Trabajo Completo: Graficador y Ejercicios Resueltos

Programación Numérica – FINESI

Universidad Nacional del Altiplano – Puno

Alumno: Wily Calib Caira Huancullo

Docente: Ing. Torres Cruz Fred

Índice general

Capítulo 1

Introducción

Este trabajo integra programación y análisis numérico para el estudio de funciones lineales y la representación de restricciones lineales. Se presentan dos herramientas desarrolladas en Python: (1) un *analizador de expresiones* que identifica variables y operaciones; (2) un *graficador de funciones lineales* que evalúa y grafica dos expresiones ingresadas por el usuario. Además, se resuelven cinco ejercicios aplicados, mostrando el razonamiento paso a paso y las gráficas correspondientes.

Capítulo 2

Metodología y pasos realizados

La metodología empleada para cada ejercicio y para la implementación de los programas fue:

1. Lectura del enunciado y definición de variables de decisión.
2. Formulación matemática (ecuaciones y/o desigualdades).
3. Cálculo algebraico de intersecciones y vértices candidatos.
4. Enumeración de combinaciones enteras factibles cuando corresponde.
5. Visualización: representación gráfica de las rectas y la región factible (con sombreado).
6. Interpretación de los resultados y conclusiones.

Capítulo 3

Analizador de funciones (resumen)

El analizador identifica letras como variables y cuenta operaciones explícitas (‘+ - * /
‘)$ymultiplicacionesimplcitas(p.ej.‘5x‘,‘xy‘).Suusoesauxiliarparavalidarexpresionesantesdegraficarl$

Capítulo 4

Código: Graficador de funciones lineales (versión final)

Aquí está el código exacto que solicitaste; guárdalo como `graficador_lineal_wily.py` y ejecútalo en tu PC (requiere `numpy` y `matplotlib`):

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

def evaluar(expr, x):
    """Eval a una expresi n matem tica en x."""
    try:
        return eval(expr, {"x": x, "np": np, "__builtins__": {}})
    except Exception:
        return None

def graficar(f1, f2, paso):
    """Grafica dos funciones lineales en el rango [-10, 10]."""
    xs = np.arange(-10, 10 + paso, paso)
    ys1 = [evaluar(f1, x) for x in xs]
    ys2 = [evaluar(f2, x) for x in xs]

    # Mostrar tabla de valores
    print("\nTABLA DE VALORES")
    print(f"{'x':>8} | {'f1(x)':>10} | {'f2(x)':>10}")
    print("-" * 35)
    for x, y1, y2 in zip(xs, ys1, ys2):
        print(f"{x:8.2f} | {y1:10.2f} | {y2:10.2f}")

    # Crear la gr fica
    plt.figure(figsize=(8, 6))
    plt.plot(xs, ys1, 'r-', label=f"f1(x) = {f1}")
    plt.plot(xs, ys2, 'b--', label=f"f2(x) = {f2}")
    plt.axhline(0, color='black', linewidth=1)
    plt.axvline(0, color='black', linewidth=1)
    plt.title("Gr fico de Funciones Lineales")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.legend()
    plt.grid(True)
    plt.show()

# Programa principal
if __name__ == "__main__":
```

```
39      print("=== GRAFICADOR DE FUNCIONES LINEALES ===")
40      f1 = input("Ingrese la funci n 1 (ejemplo: 2*x + 1): ")
41      f2 = input("Ingrese la funci n 2 (ejemplo: -x + 3): ")
42      paso = float(input("Ingrese el paso (ejemplo: 0.5): "))
43      graficar(f1, f2, paso)
```

Capítulo 5

Ejemplo breve (comprobación)

Para comprobar, use:

$$f_1(x) = 2x + 1, \qquad f_2(x) = -x + 3, \qquad \text{paso} = 1.$$

La tabla de valores y la gráfica se muestran a continuación (la gráfica está reproducida en LaTeX mediante `pgfplots`):

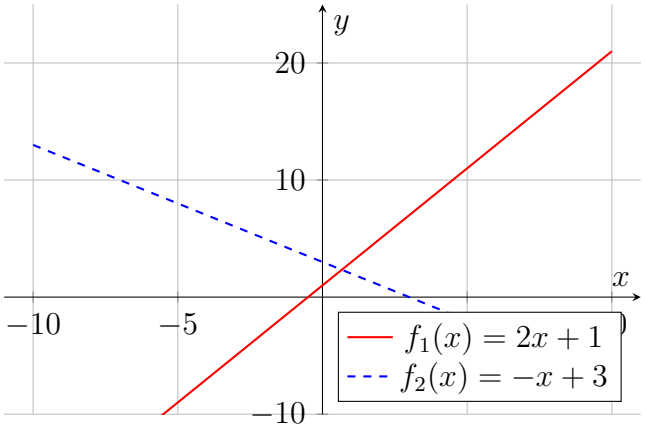

Figura 5.1: Ejemplo de comprobación: f_1 y f_2.

Capítulo 6

Ejercicios resueltos: enunciados y pasos

Ejercicio 1: Tiempo de desarrollo

Enunciado. Un desarrollador dispone de 15 horas semanales para tareas: desarrollo (variable x) y diseño (variable y). Debe dedicar al menos 5 horas al desarrollo.

Paso 1 — Variables: x = horas de desarrollo; y = horas de diseño. **Paso 2 — Restricciones:**

$$x \geq 5, \qquad x + y \leq 15, \qquad x \geq 0,\ y \geq 0.$$

Paso 3 — Intersecciones y vértices: La frontera principal es la recta $x + y = 15$. Los vértices candidatos en el primer cuadrante y cumpliendo $x \geq 5$ son:

$$(5, 10), \quad (15, 0), \quad (5, 0) \text{ (candidato de esquina)}.$$

Paso 4 — Combinaciones enteras (si se requiere): Enumerar (x, y) enteros con $x \geq 5$ y $x + y \leq 15$. Ejemplo: $(5, 0), (5, 1), \ldots, (5, 10), (6, 0), \ldots, (15, 0)$. **Paso 5 — Interpretación:** El programador debe dedicar al menos 5 horas; la región factible muestra todas las distribuciones válidas. La elección óptima depende de criterios adicionales (p. ej. maximizar output del desarrollo).

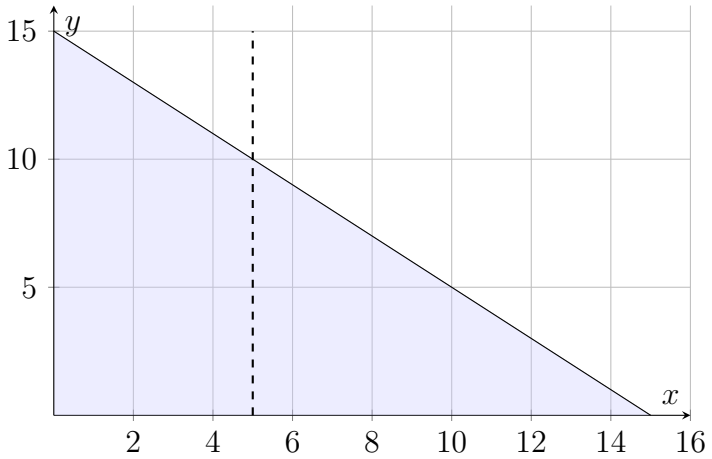

Figura 6.1: Región factible del ejercicio 1.

Ejercicio 2: Servidores en la nube

Enunciado. Cada servidor (x) cuesta 3 unidades, cada unidad de almacenamiento (y) cuesta 5; presupuesto máximo 20.

Paso 1 — Variables: x = cantidad servidores, y = cantidad almacenamiento. **Paso 2 — Restricción:** $3x + 5y \leq 20$ (si se acepta menor o igual) o $= 20$ si el presupuesto se consume exactamente. **Paso 3 — Intersecciones:** Si $3x + 5y = 20$, al despejar $y = (20 - 3x)/5$. Intersecciones con ejes: $(0, 4)$ y $(20/3, 0) \approx (6{,}67, 0)$. **Paso 4 — Combinaciones enteras:** Buscar parejas (x, y) enteras no negativas con $3x + 5y \leq 20$. Por ejemplo: $(0, 0), (0, 1), (1, 3)$ no cumple; cálculo sistemático en el código entregado. **Paso 5 — Interpretación:** La recta muestra combinaciones que usan todo el presupuesto; el área por debajo serían combinaciones con menor gasto.

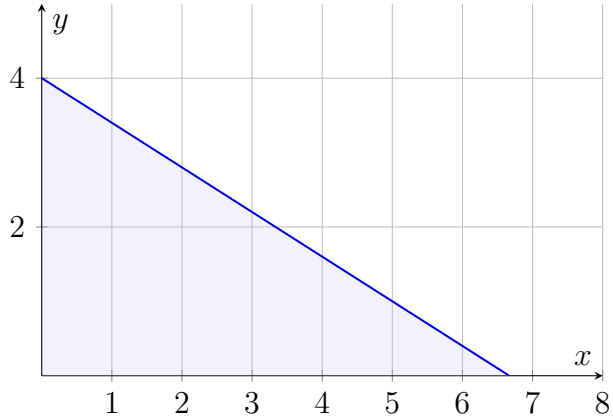

Figura 6.2: Recta presupuesto y área factible.

Ejercicio 3: Organización del tiempo

Enunciado. El estudiante dedica tiempo a estudio (x) y descanso (y). Debe cumplir: $x \geq 4$, $y \geq 6$, $x + y \leq 12$.

Paso 1 — Variables: x horas de estudio, y horas de descanso. **Paso 2 — Restricciones:** $x \geq 4,\ y \geq 6,\ x + y \leq 12$. **Paso 3 — Intersecciones:** Intersección de $x + y = 12$ con $x = 4$ da $(4, 8)$; con $y = 6$ da $(6, 6)$. Vértices candidatos: $(4, 8), (6, 6), (4, 6)$ y esquinas del dominio. **Paso 4 — Combinaciones enteras:** Enumerar pares enteros en el polígono convexo formado. **Paso 5 — Interpretación:** Región factible limitada y pequeña; muestra opciones de equilibrio entre estudio y descanso.

Ejercicio 4: Producción de assets

Enunciado. Dos productos: P1 requiere 2h/unidad (x), P2 requiere 3h/unidad (y). Tiempo total disponible: 18 h.

Paso 1 — Variables: x = unidades P1, y = unidades P2. **Paso 2 — Restricción:** $2x + 3y \leq 18$. Intersecciones: con eje x en $(9, 0)$ y con eje y en $(0, 6)$. **Paso 3 — Vértices**

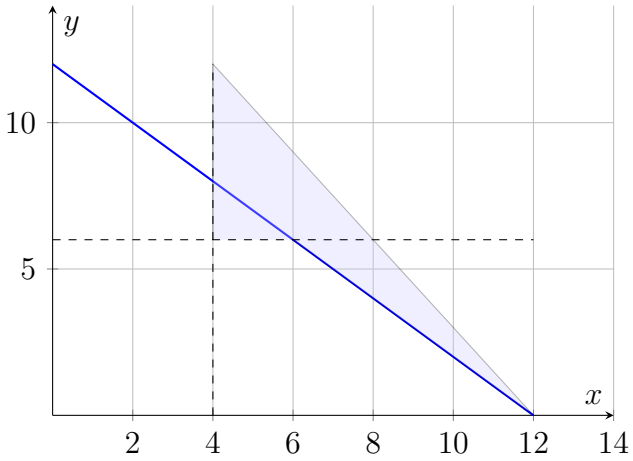

Figura 6.3: Región factible del ejercicio 3.

candidatos: $(0,0), (9,0), (0,6)$ y puntos en la frontera. **Paso 4 — Combinaciones enteras:** Enumerar pares enteros (x, y) con $2x + 3y \leq 18$. Ejemplo: $(0,0), (1,0), (0,1), (3,4)$ verificarían o no según la desigualdad. **Paso 5 — Interpretación:** Permite planificar producción sin exceder el tiempo total.

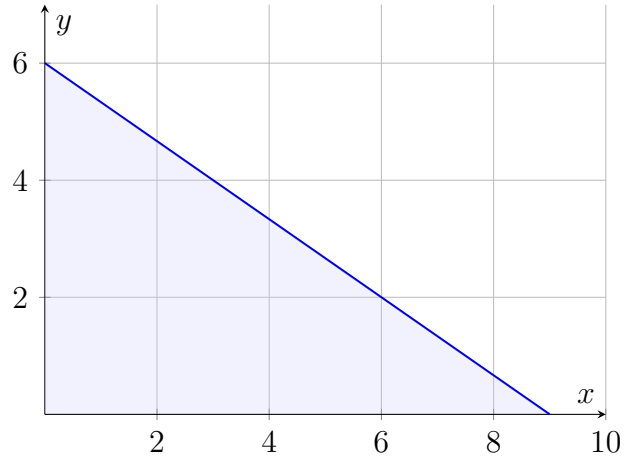

Figura 6.4: Región factible del ejercicio 4.

Ejercicio 5: Componentes para ensamblaje

Enunciado. Dos tipos de componentes requieren recursos: A consume 5 unidades, B consume 10 unidades; máximo 50 unidades disponibles. Encuentre combinaciones factibles.

Paso 1 — Variables: x = unidades A, y = unidades B. **Paso 2 — Restricción:** $5x + 10y \leq 50 \Rightarrow$ simplificable a $x + 2y \leq 10$. Intersecciones: $(10,0)$ y $(0,5)$. **Paso 3 — Vértices:** $(0,0), (10,0), (0,5)$ y puntos en la frontera. **Paso 4 — Combinaciones enteras:** Buscar pares enteros (x, y) con $x + 2y \leq 10$ (p. ej. $(0,0), (0,1), (1,0), (8,1)$ etc.). **Paso 5 — Interpretación:** Restricción lineal simple que facilita planificación de stock para ensamblaje.

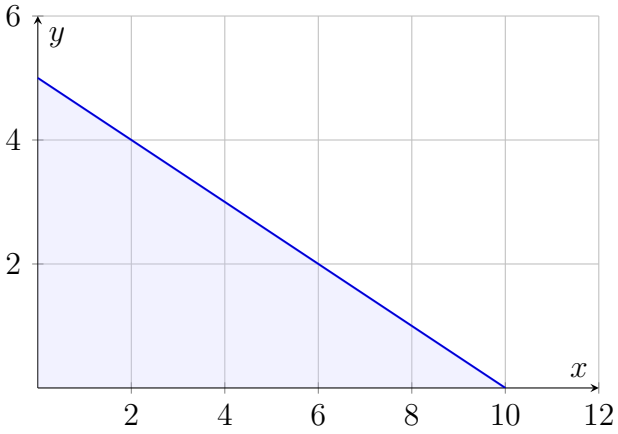

Figura 6.5: Región factible del ejercicio 5.

Capítulo 7

Conclusión

Se presentó una solución completa que integra la implementación en Python y la resolución analítica de problemas lineales. Los pasos documentados —identificación de variables, modelación, cálculo de intersecciones, enumeración de combinaciones enteras y análisis gráfico— permiten reproducir y adaptar las soluciones según criterios adicionales (optimización, costos, prioridad).