Mathematical Economics

Alpha Chiang

Chapter 4-5

Chapter 4 Linear Models and Matrix Algebra Matrices as Arrays

In [1]:
```
from sympy import symbols,Matrix
x1, x2, x3 = symbols('x_1,x_2,x_3')
A = Matrix(([6, 3, 1], [1, 4, -2], [4, -1 , 5]))
A
```

Out[1]:
$$\begin{bmatrix} 6 & 3 & 1 \\ 1 & 4 & -2 \\ 4 & -1 & 5 \end{bmatrix}$$

In [2]:
```
x = Matrix((x1,x2,x3))
x
```

Out[2]:
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

In [3]:
```
d = Matrix((22,12,10))
d
```

Out[3]:
$$\begin{bmatrix} 22 \\ 12 \\ 10 \end{bmatrix}$$

In [4]:
```
A * x
```

Out[4]:

$$\begin{bmatrix} 6x_1 + 3x_2 + x_3 \\ x_1 + 4x_2 - 2x_3 \\ 4x_1 - x_2 + 5x_3 \end{bmatrix}$$

In [5]:

```
import numpy as np
npA = np.array(([6, 3, 1], [1, 4, -2], [4, -1 , 5]))
# To be able to solve this system linearly, we need to use numpy arrays
npA
npd = np.array((22,12,10))
npd
x = np.linalg.solve(npA, npd)
x
```

Out[5]:

```
array([2., 3., 1.])
```

Basic Matrix Operations and other operations can be found the below websites

https://numpy.org/doc/stable/reference/generated/numpy.matrix.html

https://docs.sympy.org/latest/tutorial/matrices.html

Example 6 --PAGE 78--

In [7]:

```
from sympy import symbols, Eq, solve
Y, C, I0, G0, a, b = symbols('Y, C, I_0, G_0, a, b')
eq1 = Eq(Y, C + I0 + G0)
eq2 = Eq(C, a + b*Y)

result = solve([eq1, eq2],(Y,C))

print(result[Y])
print(result[C])
```

```
-(G_0 + I_0 + a)/(b - 1)
-(a + b*(G_0 + I_0))/(b - 1)
```

In [8]:

```
from sympy import symbols,Matrix
from sympy import Symbol, dsolve, Function, Derivative, Eq
Y, C, I0, G0, a, b = symbols('Y, C, I_0, G_0, a, b')
eq1 = Eq(Y, C + I0 + G0)
eq1
```

Out[8]:

$Y = C + G_0 + I_0$

In [9]:
```
eq2 = Eq(C, a + b*Y)
eq2
```

Out[9]: $C = Yb + a$

In [10]:
```
from sympy import *
Y, C, I0, G0, a, b = symbols('Y, C, I_0, G_0, a, b')
eqns = [ (Y - C - I0 - G0), (C - a - b*Y)]
linsolve(eqns, [Y ,C ])
```

Out[10]: $$\left\{\left(-\frac{G_0 + I_0 + a}{b - 1},\ -\frac{G_0 b + I_0 b + a}{b - 1}\right)\right\}$$

CHAPTER 5

Linear Models and Matrix Algebra (Continued)

Example 9 --PAGE 97--

In [12]:
```
from sympy import symbols,Matrix
x1, x2, x3 = symbols('x_1,x_2,x_3')
A = Matrix(([7, -3, -3], [2, 4, 1], [0, -2 , -1]))
A
```

Out[12]: $$\begin{bmatrix} 7 & -3 & -3 \\ 2 & 4 & 1 \\ 0 & -2 & -1 \end{bmatrix}$$

In [13]:
```
x = Matrix((x1,x2,x3))
x
```

Out[13]: $$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

In [14]:
```
A * x
```

Out[14]:

$$\begin{bmatrix} 7x_1 - 3x_2 - 3x_3 \\ 2x_1 + 4x_2 + x_3 \\ -2x_2 - x_3 \end{bmatrix}$$

In [15]:
```
import numpy as np
npA = np.array(([7, -3, -3], [2, 4, 1], [0, -2 , -1]))
D1 = np.linalg.det(npA)
D1
```

Out[15]:
```
-7.999999999999998
```

In [16]:
```
npd = np.array((7,0,2))
npd
x = np.linalg.solve(npA, npd)
x
```

Out[16]:
```
array([-0.5,  1.5, -5. ])
```

In [17]:
```
from numpy.linalg import matrix_rank
matrix_rank(npA) #Rank of a Matrix
```

Out[17]:
```
3
```

5.4 Finding the Inverse Matrix

Example 1

In [18]:
```
import numpy as np
npA = np.array(([4, 1, 2], [5, 2, 1], [1, 0 , 3]))
npA
```

Out[18]:
```
array([[4, 1, 2],
       [5, 2, 1],
       [1, 0, 3]])
```

In [19]:
```
from numpy.linalg import inv
inv(npA)
```

Out[19]:
```
array([[ 1.        , -0.5       , -0.5       ],
```

```
       [-2.33333333,  1.66666667,  1.        ],
       [-0.33333333,  0.16666667,  0.5       ]])
```

Example 2

In [20]:
```
npA = np.array(([3, 2], [1, 0]))
inv(npA)
```

Out[20]:
```
array([[ 0. ,  1. ],
       [ 0.5, -1.5]])
```

Example 3

In [21]:
```
npA = np.array(([4, 1, -1], [0, 3, 2], [3, 0 , 7]))
D2 = np.linalg.det(npA)
D2
```

Out[21]:
```
98.99999999999999
```

In [22]:
```
inv(npA)
```

Out[22]:
```
array([[ 0.21212121, -0.07070707,  0.05050505],
       [ 0.06060606,  0.31313131, -0.08080808],
       [-0.09090909,  0.03030303,  0.12121212]])
```

5.6 Application to Market and National-Income Models

Market Model

In [24]:
```
P1 = Symbol('P_1')
P2 = Symbol('P_2')
c0 = Symbol('c_0')
c1 = Symbol('c_1')
c2 = Symbol('c_2')
gamma0 = Symbol('\\gamma_0')
gamma1 = Symbol('\\gamma_1')
gamma2 = Symbol('\\gamma_2')
eq1 = Eq(c1*P1 + c2*P2,-c0)
eq2 = Eq(gamma1*P1 + gamma2*P2,-gamma0)
display(eq1,eq2)
```

$$P_1c_1 + P_2c_2 = -c_0$$

$$P_1\gamma_1 + P_2\gamma_2 = -\gamma_0$$

In [25]:
```python
from sympy import symbols, Eq, solve
solve((eq1,eq2), (P1,P2))
```

Out[25]:
```
{P_1: (-\gamma_0*c_2 + \gamma_2*c_0)/(\gamma_1*c_2 - \gamma_2*c_1),
 P_2: (\gamma_0*c_1 - \gamma_1*c_0)/(\gamma_1*c_2 - \gamma_2*c_1)}
```

IS-LM Model: Closed Economy

In [26]:
```python
Y = Symbol('Y')
C = Symbol('C')
I = Symbol('I')
G = Symbol('G')
a = Symbol('a')
b = Symbol('b')
t = Symbol('t')
d = Symbol('d')
e = Symbol('e')
i = Symbol('i')
G0 = Symbol('G_0')
M0 = Symbol('M_0')
Md = Symbol("M_d")
Ms = Symbol("M_s")
l = Symbol('l')
k = Symbol('k')
eq1 = Eq(Y, C + I + G)
eq2 = Eq(Md,Ms)
eq3 = Eq(C, a + b*(1 - t)*Y)
eq4 = Eq(I, d -e*i)
eq5 = Eq(G, G0)
eq6 = Eq(M0, k*Y -l*i)
```

In [28]:
```python
display(eq1,eq2,eq3,eq4,eq5,eq6)
```

$$Y = C + G + I$$

$$M_d = M_s$$

$$C = Yb(1 - t) + a$$

$$I = d - ei$$

$$G = G_0$$

$M_0 = Yk - il$

In [29]:
```
from sympy import symbols,Matrix
A = Matrix(([1, -1, -1, 0], [b*(1-t), -1, 0, 0],
            [0, 0 , 1, e], [k,0,0,-l]))
A
```

Out[29]:

$$\left[\begin{matrix}1 & -1 & -1 & 0\\b \left(1 - t\right) & -1 & 0 & 0\\0 & 0 & 1 & e\\k & 0 & 0 & -l\end{matrix}\right]$$

In [30]:
```
x = Matrix((Y,C,I,i))
x
```

Out[30]:

$$\left[\begin{matrix}Y\\C\\I\\i\end{matrix}\right]$$

In [31]:
```
d = Matrix((G0,-a,d,M0))
d
```

Out[31]:

$$\left[\begin{matrix}G_{0}\\- a\\d\\M_{0}\end{matrix}\right]$$

In [32]:
```
A * x
```

Out[32]:

$$\left[\begin{matrix}- C - I + Y\\- C + Y b \left(1 - t\right)\\I + e i\\Y k - i l\end{matrix}\right]$$

In [33]:
```
from sympy import *
Y, C, I, G, G0, a, b = symbols('Y, C, I, G, G_0, a, b')
```

```
t, d, e, i, M0 ,Md ,Ms ,l ,k = symbols("t, d, e, i,M_0,M_d,M_s,l,k")
eqns = [(Y - C - I - G0), (-C + a + (b*(1 - t)*Y)), (I - d + (e*i)) ,(-M0 + (k*Y) + (-l*i))]
eqns
```

Out[33]: [-C - G_0 - I + Y, -C + Y*b*(1 - t) + a, I - d + e*i, -M_0 + Y*k - i*l]

In [46]:
```
SOL = linsolve(eqns, [Y,C,i,I])
# Run the SOL
```

EXERCISE 5.6 --Q3--

In [38]:
```
from sympy import symbols,Matrix
A = Matrix(([0.3, 100], [0.25, -200]))
A
```

Out[38]:

$$\begin{bmatrix} 0.3 & 100 \\ 0.25 & -200 \end{bmatrix}$$

In [39]:
```
x = Matrix((Y,i))
x
```

Out[39]:

$$\begin{bmatrix} Y \\ i \end{bmatrix}$$

In [40]:
```
d = Matrix((252,176))
d
```

Out[40]:

$$\begin{bmatrix} 252 \\ 176 \end{bmatrix}$$

In [41]:
```
A * x
```

Out[41]:

$$\begin{bmatrix} 0.3Y + 100i \\ 0.25Y - 200i \end{bmatrix}$$

In [42]:
```
import numpy as np
npA = np.array(([0.3, 100], [0.25, -200]))
npA
npd = np.array((252,176))
npd
```

```python
x = np.linalg.solve(npA, npd)
x
```

Out[42]:

```
array([8.0e+02, 1.2e-01])
```

Furkan zengin