Funciones Reales y Modelización de Funciones Económicas

Tecnicatura Universitaria en Análisis y Gestión de Datos

Manuel Fernández San Martín

Laboratorio de Métodos Cuantitativos Aplicados a la Gestión
Universidad de Buenos Aires

Primer cuatrimestre 2026

Aplicar Python para modelizar funciones económicas y determinar puntos de equilibrio de mercado.

Contenidos de la clase

¿Qué vamos a ver hoy?

- Repaso de funciones reales y sus tipos principales
- Modelización de funciones económicas: demanda, oferta, costo, ingreso y beneficio
- Construcción e interpretación de funciones a partir de datos del problema
- Concepto de equilibrio de mercado y resolución de sistemas
- Efectos de impuestos, subsidios y cambios en preferencias
- Implementación en Python con `NumPy`, `SymPy` y `Matplotlib`

Sección 1

Repaso de funciones reales

¿Qué es una función?

Definición

Una función f es una relación que asigna a cada elemento x de un conjunto A (dominio) **exactamente un** elemento $f(x)$ de un conjunto B (codominio).

Conceptos clave:

- **Dominio:** valores de x para los que f está definida
- **Imagen:** valores que puede tomar $f(x)$

En economía trabajamos con funciones reales:

$$f : \mathbb{R} \to \mathbb{R}$$

¿Por qué importa en gestión?

Toda relación económica cuantificable puede modelizarse como una función: precio–demanda, producción–costo, tiempo–beneficio, etc.

Tipos de funciones reales — resumen

Tipo	**Forma general**	**Aplicación económica típica**
Lineal	$f(x) = mx + b$	Oferta/demanda lineal, costo lineal
Cuadrática	$f(x) = ax^2 + bx + c$	Costo variable, beneficio
Exponencial	$f(x) = a \cdot e^{bx}$	Crecimiento compuesto, inflación
Logarítmica	$f(x) = \ln(x)$	Utilidad marginal decreciente
Homográfica	$f(x) = \dfrac{ax + b}{cx + d}$	Demanda hiperbólica
Polinómica	$f(x) = a_n x^n + \cdots + a_0$	Modelos de costos complejos

En Python: `import numpy as np` y `import matplotlib.pyplot as plt`

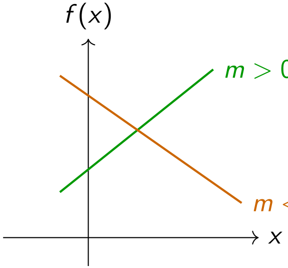
Función lineal

Forma

$$f(x) = mx + b$$

- m: pendiente (tasa de cambio)
- b: ordenada al origen

```
import numpy as np, matplotlib.pyplot as plt
m, b = 2, 3
x = np.linspace(-5, 5, 100)
y = m * x + b
plt.plot(x, y, label=f'y = {m}x + {b}')
plt.axhline(0, color='black', lw=0.5)
plt.axvline(0, color='black', lw=0.5)
plt.grid(True); plt.legend(); plt.show()
```



Clave

$m > 0$: relación positiva (típico en oferta)
$m < 0$: relación negativa (típico en demanda)

Funciones cuadrática y logarítmica

Cuadrática: $f(x) = ax^2 + bx + c$

- $a > 0$: parábola con mínimo (costos convexos)
- $a < 0$: parábola con máximo (beneficio)
- Raíces: $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

```python
a, b, c = 1, -10, 600
y = a*x**2 + b*x + c
plt.plot(x, y, color='purple')
```

Logarítmica: $f(x) = \ln(x)$

- Dominio: $x > 0$
- Crece pero a ritmo decreciente
- Modela utilidad marginal o costos con rendimientos decrecientes

```python
x = np.linspace(0.01, 10, 100)
y = np.log(x)    # logaritmo natural
plt.plot(x, y, 'green')
```

Nota

`np.log(x)` $= \ln(x)$ `np.log10(x)` $= \log_{10}(x)$

Sección 2

Modelización de funciones de oferta y demanda

Funciones de oferta y demanda

Función de demanda $D(p)$

Relaciona la **cantidad demandada** x con el precio p.

$$x = D(p) \qquad \Longleftrightarrow \qquad p = D^{-1}(x)$$

A mayor precio, menor cantidad demandada: **pendiente negativa**.

Función de oferta $O(p)$

Relaciona la **cantidad ofrecida** x con el precio p.

$$x = O(p) \qquad \Longleftrightarrow \qquad p = O^{-1}(x)$$

A mayor precio, mayor cantidad ofrecida: **pendiente positiva**.

Importante

Cuando las funciones están en términos de p, se invierte para expresar p en función de x (precio como función de cantidad).

Ejemplo — Ecuación de oferta (CDs)

Problema: La curva de oferta es lineal. Cuando $p = \$30$ hay 35 unidades disponibles; cuando $p = \$35$ hay 50.

Resolución paso a paso

$$m = \frac{p_2 - p_1}{x_2 - x_1} = \frac{35 - 30}{50 - 35} = \frac{1}{3}$$

$$b = p_1 - m \cdot x_1 = 30 - \tfrac{1}{3} \cdot 35 = \tfrac{55}{3}$$

$$\Rightarrow \quad p = \frac{1}{3}x + \frac{55}{3}$$

```python
x1, p1 = 35, 30
x2, p2 = 50, 35

# Pendiente
m = (p2 - p1) / (x2 - x1)

# Ordenada al origen
b = p1 - m * x1

print(f"p = {m:.2f}x + {b:.2f}")
# p = 0.33x + 18.33
```

Interpretación

Por cada unidad adicional ofrecida, el precio sube $\approx \$0{,}33$. El precio mínimo de oferta es $\approx \$18{,}33$.

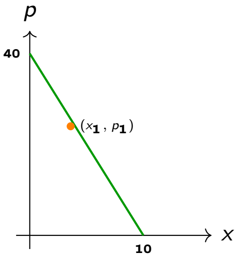
Ejemplo — Curva de demanda: $4x + p - 40 = 0$

Reescribimos: $p = -4x + 40 \quad \Leftrightarrow \quad x = \dfrac{40 - p}{4}$

```python
def cantidad_demandada(p):
    return (40 - p) / 4

def precio_demanda(x):
    return -4*x + 40

# a) Cantidad para p=4 y p=24
print(cantidad_demandada(4))    # 9.0
print(cantidad_demandada(24))   # 4.0

# b) Precio para x=1 y x=5
print(precio_demanda(1))   # 36
print(precio_demanda(5))   # 20

# c) Mayor precio (x=0)
print(precio_demanda(0))   # 40

# d) Cantidad si p=0
print(cantidad_demandada(0))  # 10.0
```

Interpretaciones

- **c)** El mayor precio que se pagaría es \$40 (cuando $x = 0$, intersección eje p)
- **d)** Si el bien es gratis ($p = 0$), se demandan 10 unidades (intersección eje x)



Sección 3

Funciones de costo, ingreso y beneficio

Funciones de costo

Costo Total: $C(x)$

$$C(x) = C_F + C_V(x)$$

- C_F: costo **fijo** (independiente de x)
- $C_V(x)$: costo **variable** (depende de x)

Costo Medio: $C_{med}(x)$

$$C_{med}(x) = \frac{C(x)}{x} \qquad (x > 0)$$

Costo promedio por unidad producida.

Ingreso Total: $I(x)$

$$I(x) = p(x) \cdot x$$

Donde $p(x)$ es la función inversa de demanda.

Beneficio Total: $B(x)$

$$B(x) = I(x) - C(x)$$

Criterio de decisión

$B(x) > 0$: ganancias $B(x) = 0$: punto de equilibrio $B(x) < 0$: pérdidas

Ejemplo — Costo cuadrático y demanda lineal (1/2)

Datos: $C(x) = \frac{1}{10}x^2 - 10x + 600$ y $x = -10p + 600$

```python
def costo_total(x):
    return (1/10)*x**2 - 10*x + 600

def demanda_inversa(x):
    return (600 - x) / 10

def ingreso_total(x):
    return demanda_inversa(x) * x

# a) Beneficio total
def beneficio_total(x):
    return ingreso_total(x) - costo_total(x)

# b) Beneficio medio
def beneficio_medio(x):
    return beneficio_total(x) / x

# c) Para x = 200
x_ej = 200
print(f"B(200) = {beneficio_total(x_ej)}")
print(f"Bmed(200) = {beneficio_medio(x_ej):.2f}")
# B(200) = 200.0
# Bmed(200) = 1.0
```

Funciones derivadas

Demanda inversa: $p(x) = \dfrac{600 - x}{10}$ Ingreso:

$I(x) = \dfrac{600x - x^2}{10}$ Beneficio:

$$B(x) = I(x) - C(x)$$

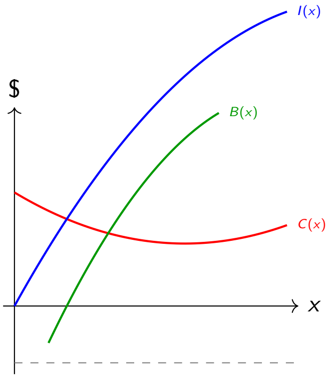
$$= -\frac{x^2}{5} + 70x - 600$$

Para 200 unidades

$B(200) = \$200$ $\quad B_{med}(200) = \$1$ por unidad

Ejemplo — Costo cuadrático y demanda lineal (2/2)

```
x = np.linspace(0, 500, 100)

plt.figure(figsize=(9,5))
plt.plot(x, costo_total(x),   'red',   lw=2, label='Costo Total')
plt.plot(x, ingreso_total(x), 'blue',  lw=2, label='Ingreso Total')
plt.plot(x, beneficio_total(x),'green',lw=2, label='Beneficio Total'
    )
plt.axhline(0, color='black', lw=0.8)
plt.xlabel('Cantidad (x)')
plt.ylabel('$')
plt.title('Ingreso, Costo y Beneficio')
plt.legend(); plt.grid(True); plt.show()
```



¿Qué observar en el gráfico?

- Los dos puntos donde $B(x) = 0$ (umbrales de rentabilidad)
- El máximo del beneficio entre esos puntos
- Zona de pérdidas cuando costo supera al ingreso

Conclusión

La empresa es rentable entre los dos umbrales de ganancia cero.

Ejemplo — Costo lineal y competencia perfecta

Datos: $C_F = \$800$; para 100 unidades $C(100) = \$1400$; precio de venta $p = \$10$.

```python
# Hallamos la funcion de costo lineal
x0, c0 = 0, 800       # Costo fijo
x1, c1 = 100, 1400    # Punto conocido

m = (c1 - c0) / (x1 - x0)  # m = 6
b = c0                     # b = 800
print(f"C(x) = {m}x + {b}")
# C(x) = 6x + 800

def costo_total(x): return 6*x + 800
def ingreso(x):     return 10*x
def beneficio(x):   return ingreso(x) - costo_total(x)

# Punto de equilibrio: I(x) = C(x)
# 10x = 6x + 800   =>   x* = 200
x_eq = 800 / (10 - 6)
print(f"Punto de equilibrio: x* = {x_eq}")
# Punto de equilibrio: x* = 200.0
```

Razonamiento algebraico

$$\begin{aligned} I(x) &= C(x) \\ 10x &= 6x + 800 \\ 4x &= 800 \\ x^* &= 200 \text{ unidades} \end{aligned}$$

Interpretación

Con precio de venta constante (**competencia perfecta**), la empresa cubre costos produciendo **200 unidades**. Por debajo $\rightarrow$ pérdidas. Por encima $\rightarrow$ ganancias.

Ejemplo — Costo logarítmico

Datos: Demanda: $10p + x = 300$ Costo: $C(x) = 100\ln(x+1) + 2x + 200$

```python
import numpy as np

import matplotlib.pyplot as plt

def demanda_inv(x): return (300 - x) / 10
def ingreso(x):     return demanda_inv(x) * x
def costo(x):       return 100*np.log(x+1) + 2*x + 200
def costo_med(x):   return costo(x) / x
def beneficio(x):   return ingreso(x) - costo(x)

# Para x = 100 unidades
x_prod = 100
print(f"I(100)    = ${ingreso(x_prod):.2f}")
print(f"C(100)    = ${costo(x_prod):.2f}")
print(f"Cmed(100) = ${costo_med(x_prod):.2f}")
print(f"B(100)    = ${beneficio(x_prod):.2f}")
# I(100)    = $2000.00
# C(100)    = $861.51
# Cmed(100) = $8.62
# B(100)    = $1138.49
```

Funciones derivadas

$$p(x) = \frac{300 - x}{10} \qquad I(x) = p(x)\,x = 30x - \frac{x^2}{10}$$

$$B(x) = I(x) - C(x)$$

Para $x = 100$

$B(100) \approx \$1.138$ y $C_{\text{med}}(100) \approx \$8{,}62/\text{u}.$

¿Por qué ln?

Economías de escala: el costo crece, pero el incremento marginal se reduce a medida que aumenta x.

Sección 4

Determinación de puntos de equilibrio

Equilibrio de mercado

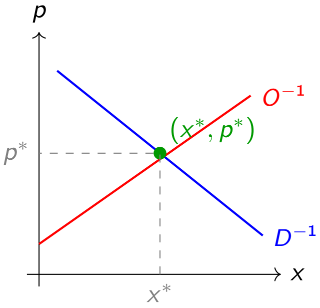
Definición

El **equilibrio de mercado** es el par (x^*, p^*) tal que la cantidad ofrecida y la cantidad demandada son iguales:

$$D(p^*) = O(p^*) \quad \Longleftrightarrow \quad D^{-1}(x^*) = O^{-1}(x^*)$$

Métodos de resolución:

- **Algebraico:** igualar las funciones y despejar x
- **Simbólico (SymPy):** resolver el sistema de ecuaciones
- **Numérico:** encontrar raíces de $D(x) - O(x) = 0$



Resolución con SymPy

```python
import sympy as sp

# Paso 1 - Variables simbolicas
x, p = sp.symbols('x p')

# Paso 2 - Definir ecuaciones
# eq1: funcion de demanda (inversa)
# eq2: funcion de oferta (inversa)
eq1 = sp.Eq(p, -(1/3)*x + 5)
eq2 = sp.Eq(p, (1/2)*x + 3/2)

# Paso 3 - Resolver el sistema
solucion = sp.solve((eq1, eq2), (x, p))
x_eq = solucion[x]
p_eq = solucion[p]

print(f"Equilibrio: x* = {x_eq}, p* = {p_eq}")
# Equilibrio: x* = 7.0, p* = 2.67
```

Ventaja de SymPy

Resuelve sistemas simbólica y exactamente. Útil cuando hay múltiples soluciones o funciones no lineales.

Sistema

$$\begin{cases} p = -\dfrac{1}{3}x + 5 \\ p = \dfrac{1}{2}x + \dfrac{3}{2} \end{cases}$$

Igualando:

$$-\tfrac{1}{3}x + 5 = \tfrac{1}{2}x + \tfrac{3}{2}$$

$$\tfrac{5}{6}x = \tfrac{7}{2} \;\Rightarrow\; x^* = \tfrac{21}{5}$$

$$p^* \approx 2{,}67$$

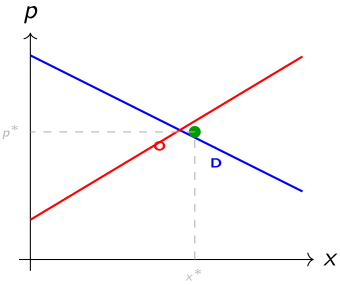
Sistema 1 — Oferta y demanda lineales (gráfico)

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

def demanda(x): return -(1/3)*x + 5
def oferta(x):  return (1/2)*x + 3/2

valores_x = np.linspace(0, 10, 100)

plt.figure(figsize=(7,5))
plt.plot(valores_x, demanda(valores_x),
         'blue', lw=2, label='Demanda: $p=-\\frac{1}{3}x+5$')
plt.plot(valores_x, oferta(valores_x),
         'red',  lw=2, label='Oferta: $p=\\frac{1}{2}x+\\frac{3}{2}$')
plt.scatter(x_eq, p_eq, color='green', s=100,
    label=f'Equilibrio ({x_eq:.1f}, {p_eq:.2f})')
plt.xlabel('Cantidad (x)')
plt.ylabel('Precio (p)')
plt.legend(); plt.grid(True); plt.show()
```

Interpretación

En $(x^* = 4{,}2;\ p^* = 3{,}6)$:

- Productores y consumidores están de acuerdo
- No hay exceso de oferta ni de demanda
- El mercado se **vacía**



Sistema 2 — Demanda lineal y oferta cuadrática

Sistema: $p = 7{,}5 - 0{,}25x \quad \text{y} \quad p = 2 + 0{,}2x + 0{,}01x^2$

```python
x, p = sp.symbols('x p')
eq1 = sp.Eq(p, 7.5 - 0.25*x)
eq2 = sp.Eq(p, 2 + 0.2*x + 0.01*x**2)

solucion = sp.solve((eq1, eq2), (x, p))
print(f"Soluciones: {solucion}")

# Filtramos soluciones economicamente validas
soluciones_validas = [
    sol for sol in solucion
    if sol[0] > 0 and sol[1] > 0
]
print(f"Equilibrio: {soluciones_validas[0]}")
# Soluciones: [(-27.4,...), (10.0, 5.0)]
# Equilibrio: (10.0, 5.0)
```

Resolución

Igualamos: $7{,}5 - 0{,}25x = 2 + 0{,}2x + 0{,}01x^2$

$$0{,}01x^2 + 0{,}45x - 5{,}5 = 0$$

Usando la fórmula cuadrática obtenemos $x^* = 10$ y $p^* = 5$.

Importante

Al resolver sistemas no lineales puede haber **varias soluciones matemáticas**. Siempre filtramos las **económicamente válidas** ($x > 0$, $p > 0$).

Equilibrio

$(x^* = 10; \ p^* = 5)$

Sistema 3 — Demanda hiperbólica y oferta lineal

Sistema: $p = \dfrac{8000}{x}$ y $p = \dfrac{x}{40} + 10$

```python
x, p = sp.symbols('x p')
eq1 = sp.Eq(p, 8000/x)
eq2 = sp.Eq(p, (1/40)*x + 10)



solucion = sp.solve((eq1, eq2), (x, p))

# Filtramos soluciones validas
soluciones_validas = [
    sol for sol in solucion
    if sol[0] > 0 and sol[1] > 0
]
x_eq, p_eq = soluciones_validas[0]
print(f"x* = {float(x_eq):.2f}")
print(f"p* = {float(p_eq):.2f}")
# x* = 200.00
## p* = 15.00
```

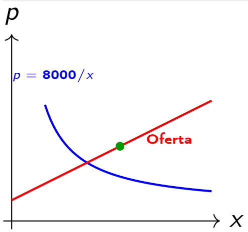
Equilibrio

$(x^* = 200;\ p^* = 15)$

Forma hiperbólica de la demanda

$$p\,x = 8000 \text{ (cte.)}$$

El gasto total es constante: **elasticidad precio** $= 1$.



Sección 5

Efectos sobre el equilibrio: impuestos, subsidios y preferencias

Impuestos y subsidios al productor

Efecto sobre la oferta

Un impuesto t **encarece** el costo del productor:

$$O_{imp}^{-1}(x) = O^{-1}(x) + t$$

Un subsidio s **reduce** el costo del productor:

$$O_{sub}^{-1}(x) = O^{-1}(x) - s$$

La curva de demanda **no se modifica**.

```
def oferta_imp(x): return 1/2*x + 3/2 + 1
def oferta_sub(x): return 1/2*x + 3/2 - 0.8

# Equilibrio con impuesto
eq3 = sp.Eq(p, -(1/3)*x + 5)
eq4 = sp.Eq(p, 1/2*x + 3/2 + 1)
sol_imp = sp.solve((eq3, eq4), (x,p))
# x* = 3.2, p* = 3.93 (sube para consumidor)
```

Conclusiones

Con **impuesto** ($t = 1$):

- Precio consumidor $\uparrow$: de 3,6 a 4
- Precio productor (neto) $\downarrow$: de 3,6 a 3
- Cantidad de equilibrio $\downarrow$: de 4,2 a 3,2

Con **subsidio** ($s = 0{,}8$):

- Precio consumidor $\downarrow$
- Cantidad de equilibrio $\uparrow$

Desplazamientos de la curva de demanda

Causas de desplazamiento

- **Preferencias** $\uparrow$ (publicidad): demanda sube $\rightarrow$ intercepto $\uparrow$
- **Bien sustituto más barato**: demanda baja $\rightarrow$ intercepto $\downarrow$

```python
# Original: p = 7.5 - 0.25x
def demanda_orig(x):  return 7.5 - 0.25*x
def oferta_2(x):      return 2 + 0.2*x + 0.01*x**2

# Con mayor preferencia (intercepto +1.5)
def dem_preferencia(x): return 9 - 0.25*x
# Con bien sustituto (intercepto -1.5)
def dem_sustituto(x):   return 6 - 0.25*x

# Equilibrio con preferencias:
# x* = 12.23, p* = 5.94
# Equilibrio con sustituto:
# x* = 7.55, p* = 4.11
```

Situación	x^*	p^*
Original	10,0	5,00
Mayor preferencia	12,2	5,94
Bien sustituto	7,5	4,11

Regla general

Demanda $\uparrow$ $\rightarrow$ precio $\uparrow$ y cantidad $\uparrow$

Demanda $\downarrow$ $\rightarrow$ precio $\downarrow$ y cantidad $\downarrow$

Cambios simultáneos en oferta y demanda

Situación: Demanda +20% (mayores ingresos) y costos de producción +15% con costo fijo +2.

```
# Original: p = 8000/x ; p = x/40 + 10
# Cambios simultaneos:
def dem_aum(x):    return (8000/x) * 1.2
def oferta_cost(x): return (1/40)*1.15*x + 12

eq7 = sp.Eq(p, 9600/x)
eq8 = sp.Eq(p, (1.15/40)*x + 12)
sol_nueva = sp.solve((eq7, eq8), (x, p))
# Solucion valida: x* ~= 205.6, p* ~= 17.72
```

	x^*	p^*
Original	200	15,00
Con cambios	205,6	17,72

Conclusiones

Cuando demanda y oferta se mueven en **sentidos opuestos**:

- Demanda $\uparrow$ + Oferta $\downarrow$ $\rightarrow$ precio siempre $\uparrow$
- Efecto sobre la cantidad depende de la **magnitud relativa**
- En este caso: cantidad $\uparrow$ levemente (+5,6 u.) pero precio $\uparrow$ fuertemente (+15%)

Ideas clave de la clase

Lo que vimos hoy

- Las funciones económicas (demanda, oferta, costo, ingreso, beneficio) son casos particulares de las funciones reales que ya conocemos: lineales, cuadráticas, logarítmicas, hiperbólicas.
- El **equilibrio de mercado** se obtiene resolviendo el sistema $D^{-1}(x) = O^{-1}(x)$ con `SymPy`.
- Impuestos, subsidios y cambios de preferencias desplazan las curvas y generan **nuevos equilibrios** que pueden analizarse comparativamente.

Para profundizar

A continuación veremos derivación e interpretación marginal, que nos permitirá encontrar el **máximo beneficio** de forma analítica (sin depender del gráfico).

Actividad práctica propuesta

Contexto

Una empresa produce insumos industriales. La demanda es $5x + 2p = 300$ y el costo es $C(x) = 0{,}05x^2 + 8x + 400$.

Tareas (en Google Colab):

1. Obtener la función de demanda inversa $p(x)$ y graficarla.
2. Construir las funciones de ingreso total $I(x)$ y beneficio $B(x)$.
3. Calcular $I(x)$, $C(x)$ y $B(x)$ para $x = 50, 100, 200$ unidades.
4. Graficar las tres curvas en un mismo gráfico. Identificar visualmente el máximo beneficio.
5. Suponer un impuesto al consumidor de \$4 por unidad. Reformular la demanda y recalcular el equilibrio.
6. Comparar los equilibrios en una tabla y comentar el impacto del impuesto.
7. **Extra:** ¿A qué precio unitario la empresa alcanza el punto de equilibrio ($B = 0$)?

Entregable: Notebook de Colab con código comentado, gráficos y análisis escrito.

Preguntas para reflexionar

Sobre funciones económicas

1. ¿Qué interpretación tiene la pendiente de una curva de costo lineal?
2. ¿Por qué la curva de ingreso de un monopolio es una parábola cóncava?
3. ¿Qué implica que $B(x) < 0$ para toda x?

Sobre equilibrio

4. ¿Puede haber más de un equilibrio económicamente válido? ¿En qué caso?
5. ¿Qué sucede con el equilibrio si tanto demanda como oferta aumentan simultáneamente?
6. ¿Por qué siempre filtramos soluciones con $x > 0$ y $p > 0$?

Bibliografía

Referencias principales

- Leithold, L. (1998). *Matemáticas para administración y economía*. Oxford University Press.
- Sydsaeter, K. & Hammond, P. (2012). *Matemáticas para el análisis económico*. Pearson.
- Wackerly, D., Mendenhall, W. & Scheaffer, R. (2010). *Estadística matemática con aplicaciones*. Cengage Learning.

Recursos digitales

- Documentación NumPy: `numpy.org/doc`
- Documentación SymPy: `docs.sympy.org`
- Repositorio de la cátedra: `github.com/espartaca75-prog/LMC-FCE--UBA`

¡Gracias!

Consultas y comentarios:
Laboratorio de Métodos Cuantitativos Aplicados a la Gestión — FCE UBA