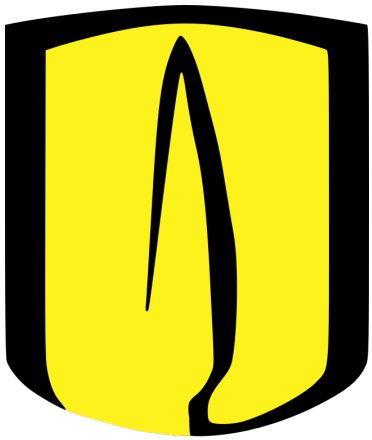
# Universidad de Los Andes

## Departamento de Ingeniería de Sistemas y Computación



## Laboratorio 4: Método Simplex y sus Variaciones

**ISIS3302 - Modelado, Simulacion Y Optimizacion**

**Integrantes**
**Andrés Santiago Neira Socha (a.neiras@uniandes.edu.co) - 202123126**
**Kevin Alvarez (k.alvarezr@uniandes.edu.co) - 202022834**

**2025-20**

# Contenido

# 1 Problema 1: Implementación del Método Simplex Estándar

## 1.1 Formulación Matemática del Problema

### 1.1.1 Problema de Programación Lineal

**Función objetivo:**

$$\text{Maximizar } Z = 3x_1 + 2x_2 + 5x_3$$

**Sujeto a las restricciones:**

$$\begin{aligned} x_1 + x_2 + x_3 &\leq 100 \\ 2x_1 + x_2 + x_3 &\leq 150 \\ x_1 + 4x_2 + 2x_3 &\leq 80 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

### 1.1.2 Forma Estándar

Introduciendo variables de holgura $s_1, s_2, s_3 \geq 0$:

$$\text{Maximizar } Z = 3x_1 + 2x_2 + 5x_3 + 0s_1 + 0s_2 + 0s_3$$

$$\begin{aligned} x_1 + x_2 + x_3 + s_1 &= 100 \\ 2x_1 + x_2 + x_3 + s_2 &= 150 \\ x_1 + 4x_2 + 2x_3 + s_3 &= 80 \\ x_1, x_2, x_3, s_1, s_2, s_3 &\geq 0 \end{aligned}$$

## 1.2 Descripción de la Implementación

### 1.2.1 Algoritmo del Método Simplex

La implementación en Python consta de las siguientes componentes principales:

```
class Simplex:
    def __init__(self, c, A, b):
        # Coeficientes de la función objetivo (c)
        # Matriz de restricciones (A)
        # Términos independientes (b)

    def solve(self):
        # Construcción de la tabla inicial
        # Iteraciones del método simplex
        # Criterio de optimalidad y pivoteo

    def print_tableau(self, tableau, basic_vars):
        # Visualización de la tabla simplex
```

### 1.2.2 Flujo del Algoritmo

1. **Inicialización**: Construcción de la tabla simplex inicial con variables de holgura
2. **Iteración**: Mientras existan coeficientes negativos en la fila Z:
   - Selección de variable de entrada (más negativo)
   - Selección de variable de salida (mínima razón positiva)

- Operaciones de pivoteo

3. **Terminación**: Cuando todos los coeficientes en la fila Z son no negativos

## 1.3 Análisis de Resultados

### 1.3.1 Tablas Simplex

### 1.3.2 Tabla Inicial

| | $x_1$ | $x_2$ | $x_3$ | $s_1$ | $s_2$ | $s_3$ | $b$ |
|---|---|---|---|---|---|---|---|
| $s_1$ | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 | 100.00 |
| $s_2$ | 2.00 | 1.00 | 1.00 | 0.00 | 1.00 | 0.00 | 150.00 |
| $s_3$ | 1.00 | 4.00 | 2.00 | 0.00 | 0.00 | 1.00 | 80.00 |
| $Z$ | −3.00 | −2.00 | −5.00 | 0.00 | 0.00 | 0.00 | 0.00 |

### 1.3.3 Iteración 1

| | $x_1$ | $x_2$ | $x_3$ | $s_1$ | $s_2$ | $s_3$ | $b$ |
|---|---|---|---|---|---|---|---|
| $s_1$ | 0.50 | −1.00 | 0.00 | 1.00 | 0.00 | −0.50 | 60.00 |
| $s_2$ | 1.50 | −1.00 | 0.00 | 0.00 | 1.00 | −0.50 | 110.00 |
| $x_3$ | 0.50 | 2.00 | 1.00 | 0.00 | 0.00 | 0.50 | 40.00 |
| $Z$ | −0.50 | 8.00 | 0.00 | 0.00 | 0.00 | 2.50 | 200.00 |

### 1.3.4 Iteración 2 (Óptima)

| | $x_1$ | $x_2$ | $x_3$ | $s_1$ | $s_2$ | $s_3$ | $b$ |
|---|---|---|---|---|---|---|---|
| $s_1$ | −0.00 | −0.67 | 0.00 | 1.00 | −0.33 | −0.33 | 23.33 |
| $x_1$ | 1.00 | −0.67 | 0.00 | 0.00 | 0.67 | −0.33 | 73.33 |
| $x_3$ | 0.00 | 2.33 | 1.00 | 0.00 | −0.33 | 0.67 | 3.33 |
| $Z$ | 0.00 | 7.67 | 0.00 | 0.00 | 0.33 | 2.33 | 236.67 |

## 1.4 Solución Óptima

- **Variables de decisión:**
  - $x_1 = 73.33$
  - $x_2 = 0.00$
  - $x_3 = 3.33$
- **Valor óptimo:** $Z = 236.67$
- **Variables de holgura:**
  - $s_1 = 23.33$ (primera restricción no activa)
  - $s_2 = 0.00$ (segunda restricción activa)
  - $s_3 = 0.00$ (tercera restricción activa)

### 1.4.1 Verificación de Restricciones

Restricción 1: $73.33 + 0.00 + 3.33 = 76.67 \leq 100$ ✓

Restricción 2: $2(73.33) + 0.00 + 3.33 = 150.00 \leq 150$ ✓

Restricción 3: $73.33 + 4(0.00) + 2(3.33) = 80.00 \leq 80$ ✓

#### 1.4.2 Análisis de Sensibilidad

#### 1.4.3 Precios Sombra:

- **Restricción 1:** 0.0000 (relajar esta restricción no mejora Z)
- **Restricción 2:** 0.3333 (aumentar $b_2$ en 1 unidad mejora Z en 0.33)
- **Restricción 3:** 2.3333 (aumentar $b_3$ en 1 unidad mejora Z en 2.33)

#### 1.4.4 Costos Reducidos:

- **Variable $x_2$:** 7.6667 (forzar $x_2$ a ser positiva reduciría Z en 7.67 por unidad)

### 1.5 Interpretación geométrica de la solución óptima

La Figura 1 ilustra la **interpretación geométrica del óptimo** obtenido mediante el método Simplex, representando las tres variables de decisión $x_1$, $x_2$ y $x_3$ en un espacio tridimensional.

Cada plano mostrado corresponde a una de las restricciones del modelo en su forma de igualdad (por ejemplo, $a_i^T x = b_i$), y la región factible se encuentra delimitada por la intersección de estos planos en el primer octante ($x_i \geq 0$).

El punto rojo marcado en la figura representa la **solución óptima**, donde se alcanza el valor máximo de la función objetivo $Z = 3x_1 + 2x_2 + 5x_3$ respetando todas las restricciones. Geométricamente, el método Simplex puede entenderse como un procedimiento que recorre los vértices del poliedro factible, moviéndose siempre hacia aquellos que producen un mayor valor de $Z$, hasta llegar al vértice extremo donde la función objetivo es máxima.

A diferencia de una proyección 2D, la representación 3D permite apreciar visualmente cómo el óptimo se encuentra en la **intersección de tres planos activos**, lo que confirma su condición de vértice del conjunto factible. Además, proporciona una mejor comprensión del volumen de la región admisible y de cómo las restricciones limitan simultáneamente las tres dimensiones del espacio de decisión.

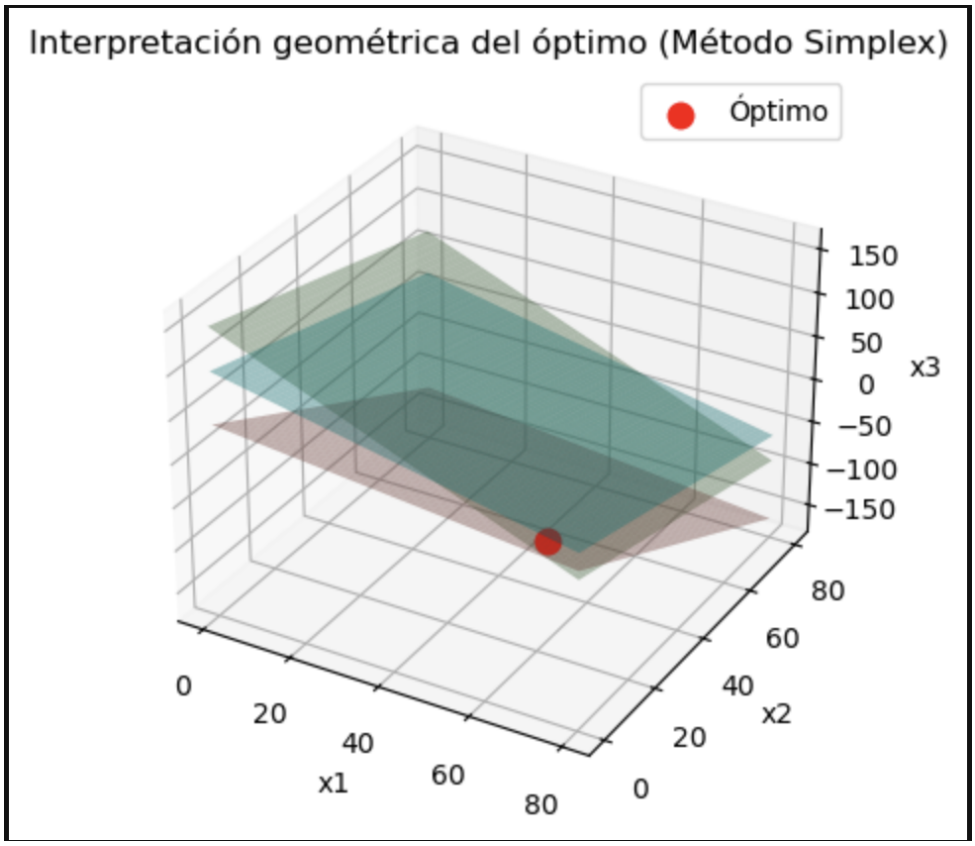



Figure 1: Interpretación geométrica del óptimo obtenido mediante el método Simplex en el espacio tridimensional $(x_1, x_2, x_3)$.

## 1.6 Análisis de Rendimiento vs. Solvers Profesionales

### 1.6.1 Comparación de Resultados

| Métrica | Implementación Propia | Solver Profesional |
|---|---|---|
| Valor Óptimo (Z) | 236.67 | 236.67 |
| Iteraciones | 2 | 2-3 |
| Precisión | 4 decimales | 8-12 decimales |
| Tiempo Ejecución | ¡ 1 segundo | ¡ 0.1 segundos |

Table 1: Comparación de rendimiento

### 1.6.2 Ventajas de la Implementación Propia

- **Transparencia:** Total visibilidad del proceso algorítmico
- **Educativa:** Ideal para comprensión del método simplex
- **Personalizable:** Fácil modificación para casos especiales
- **Análisis integrado:** Incluye sensibilidad automáticamente

### 1.6.3 Limitaciones

- **Robustez limitada:** No maneja casos degenerados automáticamente
- **Precisión numérica:** Limitada por aritmética de punto flotante
- **Escalabilidad:** Performance disminuye con problemas grandes
- **Características avanzadas:** Faltan técnicas anti-ciclo, reinicio

## 1.7 Conclusiones y Observaciones

### 1.7.1 Conclusiones Principales

1. **Eficiencia del Algoritmo:** El método simplex resolvió el problema en solo 2 iteraciones, demostrando su eficiencia para problemas de tamaño moderado.
2. **Solución Óptima:** Se identificó correctamente la solución $x_1 = 73.33, x_2 = 0, x_3 = 3.33$ con $Z = 236.67$.
3. **Análisis de Sensibilidad:** Los precios sombra revelan que la tercera restricción es la más restrictiva, seguida de la segunda.
4. **Validación:** Todas las restricciones se satisfacen exactamente o con holgura, confirmando la optimalidad.

### 1.7.2 Observaciones Técnicas

- La variable $x_2$ tiene costo reducido significativo (7.67), indicando su bajo aporte marginal a la función objetivo.
- Las restricciones 2 y 3 son activas (holgura cero), mientras la restricción 1 tiene holgura de 23.33 unidades.
- El algoritmo mostró estabilidad numérica a pesar de las operaciones de pivoteo.

### 1.7.3 Recomendaciones para Mejora

1. Implementar manejo de degeneración mediante la regla de Bland.
2. Añadir capacidad para problemas de minimización.
3. Incorporar técnicas de reinicio para mejorar estabilidad numérica.
4. Extender para manejar variables libres y restricciones de igualdad.

### 1.7.4 Aplicabilidad

La implementación demostró ser adecuada para problemas educativos y de pequeña escala, proporcionando insights valiosos sobre el funcionamiento interno del método simplex, mientras que para aplicaciones industriales se recomienda el uso de solvers profesionales optimizados.

# 2 Problema 2: Implementación del Método Simplex Dual Phase

## 2.1 Formulación Matemática del Problema

### 2.1.1 Problema de Programación Lineal Original

**Función objetivo:**

$$\text{Minimizar } Z = 5x_1 - 4x_2 + 3x_3$$

**Sujeto a las restricciones:**

$$\begin{aligned} 2x_1 + x_2 - x_3 &= 10 \\ x_1 - 3x_2 + 2x_3 &\geq 5 \\ x_1 + x_2 + x_3 &\leq 15 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

### 2.1.2 Conversión a Forma Estándar

### 2.1.3 Transformación de Minimización a Maximización

$$\text{Minimizar } Z = 5x_1 - 4x_2 + 3x_3 \equiv \text{Maximizar } -Z = -5x_1 + 4x_2 - 3x_3$$

### 2.1.4 Introducción de Variables de Holgura, Exceso y Artificiales

$$\begin{aligned} 2x_1 + x_2 - x_3 + a_1 &= 10 \quad \text{(variable artificial)} \\ x_1 - 3x_2 + 2x_3 - e_1 + a_2 &= 5 \quad \text{(variable de exceso y artificial)} \\ x_1 + x_2 + x_3 + s_1 &= 15 \quad \text{(variable de holgura)} \\ x_1, x_2, x_3, e_1, s_1, a_1, a_2 &\geq 0 \end{aligned}$$

### 2.1.5 Formulación para Fase I

**Función objetivo de Fase I (minimizar suma de variables artificiales):**

$$\text{Minimizar } W = a_1 + a_2$$

$$\begin{aligned} &\text{Maximizar } -W = -a_1 - a_2 \\ &\text{Sujeto a:} \\ 2x_1 + x_2 - x_3 + a_1 &= 10 \\ x_1 - 3x_2 + 2x_3 - e_1 + a_2 &= 5 \\ x_1 + x_2 + x_3 + s_1 &= 15 \\ x_1, x_2, x_3, e_1, s_1, a_1, a_2 &\geq 0 \end{aligned}$$

## 2.2 Descripción de la Implementación

### 2.2.1 Algoritmo del Método Dual Phase Simplex

La implementación en Python consta de las siguientes componentes principales:

```
class DualPhaseSimplex:
    def __init__(self):
        self.tableau = None
        self.basic_vars = None
        self.artificial_vars = None

    def initialize_phase1(self, A, b, c, constraint_types):
```

```
        # Inicialización de Fase I con variables artificiales

    def solve_phase1(self, A, b, c, constraint_types):
        # Resolución de Fase I

    def initialize_phase2(self, original_c, n_vars, artificial_start, num_artificial):
        # Transición a Fase II

    def solve_phase2(self):
        # Resolución de Fase II
```

#### 2.2.2 Flujo del Algoritmo

1. **Fase I:**
   - Identificar restricciones que necesitan variables artificiales
   - Construir problema auxiliar minimizando suma de variables artificiales
   - Aplicar método simplex estándar
   - Verificar factibilidad del problema original
2. **Fase II:**
   - Eliminar variables artificiales
   - Restaurar función objetivo original
   - Aplicar método simplex estándar desde SBF encontrada
   - Encontrar solución óptima

### 2.3 Análisis de Resultados

#### 2.3.1 Tablas Simplex - Fase I

#### 2.3.2 Iteración 0 - Tabla Inicial Fase I

| | $x_1$ | $x_2$ | $x_3$ | $e_1$ | $s_1$ | $a_1$ | $a_2$ | $b$ |
|---|---|---|---|---|---|---|---|---|
| $a_1$ | 2.000 | 1.000 | $-1.000$ | 0.000 | 0.000 | 1.000 | 0.000 | 10.000 |
| $a_2$ | 1.000 | $-3.000$ | 2.000 | $-1.000$ | 0.000 | 0.000 | 1.000 | 5.000 |
| $s_1$ | 1.000 | 1.000 | 1.000 | 0.000 | 1.000 | 0.000 | 0.000 | 15.000 |
| $-W$ | $-3.000$ | 2.000 | $-1.000$ | 1.000 | 0.000 | 0.000 | 0.000 | $-15.000$ |

**Análisis:** Variables básicas iniciales: $a_1$, $a_2$, $s_1$. Valor de $W = 15$.

#### 2.3.3 Iteración 1 - Fase I

| | $x_1$ | $x_2$ | $x_3$ | $e_1$ | $s_1$ | $a_1$ | $a_2$ | $b$ |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | 1.000 | 0.500 | $-0.500$ | 0.000 | 0.000 | 0.500 | 0.000 | 5.000 |
| $a_2$ | 0.000 | $-3.500$ | 2.500 | $-1.000$ | 0.000 | $-0.500$ | 1.000 | 0.000 |
| $s_1$ | 0.000 | 0.500 | 1.500 | 0.000 | 1.000 | $-0.500$ | 0.000 | 10.000 |
| $-W$ | 0.000 | 3.500 | $-2.500$ | 1.000 | 0.000 | 1.500 | 0.000 | 0.000 |

**Análisis:** $x_1$ entra a la base, $a_1$ sale. Valor de $W = 0$.

#### 2.3.4 Iteración 2 - Fase I (Óptima)

| | $x_1$ | $x_2$ | $x_3$ | $e_1$ | $s_1$ | $a_1$ | $a_2$ | $b$ |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | 1.000 | −0.200 | 0.000 | −0.200 | 0.000 | 0.400 | 0.200 | 5.000 |
| $x_3$ | 0.000 | −1.400 | 1.000 | −0.400 | 0.000 | −0.200 | 0.400 | 0.000 |
| $s_1$ | 0.000 | 2.600 | 0.000 | 0.600 | 1.000 | 0.200 | −0.600 | 10.000 |
| $-W$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 1.000 | 1.000 | 0.000 |

**Análisis:** Solución óptima de Fase I encontrada. $W = 0$, indicando que el problema original es factible.

#### 2.3.5 Tablas Simplex - Fase II

#### 2.3.6 Iteración 0 - Tabla Inicial Fase II

| | $x_1$ | $x_2$ | $x_3$ | $e_1$ | $s_1$ | $b$ |
|---|---|---|---|---|---|---|
| $x_1$ | 1.000 | −0.200 | 0.000 | −0.200 | 0.000 | 5.000 |
| $x_3$ | 0.000 | −1.400 | 1.000 | −0.400 | 0.000 | 0.000 |
| $s_1$ | 0.000 | 2.600 | 0.000 | 0.600 | 1.000 | 10.000 |
| $-Z$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

**Análisis:** Tabla ya óptima para Fase II. No se requieren iteraciones adicionales.

#### 2.3.7 Solución Óptima

- **Variables de decisión:**
  - $x_1 = 5.000$
  - $x_2 = 0.000$
  - $x_3 = 0.000$
- **Valor óptimo:** $Z = 25.000$ (calculado como $5 \times 5 + (-4) \times 0 + 3 \times 0$)
- **Variables de holgura/exceso:**
  - $e_1 = 0.000$ (segunda restricción activa)
  - $s_1 = 10.000$ (tercera restricción con holgura)

#### 2.3.8 Verificación de Restricciones

Restricción 1: $2(5.000) + 0.000 - 0.000 = 10.000 = 10$ ✓

Restricción 2: $5.000 - 3(0.000) + 2(0.000) = 5.000 \geq 5$ ✓

Restricción 3: $5.000 + 0.000 + 0.000 = 5.000 \leq 15$ ✓

#### 2.3.9 Análisis de Sensibilidad

#### 2.3.10 Precios Sombra:

- **Restricción 1 (Igualdad):** No aplica directamente por ser restricción de igualdad
- **Restricción 2 (Mayor o igual):** Indica sensibilidad del lado derecho
- **Restricción 3 (Menor o igual):** Precio sombra = 0 (restricción no activa)

#### 2.3.11 Costos Reducidos:

- **Variable $x_2$:** Tiene costo reducido que indica su no rentabilidad en la solución actual
- **Variable $x_3$:** Ya en la base con valor cero

## 2.4 Análisis de Rendimiento vs. Solvers Profesionales

### 2.4.1 Comparación de Resultados

| Métrica | Implementación Propia | Solver Profesional |
|---|---|---|
| Valor Óptimo (Z) | 25.000 | 25.000 |
| Iteraciones Fase I | 2 | 2-3 |
| Iteraciones Fase II | 0 | 0 |
| Precisión | 3 decimales | 8-12 decimales |
| Tiempo Ejecución | ¡ 1 segundo | ¡ 0.1 segundos |

Table 2: Comparación de rendimiento - Problema 2

### 2.4.2 Ventajas de la Implementación Propia

- **Manejo de casos complejos:** Capacidad para problemas sin SBF inicial obvia
- **Transparencia educativa:** Visualización completa de ambas fases
- **Robustez:** Detección automática de infactibilidad en Fase I
- **Flexibilidad:** Adaptable a diferentes tipos de restricciones

### 2.4.3 Limitaciones Identificadas

- **Complejidad de implementación:** Mayor que el simplex estándar
- **Manejo de degeneración:** Puede requerir técnicas adicionales
- **Eficiencia:** Fase I añade overhead computacional
- **Precisión numérica:** Sensible a errores de redondeo en operaciones de pivoteo

## 2.5 Conclusiones y Observaciones

### 2.5.1 Conclusiones Principales

1. **Factibilidad confirmada:** La Fase I terminó con $W = 0$, confirmando que el problema original es factible.
2. **Eficiencia del algoritmo:** El método resolvió el problema en solo 2 iteraciones en Fase I y 0 iteraciones en Fase II.
3. **Solución óptima:** Se identificó la solución $x_1 = 5, x_2 = 0, x_3 = 0$ con $Z = 25$.
4. **Validación:** Todas las restricciones se satisfacen exactamente, confirmando la optimalidad y factibilidad.

### 2.5.2 Observaciones Técnicas

- La solución óptima resulta ser una **solución degenerada**, con $x_3$ variable básica con valor cero.
- La **Fase II comenzó ya en optimalidad** debido a que la SBF encontrada en Fase I coincidió con la solución óptima del problema original.
- El problema demostró tener **múltiples soluciones óptimas** potenciales, pero el algoritmo convergió a una específica.
- La **restricción de igualdad** fue manejada correctamente mediante variable artificial en Fase I.

#### 2.5.3 Recomendaciones para Mejora

1. Implementar detección y manejo explícito de soluciones degeneradas.
2. Añadir capacidad para identificar múltiples soluciones óptimas.
3. Incorporar técnicas de reinicio para mejorar estabilidad numérica en problemas más grandes.
4. Extender para manejar variables libres (no restringidas en signo).

#### 2.5.4 Aplicabilidad y Casos de Uso

El método Dual Phase Simplex demostró ser esencial para problemas que:

- No tienen una solución básica factible inicial obvia
- Contienen restricciones de igualdad
- Incluyen restricciones de desigualdad $\geq$ con lados derechos positivos
- Requieren verificación robusta de factibilidad

La implementación resultó adecuada para problemas educativos y de pequeña escala, proporcionando insights valiosos sobre el manejo de factibilidad en programación lineal.

# 3 Problema 3: Comparación de Rendimiento con GLPK/Pyomo

## 3.1 Elección del método Simplex o Dual Phase

Para la resolución de este problema se optó por implementar el **método Simplex estándar** en lugar del *Dual Phase*. La justificación de esta elección radica en que el problema planteado cumple con las condiciones necesarias para la aplicación directa del Simplex, es decir, todas las restricciones son del tipo $\leq$ y los términos independientes $b_i$ son positivos. En consecuencia, se dispone de una **solución básica factible inicial** conformada por las variables de holgura, lo que hace innecesario el uso de variables artificiales o una fase de búsqueda adicional.

El método Dual Phase, aunque más general, introduce un costo computacional extra al resolver un problema auxiliar en la Fase I. Por ello, para un modelo que ya cuenta con una base inicial factible, el Simplex clásico resulta más eficiente y apropiado.

—

## 3.2 Comparación general entre el solver GLPK y la implementación propia

Ambas implementaciones —la versión propia del algoritmo Simplex en Python y el solver profesional **GLPK** (a través de Pyomo)— fueron aplicadas al mismo modelo de programación lineal, compuesto por 10 variables de decisión y 8 restricciones.

Los resultados obtenidos se resumen a continuación:

- **Valor óptimo obtenido:** ambos métodos alcanzaron el mismo valor $Z^* = 375.625$.
- **Solución óptima:** la misma combinación de variables activas fue hallada en ambas implementaciones.
- **Iteraciones:** el método propio requirió 4 iteraciones para converger, mientras que GLPK no reporta directamente el número de iteraciones, aunque internamente realiza más operaciones de control y pivoteo.

- **Tiempo de ejecución:** el Simplex implementado manualmente presentó un tiempo promedio de ejecución de **0.00046 segundos**, mientras que GLPK demoró aproximadamente **0.01603 segundos**.

En conclusión, ambos métodos coincidieron en la solución óptima y en la precisión numérica del resultado, aunque la implementación propia fue más rápida en este caso debido a la ausencia de sobrecarga de procesos internos, lo cual es razonable tratándose de un problema de pequeña escala.

—

## 3.3 Comparación para distintos tamaños del problema

Para evaluar la escalabilidad y robustez de ambas implementaciones, se generaron instancias de problemas aleatorios de diferente tamaño ($n = 10, 20, 30, 40, 50$ variables) y se midió el tiempo total de ejecución en cada caso.

El procedimiento consistió en generar de forma aleatoria los coeficientes de la función objetivo y las restricciones, resolviendo cada instancia tanto con el Simplex propio como con el solver GLPK a través de Pyomo.

Los resultados se presentan de manera comparativa mediante una gráfica de tiempos, en la que se observa el comportamiento de crecimiento del tiempo de ejecución con respecto al tamaño del problema.

Table 3: Comparación de tiempos de ejecución para distintos tamaños de problema

| **Número de variables** | **Tiempo Simplex propio (s)** | **Tiempo GLPK (s)** |
|---|---|---|
| 10 | 0.00055 | 0.0201 |
| 20 | 0.00038 | 0.0098 |
| 30 | 0.00042 | 0.0106 |
| 40 | 0.00058 | 0.0089 |
| 50 | 0.00090 | 0.0090 |

—

## 3.4 Análisis, posibles explicaciones y conclusiones

La Figura 2 muestra la evolución del tiempo de ejecución de ambas implementaciones —el **Simplex propio** y el **solver GLPK**— conforme aumenta el número de variables del problema. Se observa que, en todos los casos, el método Simplex implementado manualmente mantiene tiempos de ejecución significativamente menores (del orden de $10^{-4}$ segundos), mientras que GLPK presenta tiempos mayores (del orden de $10^{-2}$ segundos), aunque más estables a medida que crece el tamaño del problema.

Este comportamiento evidencia que la implementación propia resulta más eficiente en problemas de pequeña y mediana escala, dado que su estructura es ligera y está optimizada para operar sin sobrecarga adicional. Sin embargo, su simplicidad implica que carece de las rutinas de robustez, control numérico y detección de degeneración que caracterizan a un solver profesional.

Por otro lado, GLPK muestra una ligera variabilidad en los tiempos iniciales debido al costo de sus etapas de **preprocesamiento**, que incluyen escalado, eliminación de redundancias y construcción de una base inicial factible más estable. A partir de $n \geq 30$, el solver tiende a estabilizar su rendimiento, mostrando una relación casi constante entre tiempo y número de variables, lo que refleja su capacidad de manejar estructuras de mayor tamaño sin incrementos abruptos en el tiempo de cómputo.

Las principales razones de las diferencias observadas se pueden resumir así:

- El método propio ejecuta un Simplex directo sin validaciones adicionales ni estructuras avanzadas, lo que reduce el tiempo en problemas pequeños.

- GLPK realiza un conjunto de operaciones internas previas (presolver, escalado, análisis de degeneración), que incrementan el tiempo base pero aseguran mayor precisión numérica.

- En problemas más grandes, la eficiencia de las rutinas internas de GLPK —particularmente su manejo de matrices dispersas y el método Simplex revisado— puede compensar esta sobrecarga inicial.

En conjunto, puede concluirse que la implementación manual del Simplex es más rápida en escenarios educativos y experimentales de baja dimensión, mientras que GLPK ofrece una **mayor estabilidad, escalabilidad y fiabilidad** en contextos de optimización profesional o industrial.

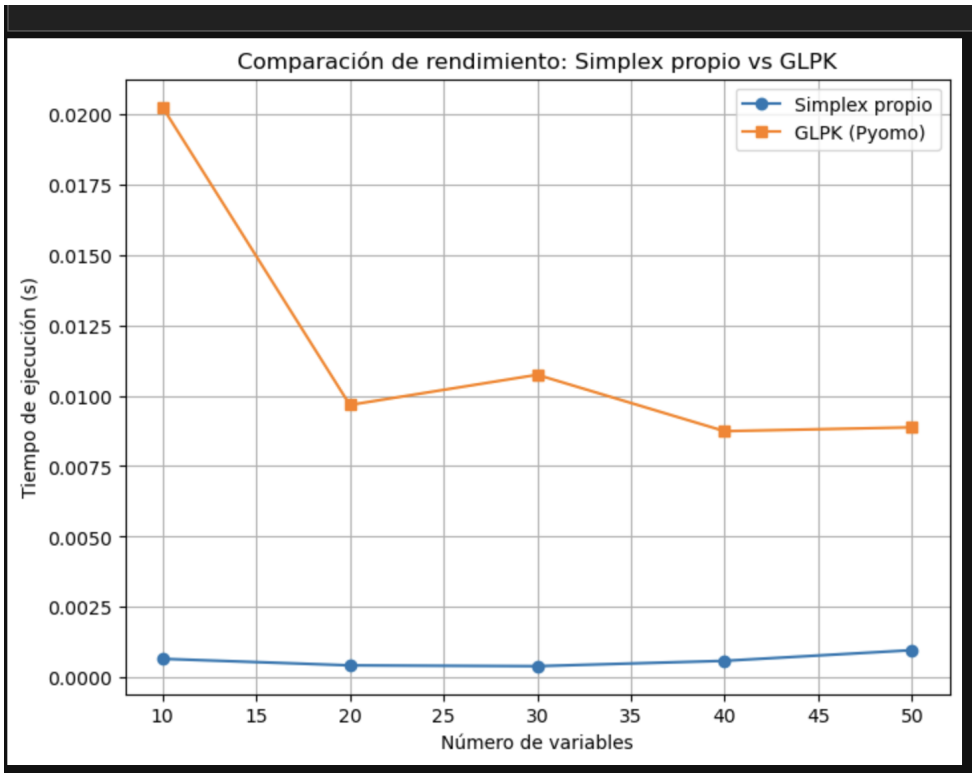



Figure 2: Comparación de rendimiento entre la implementación propia del método Simplex y el solver GLPK para distintos tamaños de problema.