние S на стоимость опциона. Здесь мы будем поддерживать постоянными все переменные, кроме текущей цены акций S, и рассмотрим, как меняется стоимость контрактов и опционов.

Ввод [350]:

```
K = 100
r = 0.1
T = 1
sigma = 0.3

S = np.arange(60,140,0.1)

calls = [BS_CALL(s, K, T, r, sigma) for s in S]
puts = [BS_PUT(s, K, T, r, sigma) for s in S]
plt.plot(S, calls, label='Call Value')
plt.plot(S, puts, label='Put Value')
plt.xlabel('$S_0$')
plt.ylabel(' Value')
plt.legend()
```

Out[350]:

```
<matplotlib.legend.Legend at 0x212f3c1f910>
```



Влияние σ на стоимость по Блэку-Шоулзу

Как и следовало ожидать, при постоянстве других переменных и увеличении параметра волатильности стоимость контрактов и опционов увеличивается линейно, как показано ниже.

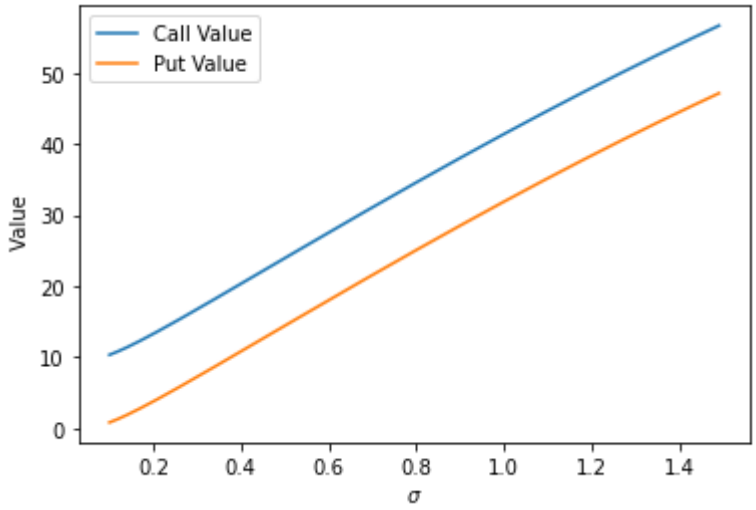
Чтобы понять, почему стоимость контрактов строго больше стоимости опционов в зависимости от волатильности, изменим процентную ставку r до 0 и заметим, что кривые точно совпадают. Вместо того чтобы строить графики влияния на процентные ставки, мы можем сделать вывод, что увеличение процентных ставок увеличивает стоимость контрактов и уменьшает стоимость опционов.

Ввод [351]:

```
K = 100
r = 0.1
T = 1
Sigmas = np.arange(0.1, 1.5, 0.01)
S = 100

calls = [BS_CALL(S, K, T, r, sig) for sig in Sigmas]
puts = [BS_PUT(S, K, T, r, sig) for sig in Sigmas]
plt.plot(Sigmas, calls, label='Call Value')
plt.plot(Sigmas, puts, label='Put Value')
plt.xlabel('$\sigma$')
plt.ylabel(' Value')
plt.legend()
```

Out[351]:

```
<matplotlib.legend.Legend at 0x212f3c8ca90>
```



Влияние времени на цену по Блэку-Шоулзу

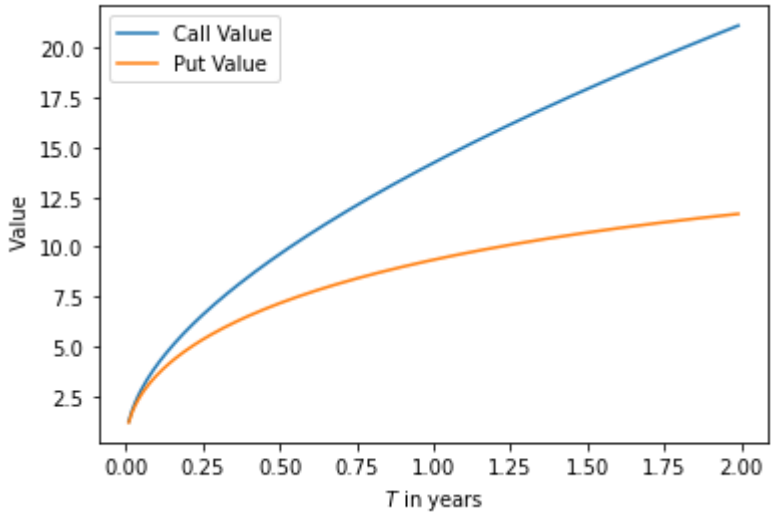
С увеличением времени увеличивается неопределенность относительно будущей цены. Поскольку неопределенность идет на пользу держателю опциона, цена опциона растет со временем. Снова попробуем установить процентную ставку на ноль, чтобы увидеть, что разница между опционами и контрактами исчезает.

Ввод [355]:

```
K = 100
r = 0.05
T = np.arange(0, 2, 0.01)
sigma = 0.3
S = 100

calls = [BS_CALL(S, K, t, r, sigma) for t in T]
puts = [BS_PUT(S, K, t, r, sigma) for t in T]
plt.plot(T, calls, label='Call Value')
plt.plot(T, puts, label='Put Value')
plt.xlabel('$T$ in years')
plt.ylabel(' Value')
plt.legend()
```

Out[355]:

```
<matplotlib.legend.Legend at 0x212f3d49be0>
```



Основная проблема подхода Блэка Шоулза заключается в том, что она не может адекватно работать при условии изменчивой волатильности.