# Mathematical Economics

# Alpha Chiang

# Chapter 10 - 11

Exponential and Logarithmic Functions

10.1 The Nature of Exponential Functions

In [2]:

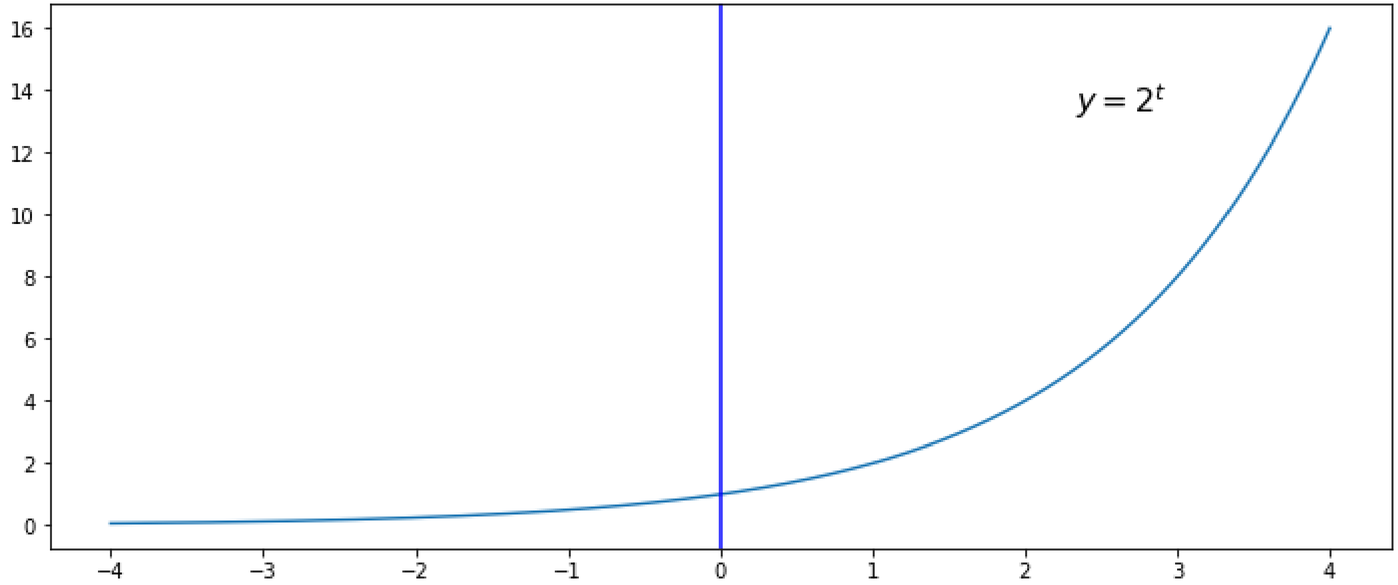
```
from sympy import Symbol, dsolve, Function, Derivative, Eq
f = Function("f")
t = Symbol('t')
b = Symbol('b')
eq1 = Eq(f(t), b**t)
eq1
```

Out[2]: $f(t) = b^t$

In [3]:

```
import matplotlib .pyplot as plt
import numpy as np

fig = plt.figure(figsize =(12,5))
ax = fig.add_subplot()
b = np.linspace(-4, 4, 1000)
func, = ax.plot(b, 2**b)
ax.annotate ('$y = 2^t$', xy =(0.8 , 0.8) , fontsize =16, xycoords='axes fraction',
ha='center')
ax.axvline(x = 0, color = 'b', label = 'axvline - full height')
plt.show()
```




# Chapter 11

The Case of More than One Choice Variable

Example 1

In [4]:
```
from sympy import Symbol, dsolve, Function, Derivative, Eq
from sympy import exp, sin, sqrt, diff,cos,pi ,latex ,simplify
f = Function("f")
y = Symbol('y')
x = Symbol('x')
def z(x,y):
    return x**3 + 5*x*y - y**2

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

$3x^2 + 5y$

$5x - 2y$

$6x$

$-2$

$5$

Example 2

In [5]:
```
y = Symbol('y')
x = Symbol('x')
def z(x,y):
    return x**2*exp(-y)

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

$2xe^{-y}$

$-x^2e^{-y}$

$2e^{-y}$

$x^2e^{-y}$

$-2xe^{-y}$

Example 4

In [6]:
```
from scipy import optimize
def z(x,y):
    return 8*x**3 + 2*x*y -3*x**2 + y**2 + 1
def f(x,y):
    d1 = diff(z(x, y), x)
    d1_ = diff(z(x,y),x,2)
    d2 = diff(z(x, y), y)
    d2_ = diff(z(x,y),y,2)
    return d1,d1_,d2,d2_
f(x,y)
```

Out[6]: (24*x**2 - 6*x + 2*y, 6*(8*x - 1), 2*x + 2*y, 2)

In [7]:
```
from sympy import *

x, y = symbols('x, y')
eq1 = Eq(24*x**2 - 6*x + 2*y, 0)
eq2 = Eq(2*x + 2*y, 0)

sol = solve([eq1, eq2], [x, y])
sol
```

Out[7]: [(0, 0), (1/3, -1/3)]

Example 5

In [8]:
```
def z(x,y):
    return x + 2*exp(1)*y - exp(x) - exp(2*y)
def f(x,y):
    d1 = diff(z(x, y), x)
    d1_ = diff(z(x,y),x,2)
    d2 = diff(z(x, y), y)
    d2_ = diff(z(x,y),y,2)
    return d1,d1_,d2,d2_
f(x,y)
```

Out[8]: (1 - exp(x), -exp(x), -2*exp(2*y) + 2*E, -4*exp(2*y))

In [9]:
```
x, y = symbols('x, y')
eq1 = Eq(1 - exp(x) + 2*y, 0)
eq2 = Eq(-2*exp(2*y) + 2*exp(1), 0)

sol = solve([eq1, eq2], [x, y])
sol
```

Out[9]: [(log(2), 1/2)]

In [10]:
```
from sympy import *
r = Symbol('r')
def f(r):
    S = Matrix([[ 2 - r, 2],
                [ 2, -1 - r]])
```

```python
    return S.det()
f(r)
```

Out[10]: $r^2 - r - 6$

In [11]:
```python
roots = solve(r**2 - r - 6,r)
roots
```

Out[11]: [-2, 3]

In [12]:
```python
x1 = Symbol('x_1')
x2 = Symbol('x_2')
M1 = Matrix([[-1, 2],
            [2, -4]])
M2 = Matrix((x1,x2))
M1*M2
```

Out[12]:
$$\begin{bmatrix} -x_1 + 2x_2 \\ 2x_1 - 4x_2 \end{bmatrix}$$

11.4 Objective Functions with More than Two Variables

Example 1

In [15]:
```python
x1 = Symbol('x_1')
x2 = Symbol('x_2')
x3 = Symbol("x_3")
def z(x1, x2, x3):
    return (2*x1**2 + x1*x2 + 4*x2**2 +
            x1*x3 + x3**2 + 2)

F11 = diff(z(x1,x2,x3), x1,2)
F12 = diff(z(x1,x2,x3), x1,x2)
F13 = diff(z(x1,x2,x3), x1,x3)

F21 = diff(z(x1,x2,x3), x2,x1)
F22 = diff(z(x1,x2,x3), x2,2)
F23 = diff(z(x1,x2,x3), x2,x3)

F31 = diff(z(x1,x2,x3), x3,x1)
F32 = diff(z(x1,x2,x3), x3,x2)
F33 = diff(z(x1,x2,x3), x3,2)
```

```python
M1 = Matrix([[F11, F12, F13],
             [F21, F22, F23],
             [F31, F32, F33]])
M1
```

Out[15]:

$$\left[\begin{matrix}4 & 1 & 1\\1 & 8 & 0\\1 & 0 & 2\end{matrix}\right]$$

In [16]:

```python
M1.det()
```

Out[16]: $54$

In [14]:

```python
# Another and easy way

from sympy import symbols, Matrix, Function, simplify, exp, hessian, solve, init_printing
init_printing()

x1, x2,x3 = symbols('x1 x2 x3')
f, g, h = symbols('f g h', cls=Function)

X = Matrix([x1,x2,x3])
f = Matrix([2*x1**2 + x1*x2 + 4*x2**2 +
x1*x3 + x3**2 + 2])
hessianf = simplify(hessian(f, X))
hessianf
```

Out[14]:

$$\left[\begin{matrix}4 & 1 & 1\\1 & 8 & 0\\1 & 0 & 2\end{matrix}\right]$$

11.6 Economic Applications

In [17]:

```python
P1 = Symbol('P_1')
P2 = Symbol('P_2')
Q1 = Symbol("Q_1")
Q2 = Symbol("Q_2")
```

```
def z(P1,P2,Q1,Q2):
    return (P1*Q1 + P2*Q2 - 2*Q1**2 -Q1*Q2
           - 2*Q2**2)
d1 = diff(z(P1,P2,Q1,Q2),Q1)
d2 = diff(z(P1,P2,Q1,Q2),Q2)
display(d1,d2)
```

$$P_1 - 4Q_1 - Q_2$$

$$P_2 - Q_1 - 4Q_2$$

In [18]:
```
from sympy import symbols, Eq, solve

eq1 = Eq(P1, 4*Q1 + Q2)
eq2 = Eq(P2, Q1 + 4*Q2)
eq3 = Eq(12, 4*Q1 + Q2)
eq4 = Eq(18, Q1 + 4*Q2)

result2 = solve([eq3, eq4],(Q1, Q2))
result = solve([eq1, eq2],(Q1, Q2))
display(result,result2)
```

$$\left\{Q_1 : \frac{4P_1}{15} - \frac{P_2}{15},\ Q_2 : -\frac{P_1}{15} + \frac{4P_2}{15}\right\}$$

$$\{Q_1 : 2,\ Q_2 : 4\}$$

In [19]:
```
F11 = diff(z(P1,P2,Q1,Q2),Q1,2)
F12 = diff(z(P1,P2,Q1,Q2),Q1,Q2)


F21 = diff(z(P1,P2,Q1,Q2),Q1,Q2)
F22 = diff(z(P1,P2,Q1,Q2),Q1,2)


M1 = Matrix([[F11, F12],
             [F21, F22]])
display(M1,M1.det())
```

$$\begin{bmatrix} -4 & -1 \\ -1 & -4 \end{bmatrix}$$

$15$

In [20]:
```
P1 = Symbol('P_1')
P2 = Symbol('P_2')
P3 = Symbol('P_3')
Q = Symbol("Q")
Q1 = Symbol("Q_1")
Q2 = Symbol("Q_2")
Q3 = Symbol('Q_2')

def z(Q1):
    R1 = (63*Q1 - 4*Q1**2)
    return R1

def z2(Q3):
    R3 = (75*Q3 - 6*Q3**2)
    return R3

def z3(Q2):
    R2 = (105*Q2 - 5*Q2**2)
    return R2

def z4(Q):
    C = 20 + 15*Q
    return C
d1 = diff(z(Q1),Q1)
d2 = diff(z2(Q3),Q3)
d3 = diff(z3(Q2),Q2)
d4 = diff(z4(Q),Q)
display(d1,d2,d3,d4)
```

$63 - 8Q_1$

$75 - 12Q_2$

$105 - 10Q_2$

$15$

In [21]:
```
a = np.array([[-8, 0, 0], [0, -10, 0], [0, 0, -12]])
b = np.array([-48, -90, -60])
x = np.linalg.solve(a, b)
x
```

Out[21]:
```
array([6., 9., 5.])
```

Example 5

In [22]:
```
P = Symbol('P')
L = Symbol('L')
K = Symbol("K")
alpha = Symbol("\\alpha")
w = Symbol("w")
r = Symbol("r")


def p(P,L,K,w,r,alpha):
    return (P*L**(alpha) * K**(alpha) -
           w*L - r*K)

F11 = diff(p(P,L,K,w,r,alpha) ,L,2)
F12 = diff(p(P,L,K,w,r,alpha) ,L,K)


F21 = diff(p(P,L,K,w,r,alpha), K,L)
F22 = diff(p(P,L,K,w,r,alpha),K,2)


M1 = Matrix([[F11, F12],
             [F21, F22]])
display(M1,M1.det())
```

$$\begin{bmatrix} \frac{K^{\alpha}L^{\alpha}P\alpha(\alpha-1)}{L^2} & \frac{K^{\alpha}L^{\alpha}P\alpha^2}{KL} \\ \frac{K^{\alpha}L^{\alpha}P\alpha^2}{KL} & \frac{K^{\alpha}L^{\alpha}P\alpha(\alpha-1)}{K^2} \end{bmatrix}$$

$$-\frac{2K^{2\alpha}L^{2\alpha}P^2\alpha^3 - K^{2\alpha}L^{2\alpha}P^2\alpha^2}{K^2L^2}$$

In [23]:
```
P = Symbol('P')
L = Symbol('L')
K = Symbol("K")
alpha = Symbol("\\alpha")
w = Symbol("w")
r = Symbol("r")
```

```
def p(P,L,K,w,r,alpha):
    return (P*L**(alpha) * K**(alpha) -
           w*L - r*K)
```

In [24]:
```
d1 = diff(p(P,L,K,w,r,alpha) ,L)
d2 = diff(p(P,L,K,w,r,alpha),K)
display(d1,d2)
```

$$\frac{K^{\alpha}L^{\alpha}P\alpha}{L} - w$$

$$-r + \frac{K^{\alpha}L^{\alpha}P\alpha}{K}$$

In [25]:
```
from sympy import symbols, Eq, solve

eq1 = Eq(w, (K**(alpha) * L**(alpha)*P*alpha)/L)
eq2 = Eq(r, (K**(alpha) * L**(alpha)*P*alpha)/K)

result = solve([eq1, eq2],(K, L))
display(result)
```

$$\left[ \left( \left( \frac{w\left(\left(\frac{1}{P\alpha}\right)^{\frac{1}{2\alpha-1}}\left(rw^{\frac{1-\alpha}{\alpha}}\right)^{\frac{\alpha}{2\alpha-1}}\right)^{1-\alpha}}{P\alpha} \right)^{\frac{1}{\alpha}}, \left(\frac{1}{P\alpha}\right)^{\frac{1}{2\alpha-1}}\left(rw^{\frac{1-\alpha}{\alpha}}\right)^{\frac{\alpha}{2\alpha-1}} \right) \right]$$

Furkan zengin