Mathematical Economics

Alpha Chiang

Chapter 8

Comparative-Static Analysis of General Function Models

Differentials and Point Elasticity

In [3]:
```
from sympy import Symbol, dsolve, Function, Derivative,symbols
from sympy import diff, Eq
Q = Function("Q")
P = Function("P")
a, d, epsilond = symbols("a d \\epsilon_d")
#Point elasticity of demand
eq1 = Eq(epsilond , (Derivative(Q(a),a,1)/Q(a)) / ((Derivative(P(a),a))/Q(a)))
eq1
```

Out[3]:

$$\epsilon_d = \frac{\frac{d}{da} Q(a)}{\frac{d}{da} P(a)}$$

Example 1

In [6]:
```
from scipy.misc import derivative
from sympy import simplify
x = Symbol("x")

def demand(x):
    "x = P"
    return 100-2*x

def deriv(x):
    return derivative(demand,x)

def avg(x):
```

```
    return demand(x)/x

def elasticity(x):
    return deriv(x)/avg(x)

E = elasticity(x)
simplify(E)
```

Out[6]:

$$\frac{1.0x}{x - 50}$$

Example 2

In [7]:

```
def demand(x):
    return x**2 + 7*x

def deriv(x):
    return derivative(demand,x)

def avg(x):
    return demand(x)/x

def elasticity(x):
    return deriv(x)/avg(x)

E = elasticity(x)
simplify(E)
```

Out[7]:

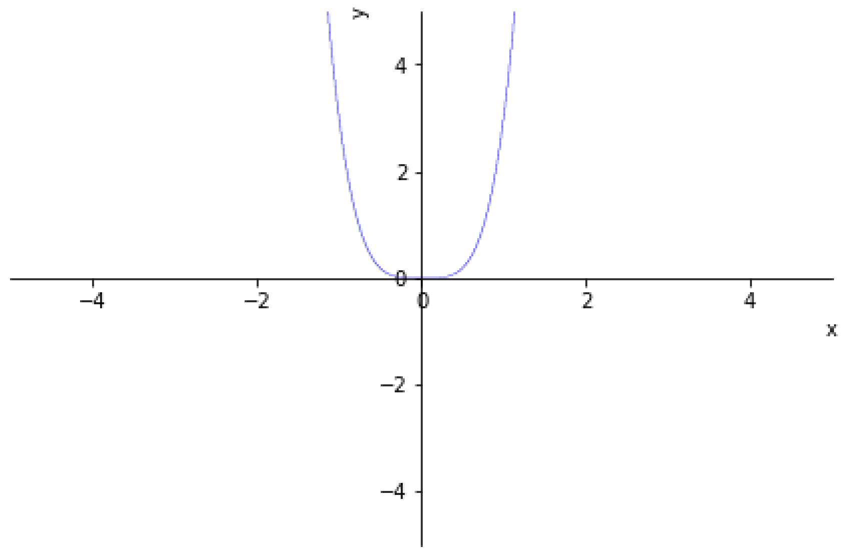
$$\frac{1.0\,(2.0x + 7.0)}{1.0x + 7.0}$$

8.5 Derivatives of Implicit Functions

Example 1

In [9]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import sympy as sy
x, y = symbols('x y')
eq1 = Eq(y - 3*x**4, 0)
sy.plot_implicit(eq1)
```



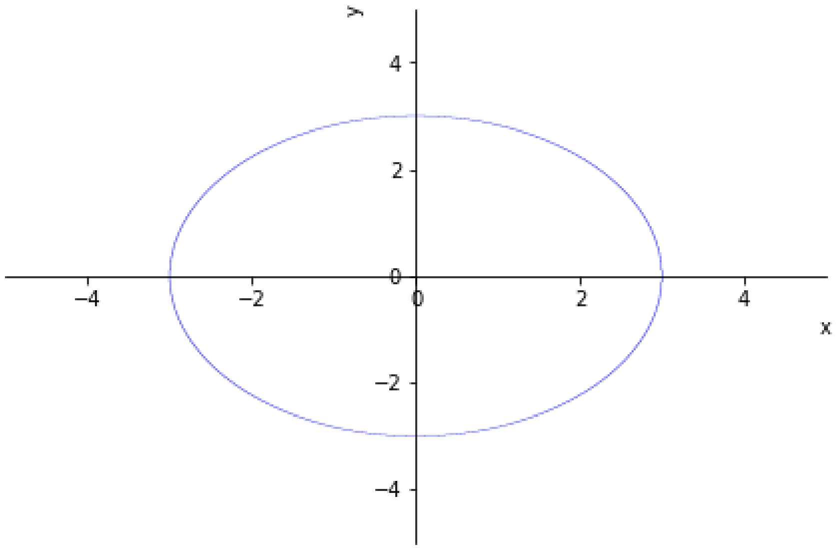
Out[9]: <sympy.plotting.plot.Plot at 0x2d4c3566df0>

In [10]:
```
eq1 = y - 3*x**4
deq1 = sy.idiff(eq1, y, x)
deq1
```

Out[10]: $12x^3$

Example 2

In [12]:
```
eq2 = Eq(x**2 + y**2 - 9, 0)
sy.plot_implicit(eq2)
```



Out[12]: <sympy.plotting.plot.Plot at 0x2d4c4676c10>

In [13]:
```
eq2 = x**2 + y**2 - 9
deq2 = sy.idiff(eq2, y, x)
deq2
```

Out[13]: $-\frac{x}{y}$

Example 3

In [14]:
```
x, y, w, z= symbols('x y w,z')
eq3 = (y**3 * x**2) + w**3 +y*x*w - 3
deq3 = sy.idiff(eq3, y, x)
display(deq3)
deq3.subs({x:1, y:1, w:1}) # At the point (1, 1, 1)
```

$$-\frac{y\left(w+2xy^2\right)}{x\left(w+3xy^2\right)}$$

Out[14]: $-\frac{3}{4}$

In [15]:
```
dx, dy, dw, dz = symbols('dx dy dw dz')
```

```python
def f(x, y, w):
    eq1 = x*y - w
    F1 = diff(eq1,x)
    F1_1 = diff(eq1,y)
    F1_2 = diff(eq1,w)
    return F1*dx + F1_1*dy + F1_2*dw


def f2(z, y, w):
    eq2 = y - w**3 - 3*z
    F2 = diff(eq2,z)
    F2_1 = diff(eq2,y)
    F2_2 = diff(eq2,w)
    return F2*dz + F2_1*dy + F2_2*dw

def f3(w,z):
    eq3 = w**3 + z**3 -2*w*z
    F3 = diff(eq3,z)
    F3_1 = diff(eq3,w)
    return F3*dz + F3_1*dw
display(f(x,y,w), f2(z,y,w), f3(w,z))
TotalD = [f(x,y,w), f2(z,y,w), f3(w,z)]
TotalD
```

$$-dw + dxy + dyx$$

$$-3dww^2 + dy - 3dz$$

$$dw\left(3w^2 - 2z\right) + dz\left(-2w + 3z^2\right)$$

Out[15]:
```
[-dw + dx*y + dy*x,
 -3*dw*w**2 + dy - 3*dz,
 dw*(3*w**2 - 2*z) + dz*(-2*w + 3*z**2)]
```

In [16]:
```python
import sympy as sp
M = sp.Matrix(([y,x,-1],[0,1,-3*w**2],[0,0,3*w**2 -2*z]))
display(M)
Det1 = sp.det(M)
display(Det1)
Det1.subs({y:4,w:1,z:1}) #the Jacobian determinant
```

$$\begin{bmatrix} y & x & -1 \\ 0 & 1 & -3w^2 \\ 0 & 0 & 3w^2 - 2z \end{bmatrix}$$

$$3w^2y - 2yz$$

Out[16]: 4

Example 6

In [17]:

```
from sympy import diff, Eq
Y,C,G0,I0,T,alpha = symbols('Y C G_0 I_0 T \\alpha')
beta , delta ,gamma = symbols('\\beta \\delta \\gamma')
eq1 =  Y - C - I0 - G0
eq2 = C - alpha - beta*(Y - T)
eq3 = T - gamma - delta*Y
M2 = sp.Matrix(([diff(eq1,Y),diff(eq1,C),diff(eq1,T)],
    [diff(eq2,Y),diff(eq2,C),diff(eq2,T)],
    [diff(eq3,Y),diff(eq3,C),diff(eq3,T)]))
display(M2)
Det2 = sp.det(M2)
display(Det2)
```

$$\begin{bmatrix} 1 & -1 & 0 \\ -\beta & 1 & \beta \\ -\delta & 0 & 1 \end{bmatrix}$$

$$\beta\delta - \beta + 1$$

Furkan zengin