Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Escuela Profesional de Ingeniería de Estadística e Informática

Optimization Methods

Métodos de Solución de Ecuaciones

Estudiante: Herson Romario Condori Mamani

Profesor: Fred Torres Cruz

20 de enero de 2025

Ejercicios

1. Supón que una compañía de reparto necesita determinar la combinación de camionetas (x) y motocicletas (y) para enviar paquetes dentro de la ciudad. El costo total (combustible, mantenimiento) está dado por la ecuación $3x + 2y = 26$, y la restricción de capacidad de envío por la ecuación $x + y = 10$. Utiliza la Regla de Cramer para encontrar la mejor combinación de vehículos.

2. Una agencia dispone de \$50 de presupuesto. El gasto en Facebook (x) y en Google Ads (y) debe satisfacer $x + y = 50$. Además, se conoce que cada dólar invertido en Facebook tiene un retorno doble al de Google Ads, por lo que se modela como $2x = y + 10$. Aplica la Regla de Cramer para determinar cuánto invertir en cada plataforma.

3. Una fábrica elabora tres productos: A, B y C. Cada uno requiere diferentes horas de mano de obra y materiales. Sea x, y, z la cantidad a producir de A, B y C, respectivamente. Las restricciones son:

$$\begin{aligned} 2x + y + z &= 40 \quad \text{(horas de mano de obra)} \\ x + 2y + 2z &= 50 \quad \text{(materia prima)} \\ 3x + 2y + z &= 60 \quad \text{(capacidad de máquina)} \end{aligned}$$

   Emplea la Regla de Cramer para determinar la producción de cada producto.

4. En una empresa de alimentos, se desea mezclar tres ingredientes (A, B, C) para producir un alimento balanceado. Las restricciones nutricionales se expresan en gramos totales de proteínas, carbohidratos y grasas. Sea x la cantidad de A, y la de B y z la de C (en kg), con el siguiente sistema:

$$\begin{aligned} 4x + 2y + z &= 30 \quad \text{(proteínas totales)} \\ x + 3y + 2z &= 28 \quad \text{(carbohidratos)} \\ 2x + y + 3z &= 33 \quad \text{(grasas)} \end{aligned}$$

   Aplica la Regla de Cramer y halla la solución de x, y, z.

```python
import streamlit as st
import numpy as np
from typing import List, Tuple
import sympy as sp

def solve_cramer(A: np.ndarray, b: np.ndarray) -> Tuple[List[float], str]:
n = len(A)
D = np.linalg.det(A)

if abs(D) < 1e-10:
```

```
return None, "El sistema no tiene soluci n  nica  (determinante = 0)"

x = []
steps = ["M todo de Cramer:\n"]
steps.append(f"Determinante principal = {D:.4f}")

for i in range(n):
Ai = A.copy()
Ai[:, i] = b
Di = np.linalg.det(Ai)
xi = Di / D
x.append(xi)
steps.append(f"x{i+1} = {Di:.4f}/{D:.4f} = {xi:.4f}")

return x, "\n".join(steps)

def solve_gauss_jordan(A: np.ndarray, b: np.ndarray) -> Tuple[List[float], str]:
n = len(A)
Ab = np.column_stack((A, b))
steps = ["M todo de Gauss-Jordan:\n"]

for i in range(n):
# Pivoteo
if abs(Ab[i][i]) < 1e-10:
for j in range(i+1, n):
if abs(Ab[j][i]) > 1e-10:
Ab[i], Ab[j] = Ab[j].copy(), Ab[i].copy()
steps.append(f"Intercambio R{i+1} <-> R{j+1}")
break
else:
return None, "El sistema no tiene soluci n  nica "

pivot = Ab[i][i]
Ab[i] = Ab[i] / pivot
steps.append(f"R{i+1} = R{i+1}/{pivot:.4f}")

for j in range(n):
if i != j:
factor = Ab[j][i]
Ab[j] = Ab[j] - factor * Ab[i]
steps.append(f"R{j+1} = R{j+1} - {factor:.4f}*R{i+1}")

x = Ab[:, -1]
return list(x), "\n".join(steps)

def solve_substitution(A: np.ndarray, b: np.ndarray) -> Tuple[List[float], str]:
n = len(A)
L = np.tril(A)
U = A - L
steps = ["M todo de Sustituci n:\n"]

if abs(np.linalg.det(L)) < 1e-10:
return None, "El sistema no puede resolverse por sustituci n"

```

```
y = np.zeros(n)
for i in range(n):
suma = sum(L[i][j] * y[j] for j in range(i))
y[i] = (b[i] - suma) / L[i][i]
steps.append(f"y{i+1} = ({b[i]:.4f} - {suma:.4f}) / {L[i][i]:.4f} =
     {y[i]:.4f}")

x = np.zeros(n)
for i in range(n-1, -1, -1):
suma = sum(U[i][j] * x[j] for j in range(i+1, n))
x[i] = y[i] - suma
steps.append(f"x{i+1} = {y[i]:.4f} - {suma:.4f} = {x[i]:.4f}")

return list(x), "\n".join(steps)

st.set_page_config(page_title="Solucionador de Sistemas de
    Ecuaciones", layout="wide")
st.title("Solucionador de Sistemas de Ecuaciones")

st.markdown("""
Esta aplicaci n resuelve sistemas de ecuaciones lineales
    utilizando diferentes m todos:
- M todo de Cramer
- M todo de Gauss-Jordan
- M todo de Sustituci n
""")
n = st.number_input("N mero de ecuaciones:", min_value=2,
    max_value=6, value=3)
col1, col2 = st.columns([3, 1])

with col1:
st.subheader("Coeficientes de las variables")
A = np.zeros((n, n))
for i in range(n):
cols = st.columns(n)
for j in range(n):
A[i][j] = cols[j].number_input(
f"a{i+1}{j+1}",
value=1.0 if i == j else 0.0,
key=f"A{i}{j}"
)

with col2:
st.subheader("T rminos independientes")
b = np.zeros(n)
for i in range(n):
b[i] = st.number_input(f"b{i+1}", value=1.0, key=f"b{i}")

if st.button("Resolver"):
results = {
        "Cramer": solve_cramer(A, b),
        "Gauss-Jordan": solve_gauss_jordan(A, b),
        "Sustituci n": solve_substitution(A, b)
}

for method, (solution, steps) in results.items():
with st.expander(f"Soluci n por {method}"):
```

```python
if solution is None:
st.error(steps)
else:
st.write("Soluci n:")
for i, xi in enumerate(solution):
st.write(f"x{i+1} = {xi:.4f}")
st.write("\nPasos:")
st.code(steps)

st.markdown("""
---
### Instrucciones:
1. Ingrese el n mero de ecuaciones que desea resolver
2. Complete los coeficientes de las variables (matriz A)
3. Complete los t rminos independientes (vector b)
4. Haga clic en "Resolver"
5. Expanda cada m todo para ver la soluci n detallada y los pasos

### Notas:
- El m todo de Cramer solo es eficiente para sistemas peque os
- El m todo de Gauss-Jordan es m s estable num ricamente
- El m todo de Sustituci n requiere que la matriz sea triangular
    inferior
""")
```

Listing 1: Código en Python para resolver sistemas de ecuaciones

References

[1] Código fuente en GitHub

[2] Aplicación en Streamlit