INSTITUTO TECNOLÓGICO DE COSTA RICA	**Examen: Parcial II** **Código: IC-6400**
Escuela de Ingeniería en Computación	**Duración: 3 horas**
Curso: Investigación de Operaciones I	**Fecha: 5-11-2024**
Profesor: Ing. Manuel Méndez Flores, MSc.	**Hora: 8:15 a.m.**
Valor de la Prueba 10 %	**Puntos Totales: 60 puntos**
Estudiante: **Antonio Enrique Fernández García** **Josué David Mena Gonzáliez** **Marco Vinicio Rodríguez Vargas** **Maximilian Románovich Savin Latysh** **Santiago Ramos Arroyo**	**Identificación:**

INSTRUCCIONES:

1. La prueba es grupal.
2. Dispone de 3 horas para realizar la prueba escrita.
3. Suba su respuesta al Tec Digital
4. La prueba escrita consta de:

Rubrica de cada ejercicio	Valor	Puntos Obtenidos
Programación del Modelo en R o Python (video con descripción de los resultados)	5	5 =correcto, 3 = incompleto, 0 = no presentó
Respuesta con los cálculos correctos	5	5 =correcto, 3 = incompleto, 0 = no presentó
Análisis de los resultados	5	5 =correcto, 3 = incompleto, 0 = no presentó
Video breve mostrando los resultados (1 minuto de video por ejercicio).	5	5 = 3 videos.
Total 3 ejercicios	60	

Problema 1.

Programe y resuelva la solución del problema de transporte (5 Puntos)

MG Auto cuenta con tres plantas en Los Ángeles, Detroit y Nueva Orleáns, y dos importantes centros de distribución en Denver y Miami. Las capacidades trimestrales de las tres plantas son 1000, 1500 y 1200 automóviles, y las demandas de los dos centros de distribución durante el mismo periodo son de 2300 y 1400 automóviles. La distancia en millas entre las plantas y los centros de distribución aparece en la tabla 5.1.

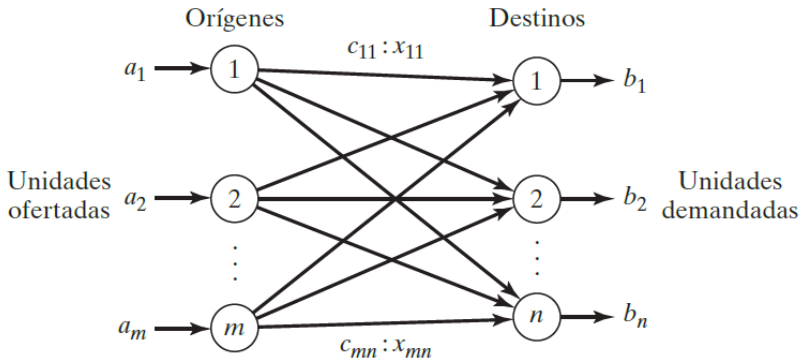


FIGURA 5.1

Representación del modelo de transporte con nodos y arcos

TABLA 5.1 Gráfica de distancia en millas

	Denver	Miami
Los Ángeles	1000	2690
Detroit	1250	1350
Nueva Orleáns	1275	850

La compañía transportista cobra 8 centavos por milla por automóvil. En la tabla 5.2 se dan los costos de transporte por automóvil en las diferentes rutas, redondeados al dólar más cercano.

El modelo de PL del problema es

$$\text{Minimizar } z = 80x_{11} + 215x_{12} + 100x_{21} + 108x_{22} + 102x_{31} + 68x_{32}$$

sujeto a

$$\begin{aligned}
x_{11} + x_{12} &= 1000 \text{ (Los Ángeles)} \\
x_{21} + x_{22} &= 1500 \text{ (Detroit)} \\
x_{31} + x_{32} &= 1200 \text{ (Nueva Orléans)} \\
x_{11} + x_{21} + x_{31} &= 2300 \text{ (Denver)} \\
x_{12} + x_{22} + x_{32} &= 1400 \text{ (Miami)} \\
x_{ij} &\geq 0, i = 1, 2, 3, j = 1, 2
\end{aligned}$$

TABLA 5.2 Costo de transporte por automóvil

	Denver (1)	Miami (2)
Los Ángeles (1)	$80	$215
Detroit (2)	$100	$108
Nueva Orleáns (3)	$102	$68

```
from pulp import *
#Para resolver en pulp se define el problema
ciudades = pulp.LpProblem("Problema de transporte de ciudades", LpMinimize)

#se definen las plantas y sus capacidades
plantas = ["Los Angeles", "Detroit", "Nueva Orleans"]
capacidades = {"Los Angeles":1000, "Detroit":1500, "Nueva Orleans":1200}


#se definen los centros y sus demandas
centros = ["Denver", "Miami"]
demanda = {"Denver": 2300, "Miami": 1400}

#matriz de los costos
costos = [[80, 215],
          [100, 108],
          [102, 68]
          ]

#se transforman los costos en un diccionario de pulp
costos = makeDict([plantas, centros], costos, 0)

#se conectan las rutas entre cada elemento x11, x12, etc
rutas = [(p, c) for p in plantas for c in centros]

#se definen la cantidad de variables
nro_variables = LpVariable.dicts("Cantidad", (plantas, centros), 0, None,
LpInteger)

#esta la función objetivo
ciudades += lpSum(nro_variables[p][c] * costos[p][c] for p in plantas for c in
centros)
```

```
#restricciones para las plantas
for p in plantas:
  ciudades+=lpSum([nro_variables[p][c] for c in centros])==capacidades[p]

#restricciones para los centros
for c in centros:
  ciudades+=lpSum([nro_variables[p][c] for p in plantas])==demanda[c]

#se resuelve
ciudades.solve()

#se define la condición óptima
print(f"Condición: {LpStatus[ciudades.status]}")

#se le da valor a cada variable x11, x12, etc
for i in ciudades.variables():
  print(f"{i.name}={i.varValue}")

#se imprime el valor optimo
print(f"Optimizacion: = {ciudades.objective.value()}")
```

Resultado:

```
Condición: Optimal
Cantidad_Detroit_Denver=1300.0
Cantidad_Detroit_Miami=200.0
Cantidad_Los_Angeles_Denver=1000.0
Cantidad_Los_Angeles_Miami=0.0
Cantidad_Nueva_Orleans_Denver=0.0
Cantidad_Nueva_Orleans_Miami=1200.0
Optimizacion: = 313200.0
```

Análisis de resultado

Para satisfacer la demanda de minimizando los costos de manera óptima, se usaron las siguientes asignaciones envíos de las plantas a los centros

- Detroit a Denver: 1300 unidades
- Detroit a Miami: 200 unidades
- Los Ángeles a Denver: 1000 unidades
- Los Ángeles a Miami: 0 unidades

- Nueva Orleans a Denver: 0 unidades
- Nueva Orleans a Miami: 1200 unidades

La función objetivo tiene un valor de $**313,200.0**, que representa el costo mínimo alcanzado en este problema de transporte

Problema 2.

Programe y resuelva la solución del problema de programación no lineal (5 Puntos)

1.- La función de beneficios de una empresa viene dada por la función:

$$B(x,y,z) = x\,y + 2\,z^2$$

donde x, y, z son las cantidades a producir de cada uno de los tres artículos que fabrica y vende.La empresa produce estos tres productos en un única sección en la que hay disponibles 120 horas semanales, empleando en la producción de una unidad del primer articulo 5 horas, en una del segundo 20 horas y en una del tercero 4 horas.

Se sabe además que por razones de demanda la empresa no puede producir menos de 5 unidades del primer articulo, ni más de 10 del segundo.

1. Determinar la producción a realizar.
2. Cuál debería ser la retribución de una hora extraordinaria?

Resultados

El código es el siguiente:

```
import numpy as np
from scipy.optimize import minimize
from scipy.optimize import NonlinearConstraint

# Función objetivo
def objetivo(x):
   return -x[0]*x[1]-2*pow(x[2],2)

# Condiciones iniciales (valores aproximados para empezar)
x0 = [30, 30, 30]  # Establecemos valores iniciales para x, y, z

rest1 = NonlinearConstraint(lambda x: -((x[0] * 5) + (x[1] * 20) + (x[2] * 4)),
-120, 0)
rest2 = NonlinearConstraint(lambda x: -x[0], -np.inf, -5)
rest3 = NonlinearConstraint(lambda x: -x[1], -10, 0)
```

```
rest4 = NonlinearConstraint(lambda x: -x[2], -np.inf, 0)

# Resolver el problema de optimización
resultado = minimize(objetivo, x0, constraints=[rest1, rest2, rest3, rest4])

# Mostrar los resultados
if resultado.success:
   print("Solución óptima encontrada:")
   print(f"x = {resultado.x[0]}, y = {resultado.x[1]}, z = {resultado.x[2]}")
   print("Solución con redondeo")
   print(f"x = {int(resultado.x[0])}, y = {int(resultado.x[1])}, z =
{int(resultado.x[2])}")
   print(f"Valor de la función objetivo: {-resultado.fun}")
else:
   print("No se encontró solución:", resultado.message)
```

1. **Determinar la producción a realizar**

```
Solución óptima encontrada:
x = 5.000000001462803, y = 1.5696066746851967e-09, z = 23.750000003313342
Solución con redondeo
x = 5, y = 0, z = 23
Valor de la función objetivo: 1128.1250003226155
```

2. **Cuál debería ser la retribución de una hora extraordinaria (Cuando son 121 horas)**

```
Solución óptima encontrada:
x = 5.000000000133461, y = 1.39046107960894e-10, z = 24.000000000293852
Solución con redondeo
x = 5, y = 0, z = 24
Valor de la función objetivo: 1152.000000028905
```

Análisis de los resultados

Cuando las horas disponibles son 120 y se quiere maximizar los beneficios, se debe producir 5 artículos “x” y 23 artículos “z”, de modo que el valor máximo es 1128.
Ahora cuando se le suma una hora a las horas disponibles, se puede producir un artículo “z” más, es decir, con 5 artículos “x” y 24 artículos “z” se alcanza el máximo, siendo este 1152

Problema 3. Programe y resuelva la solución del problema de pronósticos (5 Puntos)

Hospital Escazú proporciona servicios de laboratorio para pacientes de todos sus distritos, con un grupo de 10 médicos que atienden a familias con un nuevo programa de mantenimiento de la salud. Los administradores están interesados en pronosticar el número de análisis de sangre que se piden por semana. Una publicidad reciente acerca de los efectos dañinos del colesterol en el corazón ha ocasionado un incremento nacional en las peticiones de análisis de sangre.
Usar un promedio móvil simple de 3 meses, promedio ponderado de 0.6, 0.3 y 0.1 para los últimos 3 meses, y un modelo de suavización exponencial (α) de 0.4
El primer pronóstico para el modelo de suavización tiene un valor de 26.

A. Calcular: Error Medio Cuadrático (MSE), Error Absoluto Medio (MAE), Error Porcentual Medio (MAPE)
B. Elabore un gráfico con las proyecciones de cada método y las llegas reales.
C. ¿Cuál es el mejor método, explique sus resultados?

Semana	Llegadas	Semana	Llegadas
1	28	9	61
2	27	10	39
3	44	11	55
4	37	12	54
5	35	13	52
6	53	14	60
7	38	15	60
8	57	16	75

SOLUCIÓN EJERCICIO 3

El código fue programado en python y se coloca seguidamente.

```
# EJERCICIO 3
import numpy as np
import pandas as pd
import statsmodels.api as sm

# Datos de llegadas semanales
llegadas = np.array([28, 27, 44, 37, 35, 53, 38, 57, 61, 39,
55, 54, 52, 60, 60, 75])

# Número de semanas
n = len(llegadas)

pms = []  # Promedio Móvil Simple
pmp = []  # Promedio Móvil Ponderado
se = []   # Suavización Exponencial

# Promedio Móvil Simple (PMS)
for i in range(3, n):
     pms.append(np.mean(llegadas[i - 3:i]))

# Promedio Móvil Ponderado (PMP)
for i in range(3, n):
     pmp.append(0.6*llegadas[i - 1] + 0.3*llegadas[i - 2] +
0.1*llegadas[i - 3])

# Suavización Exponencial (SE)
alpha = 0.4
se.append(26)
for i in range(1, n):
     se.append(alpha*llegadas[i - 1] + (1 - alpha)*se[-1])

# Calcular errores
def calcular_errores(pronosticos, llegadas_reales, alineacion):
     errores = np.array(pronosticos) -
```

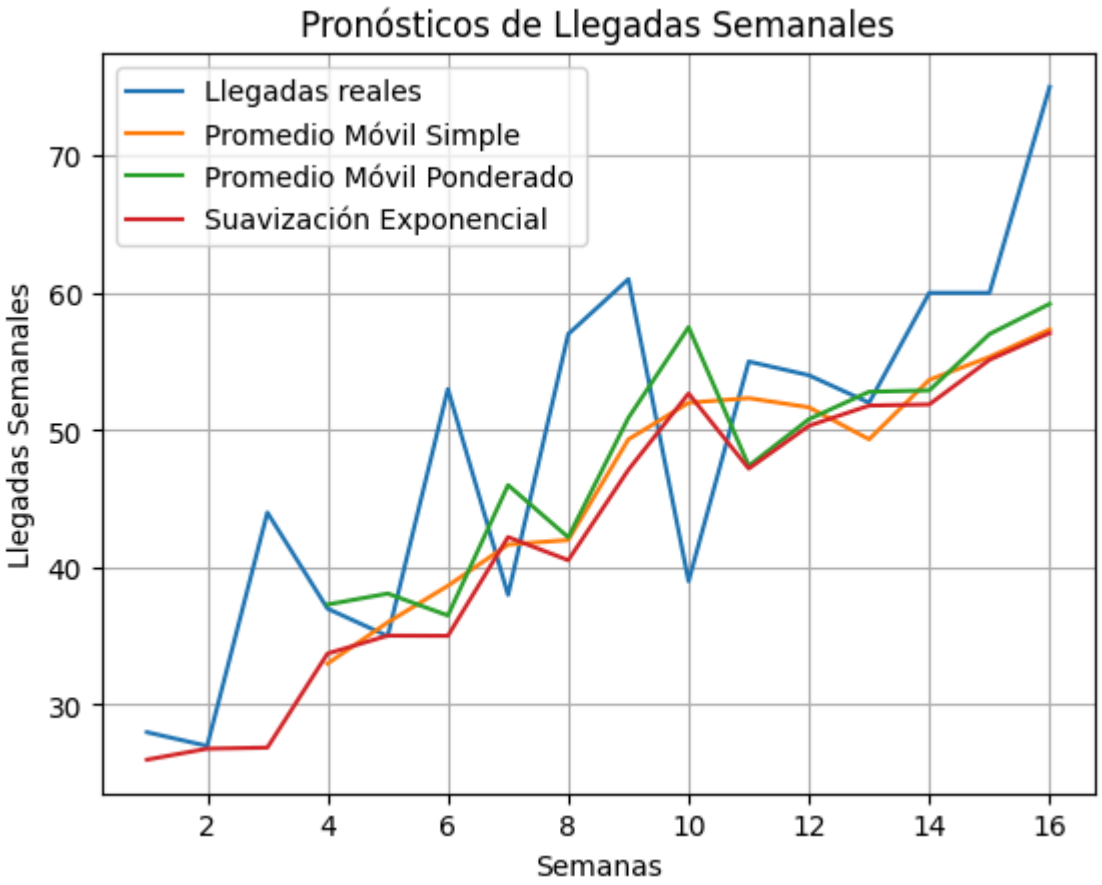
```
llegadas_reales[alineacion:]
     mse = np.mean(errores**2)
     mae = np.mean(np.abs(errores))
     mape = np.mean(np.abs(errores /
llegadas_reales[alineacion:]))*100
     return mse, mae, mape

# Calcular errores para cada método
mse_pms, mae_pms, mape_pms = calcular_errores(pms, llegadas, 3)
mse_pmp, mae_pmp, mape_pmp = calcular_errores(pmp, llegadas, 3)
mse_se, mae_se, mape_se = calcular_errores(se, llegadas, 0)

print("Resultados de Pronósticos:")
print(f"Promedio Móvil Simple: MSE={mse_pms:.2f},
MAE={mae_pms:.2f}, MAPE={mape_pms:.2f}%")
print(f"Promedio Móvil Ponderado: MSE={mse_pmp:.2f},
MAE={mae_pmp:.2f}, MAP={mape_pmp:.2f}%")
print(f"Suavización Exponencial: MSE={mse_se:.2f},
MAE={mae_se:.2f}, MAPE={mape_se: .2f}%")

# Presentación Ejercicio 3

import matplotlib.pyplot as plt

# Dimension X correponde a los semanales
x = np.arange(1, n + 1)

plt.plot(x, llegadas, label='Llegadas reales')
plt.plot(x[3:], pms, label='Promedio Móvil Simple')
plt.plot(x[3:], pmp, label='Promedio Móvil Ponderado')
plt.plot(x, se, label='Suavización Exponencial')

plt.xlabel('Semanas')
plt.ylabel('Llegadas semanales')
plt.title('Pronósticos de Llegadas Semanales')
plt.legend()
```

```
plt.grid(True)
plt.show()
```

RESPUESTA EJERCICIO 3 PREGUNTA 1

```
Resultados de Pronósticos:
Promedio Móvil Simple: MSE=89.21, MAE=7.62, MAPE=14.26%
Promedio Móvil Ponderado: MSE=106.69, MAE=8.37, MAP=16.23%
Suavización Exponencial: MSE=111.57, MAE=8.22, MAPE= 15.90%
```

RESPUESTA EJERCICIO 3 PREGUNTA 2



RESPUESTA EJERCICIO 3 PREGUNTA 3

Visualmente, los tres métodos parecen dar resultados muy parecidos. Además, se puede notar que fallan a la hora de predecir cambios muy drásticos. Sin embargo, el promedio móvil simple se destaca con los mejores índices de error (MSE, MAE y MAPE). Por esto, en este caso en particular, se afirma que el Promedio Móvil tuvo el mejor resultado.