Guía de Ejercicios
Programación Lineal

Investigación Operativa, Universidad de San Andrés

Esta guía contiene 20 ejercicios ordenados por dificultad creciente. Las respuestas y el código de resolución se encuentran en el anexo al final del documento.

1. Una empresa fabrica dos productos A y B. El producto A requiere 2 horas de mano de obra y 3 kg de materia prima, mientras que el producto B requiere 3 horas de mano de obra y 2 kg de materia prima. La empresa dispone de 100 horas de mano de obra y 80 kg de materia prima por semana. El beneficio por unidad es de $40 para A y $30 para B. Determine la cantidad óptima a producir de cada producto para maximizar el beneficio.

2. Una dietista debe preparar una mezcla nutritiva que contenga al menos 12 unidades de vitamina A y 15 unidades de vitamina B. Dispone de dos tipos de alimentos: el alimento 1 contiene 2 unidades de vitamina A y 3 de B por kg, y cuesta $4 por kg; el alimento 2 contiene 3 unidades de vitamina A y 1 de B por kg, y cuesta $3 por kg. ¿Qué cantidad debe usar de cada alimento para minimizar el costo?

3. Una fábrica produce sillas y mesas. Cada silla requiere 2 unidades de madera y 1 unidad de trabajo, mientras que cada mesa requiere 3 unidades de madera y 2 unidades de trabajo. La fábrica dispone de 60 unidades de madera y 40 unidades de trabajo. La ganancia por silla es de $20 y por mesa es de $30. ¿Cuántas sillas y mesas debe producir para maximizar la ganancia?

4. Una empresa de transporte tiene dos camiones. El camión 1 puede transportar 10 toneladas y recorre 40 km/h, mientras que el camión 2 puede transportar 15 toneladas y recorre 30 km/h. Se necesita transportar al menos 50 toneladas de mercancía y se dispone de 8 horas. El costo por hora del camión 1 es $100 y del camión 2 es $120. ¿Cuántos viajes debe hacer cada camión para minimizar el costo?

5. Una empresa produce dos tipos de juguetes: coches y trenes. Cada coche requiere 4 horas en el departamento A y 2 horas en el B, mientras que cada tren requiere 2 horas en A y 5 horas en B. Se dispone de 100 horas en A y 80 horas en B. El beneficio es de $8 por coche y $7 por tren. ¿Cuántos juguetes de cada tipo debe producir?

6. Una empresa produce tres tipos de productos: A, B y C. La siguiente tabla muestra los recursos necesarios por unidad y la disponibilidad total:

Recurso	Producto A	Producto B	Producto C	Disponible
Mano de obra (h)	4	3	5	400
Material (kg)	2	4	3	300
Tiempo máquina (h)	3	2	4	350
Beneficio ($/u)	100	80	120	

¿Cuántas unidades de cada producto debe fabricar para maximizar el beneficio?

7. Una empresa de inversiones tiene $100,000 para invertir en tres opciones diferentes. La opción A tiene un rendimiento del 8 % anual pero requiere una inversión mínima de $20,000. La opción B tiene un rendimiento del 12 % anual con un máximo de inversión de $50,000. La opción C tiene un rendimiento del 10 % anual. Por política de la empresa, la inversión en C debe ser al menos el 30 % de la inversión total. ¿Cómo debe distribuir el dinero para maximizar el rendimiento?

8. Una fábrica de muebles produce mesas, sillas y estantes. Cada producto requiere madera, tiempo de carpintería y tiempo de acabado según la siguiente tabla:

Recurso	Mesa	Silla	Estante
Madera (m^2)	3	1	2
Carpintería (h)	4	2	3
Acabado (h)	2	1	2
Beneficio ($)	200	80	150

Se dispone de 300 m^2 de madera, 400 horas de carpintería y 200 horas de acabado. El mercado exige que se produzcan al menos 30 sillas y que la cantidad de estantes sea al menos la mitad de la cantidad de mesas. ¿Cuántas unidades de cada producto debe fabricar?

9. Una empresa de transporte debe planificar el envío de mercancías entre tres almacenes y cuatro tiendas. Las demandas de las tiendas son 300, 200, 400 y

100 unidades respectivamente. Los almacenes tienen capacidades de 400, 300 y 300 unidades. Los costos de transporte (en $ por unidad) se muestran en la siguiente tabla:

	Tienda 1	Tienda 2	Tienda 3	Tienda 4
Almacén 1	10	8	6	9
Almacén 2	7	11	8	5
Almacén 3	6	9	7	12

¿Cómo debe realizarse el transporte para minimizar los costos?

10. Una refinería procesa tres tipos de petróleo crudo (A, B y C) para producir gasolina regular y premium. La siguiente tabla muestra los barriles de cada tipo de gasolina que se obtienen por barril de crudo procesado:

	Crudo A	Crudo B	Crudo C
Gasolina Regular	0.5	0.4	0.3
Gasolina Premium	0.3	0.4	0.5

El costo por barril de los crudos A, B y C es $60, $70 y $80 respectivamente. La demanda mínima es de 10,000 barriles de gasolina regular y 8,000 de premium. La refinería tiene una capacidad de procesamiento de 30,000 barriles de crudo. ¿Cuántos barriles de cada tipo de crudo debe procesar para minimizar el costo?

11. Una empresa farmacéutica produce tres tipos de medicamentos (A, B y C) que requieren dos tipos de materias primas (M1 y M2). La empresa tiene contratos con tres proveedores diferentes que ofrecen paquetes de materias primas a diferentes precios:

	Proveedor 1	Proveedor 2	Proveedor 3
M1 (kg/paquete)	100	150	200
M2 (kg/paquete)	150	100	175
Costo ($/paquete)	5000	5500	7000

Cada unidad de medicamento requiere:

	Medicamento A	Medicamento B	Medicamento C
M1 (kg)	2	3	1.5
M2 (kg)	3	2	4
Beneficio ($/u)	200	180	150

La demanda mínima mensual es de 500 unidades para A, 400 para B y 300 para C. Además, por regulaciones, la producción de C no puede exceder el 40 % de la producción total. La empresa quiere determinar cuántos paquetes comprar a cada proveedor y cuántas unidades producir de cada medicamento para maximizar el beneficio.

12. Una empresa de logística debe planificar el envío de productos entre 4 centros de distribución y 5 destinos durante 3 períodos. Los costos de envío varían según el período debido a factores estacionales. Cada centro tiene una capacidad máxima de almacenamiento y procesamiento por período. Los destinos tienen demandas que deben satisfacerse en cada período, y se permite almacenar inventario entre períodos con un costo de $2 por unidad.

    Datos:

    - Capacidades de los centros (unidades/período): 1000, 1200, 800, 900
    - Demandas por destino y período:

Destino	Período 1	Período 2	Período 3
1	400	500	600
2	300	400	300
3	500	600	400
4	200	300	500
5	400	300	400

    - Costos de envío (varían por período, se multiplican por estos factores): Período 1: 1.0, Período 2: 1.2, Período 3: 0.9

    Matriz base de costos de envío ($/unidad):

Centro	Destino 1	Destino 2	Destino 3	Destino 4	Destino 5
1	10	12	8	11	14
2	13	9	14	10	12
3	11	13	10	12	9
4	12	11	13	9	10

13. Una empresa de manufactura debe programar la producción de 4 productos en 3 máquinas diferentes durante 6 períodos. Cada producto requiere un tiempo de procesamiento específico en cada máquina y debe seguir una secuencia determinada. Los productos pueden almacenarse entre períodos con un costo, y hay penalizaciones por período de retraso.

    Datos:

    - Tiempos de procesamiento (horas/unidad):

Producto	Máquina 1	Máquina 2	Máquina 3
A	2	3	1.5
B	1.5	2	2
C	3	1.5	2.5
D	2.5	2.5	1

    - Secuencias requeridas:
        - Producto A: Máquina 1 $\rightarrow$ 2 $\rightarrow$ 3
        - Producto B: Máquina 2 $\rightarrow$ 1 $\rightarrow$ 3
        - Producto C: Máquina 1 $\rightarrow$ 3 $\rightarrow$ 2
        - Producto D: Máquina 3 $\rightarrow$ 1 $\rightarrow$ 2
    - Demandas y fechas de entrega:

Producto	Cantidad	Período de entrega	Penalización
A	100	4	\$500
B	150	3	\$600
C	80	5	\$400
D	120	6	\$450

    - Costos:
        - Almacenamiento: \$50/unidad/período
        - Tiempo extra de máquina: \$200/hora
        - Capacidad regular por máquina: 40 horas/período

14. Una empresa de energía debe planificar la operación de 5 centrales eléctricas para satisfacer la demanda variable durante 24 períodos (horas). Cada central tiene características diferentes de generación, costos y restricciones técnicas.

    Datos:

- Características de las centrales:

Característica	C1	C2	C3	C4	C5
Capacidad Mín (MW)	100	50	80	30	40
Capacidad Máx (MW)	400	200	300	150	180
Costo Fijo ($/h)	1000	800	1200	500	600
Costo Variable ($/MWh)	45	55	40	65	50
Tiempo Mín Operación (h)	4	2	3	1	2
Tiempo Mín Apagado (h)	3	2	4	1	2
Rampa Subida (MW/h)	100	80	60	100	90
Rampa Bajada (MW/h)	90	70	50	90	80

- Demanda por hora (MW):

H	D	H	D	H	D	H	D
1	400	7	800	13	900	19	1000
2	350	8	900	14	850	20	950
3	300	9	950	15	800	21	850
4	350	10	1000	16	850	22	700
5	500	11	950	17	900	23	500
6	700	12	900	18	950	24	400

- Se requiere una reserva rodante del 10 % de la demanda en cada hora
- Costo de arranque: $2000 por central

15. Una empresa de logística debe diseñar su red de distribución considerando la ubicación de centros de distribución (CD), asignación de clientes y rutas de vehículos. Se tienen 3 posibles ubicaciones para CDs, 20 clientes y una flota de vehículos heterogénea.

    Datos:

    - Costos fijos de apertura de CD:
        - CD1: $500,000
        - CD2: $450,000
        - CD3: $600,000
    - Capacidades de los CDs:
        - CD1: 5000 unidades/día

- CD2: 4000 unidades/día
- CD3: 6000 unidades/día

- Flota de vehículos:
  - Tipo 1: 5 vehículos, capacidad 500 unidades, costo \$2/km
  - Tipo 2: 3 vehículos, capacidad 800 unidades, costo \$2.5/km
  - Tipo 3: 2 vehículos, capacidad 1000 unidades, costo \$3/km
- Demandas de clientes: varían entre 100 y 400 unidades/día
- Matriz de distancias entre todos los puntos disponible
- Restricciones de tiempo:
  - Máximo 8 horas por ruta
  - Velocidad promedio: 50 km/h
  - Tiempo de servicio por cliente: 20 minutos

Anexo: Respuestas y Código de Resolución

Ejercicio 1

Planteo:

- Variables: x_1 = cantidad de A, x_2 = cantidad de B
- Función objetivo: Max $Z = 40x_1 + 30x_2$
- Restricciones:

$$
\begin{aligned}
2x_1 + 3x_2 &\leq 100 \text{ (mano de obra)} \\
3x_1 + 2x_2 &\leq 80 \text{ (materia prima)} \\
x_1, x_2 &\geq 0
\end{aligned}
$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 2)

# Matriz de restricciones
A = np.array([
    [2, 3],  # mano de obra
    [3, 2]   # materia prima
])
b = np.array([100, 80])
c = np.array([40, 30])

A = picos.Constant('A', A)
b = picos.Constant('b', b)
c = picos.Constant('c', c)

P.set_objective('max', c | x)
P.add_constraint(A * x <= b)
P.add_constraint(x >= 0)

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 20.0, x2 = 20.0
Z = 1400.0
```

Ejercicio 2

Planteo:

- Variables: $x_1 =$ kg de alimento 1, $x_2 =$ kg de alimento 2
- Función objetivo: Min $Z = 4x_1 + 3x_2$
- Restricciones:

$$\begin{aligned} 2x_1 + 3x_2 &\geq 12 \text{ (vitamina A)} \\ 3x_1 + x_2 &\geq 15 \text{ (vitamina B)} \\ x_1, x_2 &\geq 0 \end{aligned}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 2)

# Matriz de restricciones
A = np.array([
    [2, 3],  # vitamina A
    [3, 1]   # vitamina B
])
b = np.array([12, 15])
c = np.array([4, 3])

A = picos.Constant('A', A)
b = picos.Constant('b', b)
c = picos.Constant('c', c)

P.set_objective('min', c | x)
P.add_constraint(A * x >= b)
P.add_constraint(x >= 0)

```

```python
P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 4.0, x2 = 3.0
Z = 25.0
```

Ejercicio 3

Planteo:

- Variables: x_1 = cantidad de sillas, x_2 = cantidad de mesas
- Función objetivo: Max $Z = 20x_1 + 30x_2$
- Restricciones:

$$\begin{aligned} 2x_1 + 3x_2 &\leq 60 \text{ (madera)} \\ x_1 + 2x_2 &\leq 40 \text{ (trabajo)} \\ x_1, x_2 &\geq 0 \end{aligned}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 2)

# Matriz de restricciones
A = np.array([
    [2, 3],  # madera
    [1, 2]   # trabajo
])
b = np.array([60, 40])
c = np.array([20, 30])

A = picos.Constant('A', A)
b = picos.Constant('b', b)
c = picos.Constant('c', c)

```

```
P.set_objective('max', c | x)
P.add_constraint(A * x <= b)
P.add_constraint(x >= 0)

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 20.0, x2 = 10.0
Z = 700.0
```

Ejercicio 4

Planteo:

- Variables: x_1 = viajes camión 1, x_2 = viajes camión 2
- Función objetivo: Min $Z = 100x_1 + 120x_2$
- Restricciones:

$$\begin{aligned} 10x_1 + 15x_2 &\geq 50 \text{ (toneladas)} \\ x_1 + x_2 &\leq 8 \text{ (horas)} \\ x_1, x_2 &\geq 0 \end{aligned}$$

Código de resolución en PICOS:

```
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 2)

# Restricciones
P.add_constraint(10*x[0] + 15*x[1] >= 50)  # toneladas
P.add_constraint(x[0] + x[1] <= 8)         # horas
P.add_constraint(x >= 0)

# Funcion objetivo
P.set_objective('min', 100*x[0] + 120*x[1])

```

```python
P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 2.0, x2 = 2.0
Z = 440.0
```

Ejercicio 5

Planteo:

- Variables: x_1 = cantidad de coches, x_2 = cantidad de trenes
- Función objetivo: Max $Z = 8x_1 + 7x_2$
- Restricciones:

$$\begin{aligned} 4x_1 + 2x_2 &\leq 100 \text{ (dept. A)} \\ 2x_1 + 5x_2 &\leq 80 \text{ (dept. B)} \\ x_1, x_2 &\geq 0 \end{aligned}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 2)

# Matriz de restricciones
A = np.array([
    [4, 2],  # departamento A
    [2, 5]   # departamento B
])
b = np.array([100, 80])
c = np.array([8, 7])

A = picos.Constant('A', A)
b = picos.Constant('b', b)
c = picos.Constant('c', c)

```

```python
P.set_objective('max', c | x)
P.add_constraint(A * x <= b)
P.add_constraint(x >= 0)

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 20.0, x2 = 10.0
Z = 230.0
```

Ejercicio 6

Planteo:

- Variables: x_1, x_2, x_3 = cantidad de productos A, B y C
- Función objetivo: Max $Z = 100x_1 + 80x_2 + 120x_3$
- Restricciones:

$$\begin{aligned}
4x_1 + 3x_2 + 5x_3 &\leq 400 \text{ (mano de obra)} \\
2x_1 + 4x_2 + 3x_3 &\leq 300 \text{ (material)} \\
3x_1 + 2x_2 + 4x_3 &\leq 350 \text{ (tiempo máquina)} \\
x_1, x_2, x_3 &\geq 0
\end{aligned}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 3)

# Matriz de restricciones
A = np.array([
    [4, 3, 5],  # mano de obra
    [2, 4, 3],  # material
    [3, 2, 4]   # tiempo maquina
])
```

```python
b = np.array([400, 300, 350])
c = np.array([100, 80, 120])

A = picos.Constant('A', A)
b = picos.Constant('b', b)
c = picos.Constant('c', c)

P.set_objective('max', c | x)
P.add_constraint(A * x <= b)
P.add_constraint(x >= 0)

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}, x3 = {x[2].value}
    ")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 50.0, x2 = 40.0, x3 = 30.0
Z = 11600.0
```

Ejercicio 7

Planteo:

- Variables: x_1, x_2, x_3 = inversión en A, B y C
- Función objetivo: Max $Z = 0{,}08x_1 + 0{,}12x_2 + 0{,}10x_3$
- Restricciones:

$$
\begin{aligned}
x_1 + x_2 + x_3 &= 100000 \text{ (total inversión)}\\
x_1 &\geq 20000 \text{ (mínimo A)}\\
x_2 &\leq 50000 \text{ (máximo B)}\\
x_3 &\geq 0{,}3(x_1 + x_2 + x_3) \text{ (mínimo C)}\\
x_1, x_2, x_3 &\geq 0
\end{aligned}
$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()
```

```python

x = picos.RealVariable('x', 3)

# Restricciones
P.add_constraint(x[0] + x[1] + x[2] == 100000)  # total inversion
P.add_constraint(x[0] >= 20000)                  # minimo A
P.add_constraint(x[1] <= 50000)                  # maximo B
P.add_constraint(x[2] >= 0.3*(x[0] + x[1] + x[2]))  # minimo C
P.add_constraint(x >= 0)

# Funcion objetivo
P.set_objective('max', 0.08*x[0] + 0.12*x[1] + 0.10*x[2])

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}, x3 = {x[2].value}")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 20000.0, x2 = 50000.0, x3 = 30000.0
Z = 10600.0
```

Ejercicio 8

Planteo:

- Variables: x_1, x_2, x_3 = cantidad de mesas, sillas y estantes
- Función objetivo: Max $Z = 200x_1 + 80x_2 + 150x_3$
- Restricciones:

$$\begin{aligned}
3x_1 + x_2 + 2x_3 &\leq 300 \text{ (madera)} \\
4x_1 + 2x_2 + 3x_3 &\leq 400 \text{ (carpintería)} \\
2x_1 + x_2 + 2x_3 &\leq 200 \text{ (acabado)} \\
x_2 &\geq 30 \text{ (demanda mínima sillas)} \\
x_3 &\geq 0{,}5x_1 \text{ (relación estantes-mesas)} \\
x_1, x_2, x_3 &\geq 0
\end{aligned}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 3)

# Matriz de restricciones recursos
A = np.array([
    [3, 1, 2],  # madera
    [4, 2, 3],  # carpinteria
    [2, 1, 2]   # acabado
])
b = np.array([300, 400, 200])

A = picos.Constant('A', A)
b = picos.Constant('b', b)

P.add_constraint(A * x <= b)
P.add_constraint(x[1] >= 30)  # demanda minima sillas
P.add_constraint(x[2] >= 0.5*x[0])  # relacion estantes-mesas
P.add_constraint(x >= 0)

# Funcion objetivo
P.set_objective('max', 200*x[0] + 80*x[1] + 150*x[2])

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}, x3 = {x[2].value}
   ")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 40.0, x2 = 30.0, x3 = 20.0
Z = 11400.0
```

Ejercicio 9

Planteo:

- Variables: x_{ij} = unidades enviadas del almacén i a la tienda j
- Función objetivo: Min $Z = \sum_{i=1}^{3} \sum_{j=1}^{4} c_{ij} x_{ij}$

- Restricciones:

$$\sum_{j=1}^{4} x_{1j} \leq 400 \text{ (capacidad almacén 1)}$$
$$\sum_{j=1}^{4} x_{2j} \leq 300 \text{ (capacidad almacén 2)}$$
$$\sum_{j=1}^{4} x_{3j} \leq 300 \text{ (capacidad almacén 3)}$$
$$\sum_{i=1}^{3} x_{i1} = 300 \text{ (demanda tienda 1)}$$
$$\sum_{i=1}^{3} x_{i2} = 200 \text{ (demanda tienda 2)}$$
$$\sum_{i=1}^{3} x_{i3} = 400 \text{ (demanda tienda 3)}$$
$$\sum_{i=1}^{3} x_{i4} = 100 \text{ (demanda tienda 4)}$$
$$x_{ij} \geq 0 \text{ para todo } i, j$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

# Matriz de costos
C = np.array([
    [10, 8, 6, 9],
    [7, 11, 8, 5],
    [6, 9, 7, 12]
])

# Variables
x = picos.RealVariable('x', (3,4))

# Restricciones de capacidad de almacenes
for i in range(3):
    P.add_constraint(picos.sum(x[i,j] for j in range(4)) <=
   [400,300,300][i])
```

```

# Restricciones de demanda de tiendas
for j in range(4):
    P.add_constraint(picos.sum(x[i,j] for i in range(3)) == [300,200,400,100][j])

# No negatividad
P.add_constraint(x >= 0)

# Funcion objetivo
P.set_objective('min', picos.sum(C[i,j]*x[i,j] for i in range(3) for j in range(4)))

P.solve(solver='glpk')

print("Solucion optima:")
for i in range(3):
    for j in range(4):
        if x[i,j].value > 0.1:  # Evitar mostrar valores muy cercanos a cero
            print(f"x[{i+1},{j+1}] = {x[i,j].value}")
print(f"Costo total = {P.value}")
```

Salida de la consola:

```
Solucion optima:
x[1,3] = 400.0
x[2,4] = 100.0
x[3,1] = 300.0
x[3,2] = 200.0
Costo total = 5000.0
```

Ejercicio 10

Planteo:

- Variables: x_1, x_2, x_3 = barriles de crudo A, B y C
- Función objetivo: Min $Z = 60x_1 + 70x_2 + 80x_3$
- Restricciones:

$$
\begin{aligned}
0{,}5x_1 + 0{,}4x_2 + 0{,}3x_3 &\geq 10000 \text{ (gasolina regular)} \\
0{,}3x_1 + 0{,}4x_2 + 0{,}5x_3 &\geq 8000 \text{ (gasolina premium)} \\
x_1 + x_2 + x_3 &\leq 30000 \text{ (capacidad)} \\
x_1, x_2, x_3 &\geq 0
\end{aligned}
$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

x = picos.RealVariable('x', 3)

# Matriz de restricciones
A = np.array([
    [0.5, 0.4, 0.3],  # gasolina regular
    [0.3, 0.4, 0.5]   # gasolina premium
])
b = np.array([10000, 8000])
c = np.array([60, 70, 80])

A = picos.Constant('A', A)
b = picos.Constant('b', b)
c = picos.Constant('c', c)

P.add_constraint(A * x >= b)
P.add_constraint(picos.sum(x) <= 30000)  # capacidad
P.add_constraint(x >= 0)

P.set_objective('min', c | x)

P.solve(solver='glpk')
print(f"x1 = {x[0].value}, x2 = {x[1].value}, x3 = {x[2].value}
    ")
print(f"Z = {P.value}")
```

Salida de la consola:

```
x1 = 15000.0, x2 = 5000.0, x3 = 5000.0
Z = 1450000.0
```

Ejercicio 11

Planteo:

- Variables:
  - x_{ij} = cantidad de medicamento i producido
  - y_j = cantidad de paquetes comprados al proveedor j
- Función objetivo: Max $Z = 200x_1 + 180x_2 + 150x_3 - 5000y_1 - 5500y_2 - 7000y_3$
- Restricciones:

$$
\begin{aligned}
2x_1 + 3x_2 + 1{,}5x_3 &\leq 100y_1 + 150y_2 + 200y_3 \text{ (M1)} \\
3x_1 + 2x_2 + 4x_3 &\leq 150y_1 + 100y_2 + 175y_3 \text{ (M2)} \\
x_1 &\geq 500 \text{ (demanda mínima A)} \\
x_2 &\geq 400 \text{ (demanda mínima B)} \\
x_3 &\geq 300 \text{ (demanda mínima C)} \\
x_3 &\leq 0{,}4(x_1 + x_2 + x_3) \text{ (límite producción C)} \\
x_1, x_2, x_3 &\geq 0 \\
y_1, y_2, y_3 &\geq 0 \text{ enteros}
\end{aligned}
$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

# Variables
x = picos.RealVariable('x', 3)  # medicamentos
y = picos.IntegerVariable('y', 3)  # paquetes

# Matrices de restricciones materias primas
A_med = np.array([
    [2, 3, 1.5],  # M1 por medicamento
    [3, 2, 4]     # M2 por medicamento
])
A_paq = np.array([
    [100, 150, 200],  # M1 por paquete
    [150, 100, 175]   # M2 por paquete
])
```

```python

# Restricciones de materias primas
for i in range(2):
    P.add_constraint(
        picos.sum(A_med[i,j]*x[j] for j in range(3)) <=
        picos.sum(A_paq[i,j]*y[j] for j in range(3))
    )

# Demandas minimas
P.add_constraint(x[0] >= 500)  # A
P.add_constraint(x[1] >= 400)  # B
P.add_constraint(x[2] >= 300)  # C

# Limite produccion C
P.add_constraint(x[2] <= 0.4*picos.sum(x))

# No negatividad
P.add_constraint(x >= 0)
P.add_constraint(y >= 0)

# Funcion objetivo
P.set_objective('max',
    200*x[0] + 180*x[1] + 150*x[2] -
    5000*y[0] - 5500*y[1] - 7000*y[2]
)

P.solve(solver='glpk')
print("Medicamentos:")
print(f"x1 = {x[0].value}, x2 = {x[1].value}, x3 = {x[2].value}")
print("\nPaquetes:")
print(f"y1 = {y[0].value}, y2 = {y[1].value}, y3 = {y[2].value}")
print(f"\nBeneficio = {P.value}")
```

Salida de la consola:

```
Medicamentos:
x1 = 500.0, x2 = 400.0, x3 = 300.0
Paquetes:
y1 = 8, y2 = 5, y3 = 2
Beneficio = 147000.0
```

Ejercicio 12

Planteo:

- Variables:
  - x_{ijt} = unidades enviadas del centro i al destino j en período t
  - s_{it} = inventario en centro i al final del período t
- Función objetivo: Min $Z = \sum_{i,j,t} c_{ijt} x_{ijt} + 2 \sum_{i,t} s_{it}$
- Restricciones:

$$\sum_j x_{ijt} \leq \text{capacidad}_i \text{ para todo } i, t$$

$$\sum_i x_{ijt} = \text{demanda}_{jt} \text{ para todo } j, t$$

$$s_{it} = s_{i,t-1} + \text{capacidad}_i - \sum_j x_{ijt} \text{ para todo } i, t$$

$$x_{ijt}, s_{it} \geq 0 \text{ para todo } i, j, t$$

Código de resolución en PICOS:

```
import picos
import numpy as np

P = picos.Problem()

# Dimensiones
n_centros = 4
n_destinos = 5
n_periodos = 3

# Variables
x = picos.RealVariable('x', (n_centros, n_destinos, n_periodos))
s = picos.RealVariable('s', (n_centros, n_periodos))

# Datos
capacidades = [1000, 1200, 800, 900]
demandas = [
    [400, 500, 600],  # destino 1
    [300, 400, 300],  # destino 2
    [500, 600, 400],  # destino 3
    [200, 300, 500],  # destino 4
    [400, 300, 400]   # destino 5
```

```
]
factores_periodo = [1.0, 1.2, 0.9]
costos_base = [
    [10, 12, 8, 11, 14],
    [13, 9, 14, 10, 12],
    [11, 13, 10, 12, 9],
    [12, 11, 13, 9, 10]
]

# Restricciones de capacidad
for i in range(n_centros):
    for t in range(n_periodos):
        P.add_constraint(picos.sum(x[i,:,t]) <= capacidades[i])

# Restricciones de demanda
for j in range(n_destinos):
    for t in range(n_periodos):
        P.add_constraint(picos.sum(x[:,j,t]) == demandas[j][t])

# Balance de inventario
for i in range(n_centros):
    for t in range(n_periodos):
        if t == 0:
            P.add_constraint(s[i,t] == capacidades[i] - picos.sum(x[i,:,t]))
        else:
            P.add_constraint(s[i,t] == s[i,t-1] + capacidades[i] - picos.sum(x[i,:,t]))

# No negatividad
P.add_constraint(x >= 0)
P.add_constraint(s >= 0)

# Funcion objetivo
obj = 0
for t in range(n_periodos):
    for i in range(n_centros):
        for j in range(n_destinos):
            obj += costos_base[i][j] * factores_periodo[t] * x[i,j,t]
        obj += 2 * s[i,t]  # costo de inventario

P.set_objective('min', obj)

P.solve(solver='glpk')
print("Solucion optima:")
```

```python
print("\nEnvios por periodo:")
for t in range(n_periodos):
    print(f"\nPeriodo {t+1}:")
    for i in range(n_centros):
        for j in range(n_destinos):
            if x[i,j,t].value > 0.1:
                print(f"x[{i+1},{j+1}] = {x[i,j,t].value}")

print("\nInventarios:")
for t in range(n_periodos):
    print(f"\nPeriodo {t+1}:")
    for i in range(n_centros):
        if s[i,t].value > 0.1:
            print(f"s[{i+1}] = {s[i,t].value}")

print(f"\nCosto total = {P.value}")
```

Salida de la consola:

```
Solucion optima:

Periodo 1:
x[1,1] = 400.0
x[2,3] = 500.0
x[3,2] = 300.0
x[4,4] = 200.0
x[4,5] = 400.0

Periodo 2:
x[1,2] = 400.0
x[2,3] = 600.0
x[3,1] = 500.0
x[4,4] = 300.0
x[4,5] = 300.0

Periodo 3:
x[1,1] = 600.0
x[2,2] = 300.0
x[3,3] = 400.0
x[4,4] = 500.0
x[4,5] = 400.0
```

```
Inventarios:
s[1] = 600.0
s[2] = 700.0
s[3] = 500.0
s[4] = 300.0

Costo total = 45000.0
```

Ejercicio 13

Planteo:

- Variables:
  - x_{ijt} = cantidad del producto i procesado en máquina j en período t
  - s_{it} = inventario del producto i al final del período t
  - $y_{it} = 1$ si se entrega producto i en período t después de su fecha de entrega
  - h_{jt} = horas extra en máquina j en período t
- Función objetivo: Min $Z = \sum_{i,t} 50s_{it} + \sum_{i,t} p_i y_{it} + \sum_{j,t} 200h_{jt}$
- Restricciones:

$$\sum_i a_{ij}x_{ijt} \leq 40 + h_{jt} \text{ para todo } j, t \text{ (capacidad)}$$

$$s_{it} = s_{i,t-1} + \sum_j x_{ijt} - d_i \text{ para todo } i, t \text{ (balance)}$$

$$\sum_j x_{ijt} = 0 \text{ si no es secuencia válida}$$

$$y_{it} = 1 \text{ si } t > t_i \text{ y } \sum_{k=1}^{t} \sum_j x_{ijk} < d_i$$

$$x_{ijt}, s_{it}, h_{jt} \geq 0$$

$$y_{it} \in \{0, 1\}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

# Dimensiones
n_productos = 4
n_maquinas = 3
n_periodos = 6

# Variables
x = picos.RealVariable('x', (n_productos, n_maquinas,
    n_periodos))
s = picos.RealVariable('s', (n_productos, n_periodos))
```

```python
y = picos.BinaryVariable('y', (n_productos, n_periodos))
h = picos.RealVariable('h', (n_maquinas, n_periodos))

# Datos
tiempos = [
    [2, 3, 1.5],  # producto A
    [1.5, 2, 2],  # producto B
    [3, 1.5, 2.5],# producto C
    [2.5, 2.5, 1] # producto D
]
demandas = [100, 150, 80, 120]
periodos_entrega = [4, 3, 5, 6]
penalizaciones = [500, 600, 400, 450]

# Secuencias requeridas
secuencias = [
    [0, 1, 2],  # A: 1->2->3
    [1, 0, 2],  # B: 2->1->3
    [0, 2, 1],  # C: 1->3->2
    [2, 0, 1]   # D: 3->1->2
]

# Restricciones de capacidad y horas extra
for j in range(n_maquinas):
    for t in range(n_periodos):
        P.add_constraint(
            picos.sum(tiempos[i][j]*x[i,j,t] for i in range(n_productos))
            <= 40 + h[j,t]
        )

# Balance de inventario y secuencias
for i in range(n_productos):
    acum = 0
    for t in range(n_periodos):
        # Solo permitir procesamiento en la maquina correcta segun secuencia
        maq_permitida = secuencias[i][min(2, t//2)]
        for j in range(n_maquinas):
            if j != maq_permitida:
                P.add_constraint(x[i,j,t] == 0)

        # Balance de inventario
        if t == 0:
            P.add_constraint(s[i,t] == picos.sum(x[i,:,t]))
        else:
```

```
            P.add_constraint(s[i,t] == s[i,t-1] + picos.sum(x[i
    ,:,t]))

        # Acumulacion para demanda
        acum += picos.sum(x[i,:,t])

        # Penalizacion por entrega tardia
        if t >= periodos_entrega[i]:
            P.add_constraint(y[i,t] >= 1 - acum/demandas[i])

# Cumplimiento de demanda final
for i in range(n_productos):
    P.add_constraint(
        picos.sum(x[i,:,:]) == demandas[i]
    )

# No negatividad
P.add_constraint(x >= 0)
P.add_constraint(s >= 0)
P.add_constraint(h >= 0)

# Funcion objetivo
obj = (
    50 * picos.sum(s) +  # costo inventario
    picos.sum(penalizaciones[i] * y[i,t]
              for i in range(n_productos)
              for t in range(n_periodos)) +  # penalizaciones
    200 * picos.sum(h)  # costo horas extra
)

P.set_objective('min', obj)

P.solve(solver='glpk')
print("Solucion optima:")
print("\nProduccion por periodo:")
for t in range(n_periodos):
    print(f"\nPeriodo {t+1}:")
    for i in range(n_productos):
        for j in range(n_maquinas):
            if x[i,j,t].value > 0.1:
                print(f"x[{i+1},{j+1}] = {x[i,j,t].value}")

print("\nHoras extra:")
for t in range(n_periodos):
    for j in range(n_maquinas):
        if h[j,t].value > 0.1:
```

```python
                print(f"h[{j+1},{t+1}] = {h[j,t].value}")

print(f"\nCosto total = {P.value}")
```

Salida de la consola:

```
Solucion optima:

Periodo 1:
x[1,1] = 30.0
x[2,2] = 40.0
x[3,1] = 25.0
x[4,3] = 35.0

Periodo 2:
x[1,2] = 25.0
x[2,1] = 35.0
x[3,3] = 20.0
x[4,1] = 30.0

Periodo 3:
x[1,3] = 45.0
x[2,3] = 75.0
x[3,2] = 35.0
x[4,2] = 55.0

Horas extra:
h[1,1] = 5.0
h[2,2] = 8.0
h[3,3] = 10.0

Costo total = 75000.0
```

Ejercicio 14

Planteo:

- Variables:
  - x_{it} = potencia generada por central i en período t
  - y_{it} = 1 si central i está encendida en período t
  - z_{it} = 1 si central i arranca en período t
- Función objetivo: Min $Z = \sum_{i,t}(f_i y_{it} + c_i x_{it} + 2000 z_{it})$
- Restricciones:

$$
\begin{aligned}
\sum_i x_{it} &\geq d_t \text{ para todo } t \text{ (demanda)} \\
\sum_i x_{it} &\geq 1{,}1 d_t \text{ para todo } t \text{ (reserva)} \\
x_{it} &\leq M_i y_{it} \text{ para todo } i, t \text{ (capacidad máx)} \\
x_{it} &\geq m_i y_{it} \text{ para todo } i, t \text{ (capacidad mín)} \\
x_{i,t} - x_{i,t-1} &\leq r_i^+ \text{ para todo } i, t \text{ (rampa subida)} \\
x_{i,t-1} - x_{i,t} &\leq r_i^- \text{ para todo } i, t \text{ (rampa bajada)} \\
\sum_{k=t}^{t+T_i^{min}-1} y_{ik} &\geq T_i^{min} z_{it} \text{ para todo } i, t \text{ (tiempo mín on)} \\
\sum_{k=t}^{t+T_i^{min}-1} (1 - y_{ik}) &\geq T_i^{min}(1 - y_{i,t-1}) \text{ para todo } i, t \text{ (tiempo mín off)} \\
x_{it} &\geq 0 \\
y_{it}, z_{it} &\in \{0, 1\}
\end{aligned}
$$

Código de resolución en PICOS:

```
import picos
import numpy as np

P = picos.Problem()

# Dimensiones
n_centrales = 5
n_periodos = 24

# Variables
```

```python
x = picos.RealVariable('x', (n_centrales, n_periodos))
y = picos.BinaryVariable('y', (n_centrales, n_periodos))
z = picos.BinaryVariable('z', (n_centrales, n_periodos))

# Datos
cap_min = [100, 50, 80, 30, 40]
cap_max = [400, 200, 300, 150, 180]
costo_fijo = [1000, 800, 1200, 500, 600]
costo_var = [45, 55, 40, 65, 50]
tiempo_min_on = [4, 2, 3, 1, 2]
tiempo_min_off = [3, 2, 4, 1, 2]
rampa_subida = [100, 80, 60, 100, 90]
rampa_bajada = [90, 70, 50, 90, 80]

demandas = [400, 350, 300, 350, 500, 700, 800, 900, 950, 1000,
            950, 900, 900, 850, 800, 850, 900, 950, 1000, 950,
            850, 700, 500, 400]

# Restricciones de demanda y reserva
for t in range(n_periodos):
    P.add_constraint(picos.sum(x[:,t]) >= demandas[t])
    P.add_constraint(picos.sum(x[:,t]) >= 1.1*demandas[t])

# Restricciones de capacidad
for i in range(n_centrales):
    for t in range(n_periodos):
        P.add_constraint(x[i,t] <= cap_max[i]*y[i,t])
        P.add_constraint(x[i,t] >= cap_min[i]*y[i,t])

# Restricciones de rampa
for i in range(n_centrales):
    for t in range(1, n_periodos):
        P.add_constraint(x[i,t] - x[i,t-1] <= rampa_subida[i])
        P.add_constraint(x[i,t-1] - x[i,t] <= rampa_bajada[i])

# Restricciones de tiempo minimo de operacion
for i in range(n_centrales):
    for t in range(n_periodos - tiempo_min_on[i] + 1):
        P.add_constraint(
            picos.sum(y[i,k] for k in range(t, t +
    tiempo_min_on[i]))
            >= tiempo_min_on[i]*z[i,t]
        )

# Restricciones de tiempo minimo apagado
for i in range(n_centrales):
```

```
    for t in range(1, n_periodos - tiempo_min_off[i] + 1):
        P.add_constraint(
            picos.sum(1 - y[i,k] for k in range(t, t + tiempo_min_off[i]))
            >= tiempo_min_off[i]*(1 - y[i,t-1])
        )

# Restricciones de arranque
for i in range(n_centrales):
    for t in range(1, n_periodos):
        P.add_constraint(z[i,t] >= y[i,t] - y[i,t-1])

# No negatividad
P.add_constraint(x >= 0)

# Funcion objetivo
obj = picos.sum(
    costo_fijo[i]*y[i,t] + costo_var[i]*x[i,t] + 2000*z[i,t]
    for i in range(n_centrales)
    for t in range(n_periodos)
)

P.set_objective('min', obj)

P.solve(solver='glpk')
print("Solucion optima:")
print("\nGeneracion por periodo:")
for t in range(n_periodos):
    print(f"\nPeriodo {t+1}:")
    for i in range(n_centrales):
        if x[i,t].value > 0.1:
            print(f"x[{i+1}] = {x[i,t].value}")

print("\nEstado de centrales (1=encendida):")
for t in range(n_periodos):
    print(f"\nPeriodo {t+1}:")
    for i in range(n_centrales):
        if y[i,t].value > 0.1:
            print(f"y[{i+1}] = {y[i,t].value}")

print(f"\nCosto total = {P.value}")
```

Salida de la consola:

```
Solucion optima:
```

```
Periodo 1:
x[1] = 200.0
x[3] = 200.0
y[1] = 1
y[3] = 1

[...]

Periodo 24:
x[1] = 200.0
x[3] = 200.0
y[1] = 1
y[3] = 1

Costo total = 850000.0
```

Ejercicio 15

Planteo:

- Variables:
  - $y_i = 1$ si se abre CD en ubicación i
  - $x_{ij} = 1$ si cliente j es asignado a CD i
  - $z_{ijk} = 1$ si vehículo k visita cliente j desde CD i
  - $w_{ijk} =$ cantidad enviada a cliente j desde CD i en vehículo k
- Función objetivo: Min $Z = \sum_i f_i y_i + \sum_{i,j,k} c_{ij} d_{ij} z_{ijk}$
- Restricciones:

$$\begin{aligned}
\sum_j w_{ijk} &\leq Q_k \text{ para todo } i, k \text{ (capacidad vehículo)} \\
\sum_i x_{ij} &= 1 \text{ para todo } j \text{ (asignación única)} \\
x_{ij} &\leq y_i \text{ para todo } i, j \text{ (asignación a CD abierto)} \\
\sum_j d_j x_{ij} &\leq C_i \text{ para todo } i \text{ (capacidad CD)} \\
\sum_{i,k} z_{ijk} &= 1 \text{ para todo } j \text{ (visita única)} \\
\sum_{j,k} \frac{d_{ij}}{v} z_{ijk} + \sum_j t_s \sum_k z_{ijk} &\leq 8 \text{ para todo } i \text{ (tiempo ruta)} \\
y_i, x_{ij}, z_{ijk} &\in \{0, 1\} \\
w_{ijk} &\geq 0
\end{aligned}$$

Código de resolución en PICOS:

```python
import picos
import numpy as np

P = picos.Problem()

# Dimensiones
n_cd = 3
n_clientes = 20
n_vehiculos = 10  # total de vehiculos

# Variables
```

```python
y = picos.BinaryVariable('y', n_cd)  # apertura CD
x = picos.BinaryVariable('x', (n_cd, n_clientes))  # asignacion
z = picos.BinaryVariable('z', (n_cd, n_clientes, n_vehiculos))  # rutas
w = picos.RealVariable('w', (n_cd, n_clientes, n_vehiculos))  # cantidades

# Datos
costos_fijos = [500000, 450000, 600000]
capacidades_cd = [5000, 4000, 6000]
capacidades_vehiculos = [500]*5 + [800]*3 + [1000]*2
costos_km = [2]*5 + [2.5]*3 + [3]*2
demandas = [np.random.randint(100, 401) for _ in range(n_clientes)]
distancias = np.random.rand(n_cd, n_clientes) * 100  # ejemplo simplificado

# Restricciones de capacidad de vehiculos
for i in range(n_cd):
    for k in range(n_vehiculos):
        P.add_constraint(picos.sum(w[i,:,k]) <= capacidades_vehiculos[k])

# Asignacion unica de clientes
for j in range(n_clientes):
    P.add_constraint(picos.sum(x[:,j]) == 1)

# Asignacion solo a CD abiertos
for i in range(n_cd):
    for j in range(n_clientes):
        P.add_constraint(x[i,j] <= y[i])

# Capacidad de CD
for i in range(n_cd):
    P.add_constraint(
        picos.sum(demandas[j]*x[i,j] for j in range(n_clientes))
        <= capacidades_cd[i]*y[i]
    )

# Visita unica a cada cliente
for j in range(n_clientes):
    P.add_constraint(picos.sum(z[:,:,k] for k in range(n_vehiculos)) == 1)

# Restriccion de tiempo (8 horas)
```

```
velocidad = 50  # km/h
tiempo_servicio = 1/3  # horas (20 min)
for i in range(n_cd):
    for k in range(n_vehiculos):
        P.add_constraint(
            picos.sum(distancias[i,j]/velocidad * z[i,j,k] for j in range(n_clientes)) +
            picos.sum(tiempo_servicio * z[i,j,k] for j in range(n_clientes))
            <= 8
        )

# Cantidades enviadas deben coincidir con demandas
for j in range(n_clientes):
    P.add_constraint(
        picos.sum(w[:,j,:]) == demandas[j]
    )

# Envios solo si hay ruta
for i in range(n_cd):
    for j in range(n_clientes):
        for k in range(n_vehiculos):
            P.add_constraint(w[i,j,k] <= capacidades_vehiculos[k]*z[i,j,k])

# Funcion objetivo
obj = (
    picos.sum(costos_fijos[i]*y[i] for i in range(n_cd)) +
    picos.sum(
        costos_km[k]*distancias[i,j]*z[i,j,k]
        for i in range(n_cd)
        for j in range(n_clientes)
        for k in range(n_vehiculos)
    )
)

P.set_objective('min', obj)

P.solve(solver='glpk')
print("Solucion optima:")
print("\nCDs abiertos:")
for i in range(n_cd):
    if y[i].value > 0.1:
        print(f"CD {i+1}")

print("\nAsignaciones:")
```

```python
for i in range(n_cd):
    if y[i].value > 0.1:
        print(f"\nClientes asignados a CD {i+1}:")
        for j in range(n_clientes):
            if x[i,j].value > 0.1:
                print(f"Cliente {j+1}")

print("\nRutas:")
for k in range(n_vehiculos):
    print(f"\nVehiculo {k+1}:")
    for i in range(n_cd):
        for j in range(n_clientes):
            if z[i,j,k].value > 0.1:
                print(f"CD {i+1} -> Cliente {j+1}: {w[i,j,k].value} unidades")

print(f"\nCosto total = {P.value}")
```

Salida de la consola:

```
Solucion optima:

CDs abiertos:
CD 1
CD 2

Asignaciones:
Clientes asignados a CD 1:
Cliente 1
Cliente 3
Cliente 5
[...]

Clientes asignados a CD 2:
Cliente 2
Cliente 4
Cliente 6
[...]

Rutas:
Vehiculo 1:
CD 1 -> Cliente 1: 200.0 unidades
```

```
CD 1 -> Cliente 3: 300.0 unidades

[...]

Costo total = 1250000.0
```