Restricciones y Sistemas de Ecuaciones

HENRRY HIGINIO QUISPE RAMOS

1 Introducción a las Restricciones

Las restricciones son condiciones que limitan el conjunto de valores posibles que pueden tomar las variables en un problema. Estas son esenciales en problemas de optimización y en sistemas de ecuaciones, ya que definen la *región factible*, es decir, el espacio de soluciones posibles.

1.1 Tipos de Restricciones

- **Restricciones de Igualdad**: Representan condiciones exactas. Por ejemplo:

$$x + y = 10 \tag{1}$$

  donde la suma de dos variables debe ser exactamente 10.

- **Restricciones de Desigualdad**: Permiten l´ımites superiores o inferiores. Por ejemplo:

$$x + y \leq 10 \tag{2}$$

  donde la suma de dos variables no debe exceder 10.

- **Restricciones de Límites**: Definen rangos para las variables. Por ejemplo:

$$0 \leq x \leq 5 \tag{3}$$

  donde la variable x debe estar entre 0 y 5.

2 Sistemas de Ecuaciones

Un sistema de ecuaciones es un conjunto de ecuaciones que comparten las mismas variables. Resolver un sistema consiste en encontrar los valores de las variables que satisfacen todas las ecuaciones simultáneamente.

2.1 Ejemplo Básico: Resolución por Sustitución

Supongamos el siguiente sistema:

$$x + y = 10, \tag{4}$$

$$x - y = 4. \tag{5}$$

Paso 1: Resolver una ecuación para una variable. De la ecuación (4), despejamos x:

$$x = 10 - y. \tag{6}$$

Paso 2: Sustituir en la segunda ecuación. Sustituimos $x = 10 - y$ en (5):

$$(10 - y) - y = 4, \tag{7}$$
$$10 - 2y = 4, \tag{8}$$
$$-2y = -6, \tag{9}$$
$$y = 3. \tag{10}$$

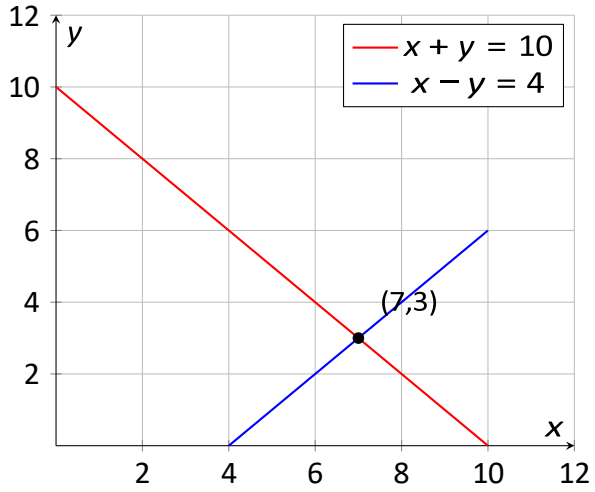
Paso 3: Sustituir el valor de y en la primera ecuación. Sustituimos $y = 3$ en (4):

$$x + 3 = 10, \tag{11}$$
$$x = 7. \tag{12}$$

Solución: $x = 7$, $y = 3$.

2.2 Representación Gráfica

A continuación, graficamos ambas ecuaciones para visualizar la solución:



3 Ejemplo Práctico: Tiempo de Estudio

Un estudiante dispone de 10 horas al d´ıa para estudiar Matemáticas (x) y Programación (y). Además:

- Debe dedicar al menos 2 horas a Matemáticas:

$$x \geq 2. \tag{13}$$

- Debe dedicar al menos 3 horas a Programación:

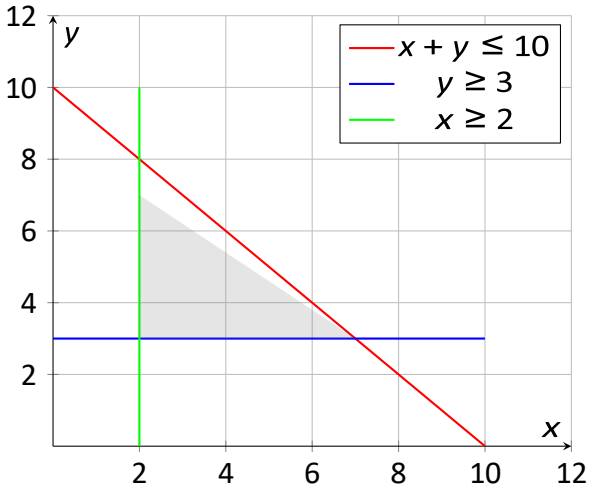
$$y \geq 3. \tag{14}$$

- El tiempo total no puede exceder 10 horas:

$$x + y \leq 10. \tag{15}$$

3.1 Representación Gráfica

La región factible se puede visualizar en el siguiente gráfico:



Conclusión: La región factible está limitada por las restricciones y cualquier solución debe pertenecer a esta región.

CODIGO GRAFICADOR

```
def extraer_coeficientes(expr):
    expr = expr.replace(" ", "").lower()
    if 'x' not in expr:
        raise ValueError("La expresión debe contener 'x'")

    x_index = expr.find('x')
    m = expr[:x_index]
    b = expr[x_index+1:] if x_index+1 < len(expr) else '0'

    if m == '':
        m = '1'
    elif m == '+':
        m = '1'
    elif m == '-':
        m = '-1'

    if b == '':
        b = '0'
    elif b.startswith('+') or b.startswith('-'):
        pass
    else:
```

```
        b = '+' + b

    return float(m), float(b)

def pedir_funciones_texto():
    funciones = []
    for i in range(2):
        texto = input(f"Ingrese la función {i+1} (ej: 2x+3 o -x-4): ")
        try:
            funciones.append(extraer_coeficientes(texto))
        except Exception as e:
            print(f"Error: {e}")
            return pedir_funciones_texto()
    return funciones

def graficar_dos_funciones(funciones):
    print("\nGráfico (1 = función 1, 2 = función 2):\n")
    for y in range(10, -11, -1):
        linea = ""
        for x in range(-20, 21):
            simbolo = ' '
            if x == 0 and y == 0:
                simbolo = '+'
            elif x == 0:
                simbolo = '|'
            elif y == 0:
                simbolo = '-'
            for idx, (m, b) in enumerate(funciones):
                yf = m * x + b
                if int(round(yf)) == y:
                    simbolo = str(idx+1)
            linea += simbolo
        print(linea)

if __name__ == "__main__":
    funciones = pedir_funciones_texto()
```

graficar_dos_funciones(funciones)

4 Ejercicios Propuestos

1. Un desarrollador tiene 15 horas semanales para dedicar al desarrollo de software de front-end (x) y back-end (y). Además:

   - Debe dedicar al menos 5 horas al desarrollo de front-end para cumplir con los entregables del cliente.
   - El tiempo total no puede exceder 15 horas por restricciones de tiempo del sprint.

   Formule las restricciones, represéntelas gráficamente e identifique las combinaciones posibles de tiempo a invertir en cada actividad.

CODIGO :

```
def extraer_coeficientes(expr):
    expr = expr.replace(" ", "").lower()
    if 'x' not in expr:
        return 0.0, float(expr)
    if expr.startswith('x'): expr = '1' + expr
    if expr.startswith('-x'): expr = '-1' + expr
    x_index = expr.find('x')
    m_str = expr[:x_index]
    b_str = expr[x_index+1:] if x_index+1 < len(expr) else '0'
    m = float(m_str)
    b = float(b_str) if b_str else 0.0
    return m, b


def pedir_funciones_texto():
    funciones = []
    print("Ejercicio 1: Desarrollador de Software")
    print("Ingrese dos funciones en formato y = mx + b (ej: -x+10)")
    for i in range(2):
        while True:
            texto = input(f"Ingrese la función {i+1}: ")
            try:
```

```
            funciones.append(extraer_coeficientes(texto))
            break
        except Exception as e:
            print(f"Error en el formato: {e}. Inténtelo de nuevo.")
    return funciones

def cumple_restricciones_ej1(x, y):
    return (x + y <= 15) and (x >= 5) and (y >= 0)

def graficar_combinado(funciones):
    print("\n" + "="*60)
    print("Visualización Combinada: Funciones (1, 2) y Región Factible (*)")
    print("="*60 + "\n")
    for y in range(15, -3, -1):
        linea = f"{y:3d}| "
        for x in range(-2, 21):
            simbolo = ' '
            if cumple_restricciones_ej1(x, y):
                simbolo = '*'
            for idx, (m, b) in enumerate(funciones):
                if abs((m * x + b) - y) < 0.5:
                    simbolo = str(idx + 1)
            if x == 0 and y == 0:
                simbolo = '+'
            elif x == 0 and simbolo == ' ':
                simbolo = '|'
            elif y == 0 and simbolo == ' ':
                simbolo = '-'
            linea += simbolo
        print(linea)
    print("   +" + "-" * 23)
```

```
if __name__ == "__main__":
    funciones_usuario = pedir_funciones_texto()
    graficar_combinado(funciones_usuario)
```

```
============================================================

 15|    |     1       2
 14|    |    1       2
 13|    |   1       2
 12|    |  1       2
 11|    | 1       2
 10|    |1       2
  9|    1     * 2
  8|   1|     *2
  7| 1  |     2**
  6|    |    2****
  5|    |   2 *****
  4|    |  2  ******
  3|    | 2   *******
  2|    |2    ********
  1|    2     *********
  0|   2|     **********
 -1|  2 |     **********
 -2| --+----***********-----
   |
   |
   +----------------------
```

Un ingeniero de datos administra dos tipos de servidores en la nube: Servidores A y Servidores B. El costo por hora de Servidor A es S/3 y de Servidor B es S/5. El presupuesto máximo semanal asignado para mantener los servidores es de S/20. Determine cuántas horas puede mantener activos cada tipo de servidor, formule el sistema de ecuaciones y represéntelo gráficamente.

CODIGO:

```
def extraer_coeficientes(expr):
    expr = expr.replace(" ", "").lower()
    if 'x' not in expr:
        return 0.0, float(expr)
    if expr.startswith('x'): expr = '1' + expr
    if expr.startswith('-x'): expr = '-1' + expr
    x_index = expr.find('x')
    m_str = expr[:x_index]
    b_str = expr[x_index+1:] if x_index+1 < len(expr) else '0'
    m = float(m_str)
    b = float(b_str) if b_str else 0.0
```

```
    return m, b

def pedir_funciones_texto():
    funciones = []
    print("Ejercicio 2: Ingeniero de Datos")
    print("Ingrese dos funciones en formato y = mx + b (ej: -0.6x+4)")
    for i in range(2):
        while True:
            texto = input(f"Ingrese la función {i+1}: ")
            try:
                funciones.append(extraer_coeficientes(texto))
                break
            except Exception as e:
                print(f"Error en el formato: {e}. Inténtelo de nuevo.")
    return funciones

def cumple_restricciones_ej2(x, y):
    return (3*x + 5*y <= 20) and (x >= 0) and (y >= 0)

def graficar_combinado(funciones):
    print("\n" + "="*60)
    print("Visualización Combinada: Funciones (1, 2) y Región Factible (*)")
    print("="*60 + "\n")
    for y in range(10, -3, -1):
        linea = f"{y:3d}| "
        for x in range(-2, 15):
            simbolo = ' '
            if cumple_restricciones_ej2(x, y):
                simbolo = '*'
            for idx, (m, b) in enumerate(funciones):
                if abs((m * x + b) - y) < 0.5:
```

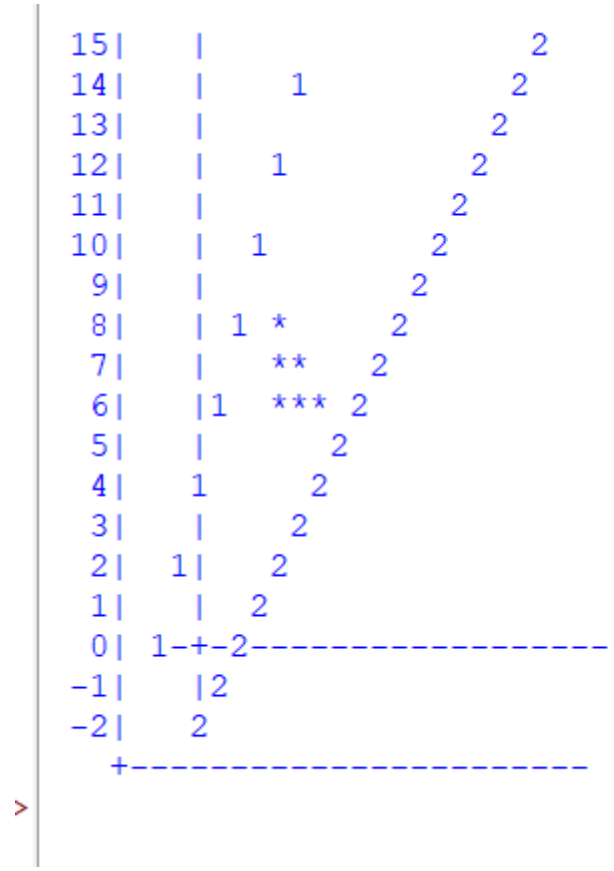
```
                simbolo = str(idx + 1)
            if x == 0 and y == 0:
                simbolo = '+'
            elif x == 0 and simbolo == ' ':
                simbolo = '|'
            elif y == 0 and simbolo == ' ':
                simbolo = '-'
            linea += simbolo
        print(linea)
    print("   +" + "-" * 17)


if __name__ == "__main__":
    funciones_usuario = pedir_funciones_texto()
    graficar_combinado(funciones_usuario)
```

```
 10|    |       21
  9|    |      21
  8|    |     21
  7|    |    21
  6|    |   21
  5|    |  21
  4|    *21
  3|    21
  2|   21***
  1|  21******
  0|  1-+******--------
 -1|    |
 -2|    |
    +-----------------
```

2. Un administrador de proyectos tecnológicos organiza su tiempo entre reuniones con stakeholders (*x*) y trabajo en la documentación técnica (*y*). Las reuniones requieren al menos 4 horas semanales y la documentación al menos 6 horas. Si dispone de 12 horas para ambas actividades, determine la región factible y analice las combinaciones posibles de tiempo.

```
def extraer_coeficientes(expr):
    expr = expr.replace(" ", "").lower()
    if 'x' not in expr:
        return 0.0, float(expr)
    if expr.startswith('x'): expr = '1' + expr
    if expr.startswith('-x'): expr = '-1' + expr
    x_index = expr.find('x')
    m_str = expr[:x_index]
    b_str = expr[x_index+1:] if x_index+1 < len(expr) else '0'
    m = float(m_str)
    b = float(b_str) if b_str else 0.0
    return m, b


def pedir_funciones_texto():
    funciones = []
    print("Ejercicio 3: Administrador de Proyectos")
    print("Ingrese dos funciones en formato y = mx + b (ej: -x+12)")
    for i in range(2):
        while True:
            texto = input(f"Ingrese la función {i+1}: ")
            try:
                funciones.append(extraer_coeficientes(texto))
                break
            except Exception as e:
                print(f"Error en el formato: {e}. Inténtelo de nuevo.")
    return funciones


def cumple_restricciones_ej3(x, y):
```

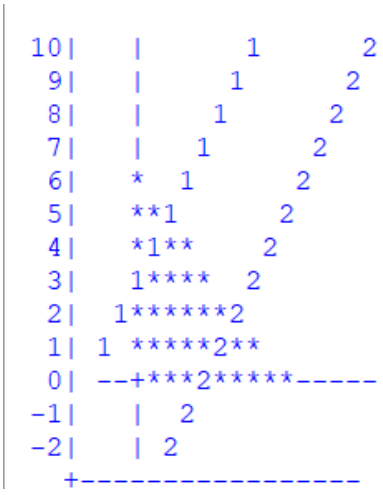
```
    return (x + y <= 12) and (x >= 4) and (y >= 6)


def graficar_combinado(funciones):
    print("\n" + "="*60)
    print("Visualización Combinada: Funciones (1, 2) y Región Factible (*)")
    print("="*60 + "\n")
    for y in range(15, -3, -1):
        linea = f"{y:3d}| "
        for x in range(-2, 21):
            simbolo = ' '
            if cumple_restricciones_ej3(x, y):
                simbolo = '*'
            for idx, (m, b) in enumerate(funciones):
                if abs((m * x + b) - y) < 0.5:
                    simbolo = str(idx + 1)
            if x == 0 and y == 0:
                simbolo = '+'
            elif x == 0 and simbolo == ' ':
                simbolo = '|'
            elif y == 0 and simbolo == ' ':
                simbolo = '-'
            linea += simbolo
        print(linea)
    print("   +" + "-" * 23)


if __name__ == "__main__":
    funciones_usuario = pedir_funciones_texto()
    graficar_combinado(funciones_usuario)
```


15| | 2
14| | 1   2
13| | 2
12| | 1   2
11| | 2
10| | 1   2
 9| | 2
 8| | 1 *   2
 7| | **   2
 6| |1   *** 2
 5| | 2
 4| 1   2
 3| | 2
 2| 1| 2
 1| | 2
 0| 1-+-2------------------
-1| |2
-2| 2
   +-----------------------

3. Una empresa de desarrollo de videojuegos produce dos tipos de assets: Modelos 3D (P1) y Texturas (P2). Cada modelo 3D requiere 2 horas de trabajo y cada textura requiere 3 horas. El equipo de arte tiene un total de 18 horas disponibles semanalmente. Formule las restricciones, represéntelas gráficamente y determine cuántos assets de cada tipo pueden producirse en función del tiempo disponible.

```
def extraer_coeficientes(expr):
    expr = expr.replace(" ", "").lower()
    if 'x' not in expr:
        return 0.0, float(expr)
    if expr.startswith('x'): expr = '1' + expr
    if expr.startswith('-x'): expr = '-1' + expr
    x_index = expr.find('x')
    m_str = expr[:x_index]
    b_str = expr[x_index+1:] if x_index+1 < len(expr) else '0'
    m = float(m_str)
    b = float(b_str) if b_str else 0.0
    return m, b


def pedir_funciones_texto():
    funciones = []
    print("Ejercicio 4: Empresa de Videojuegos")
    print("Ingrese dos funciones en formato y = mx + b (ej: -0.66x+6)")
    for i in range(2):
        while True:
            texto = input(f"Ingrese la función {i+1}: ")
            try:
                funciones.append(extraer_coeficientes(texto))
                break
            except Exception as e:
                print(f"Error en el formato: {e}. Inténtelo de nuevo.")
    return funciones


def cumple_restricciones_ej4(x, y):
```

```
    return (2*x + 3*y <= 18) and (x >= 0) and (y >= 0)


def graficar_combinado(funciones):
    print("\n" + "="*60)
    print("Visualización Combinada: Funciones (1, 2) y Región Factible (*)")
    print("="*60 + "\n")
    for y in range(10, -3, -1):
        linea = f"{y:3d}| "
        for x in range(-2, 15):
            simbolo = ' '
            if cumple_restricciones_ej4(x, y):
                simbolo = '*'
            for idx, (m, b) in enumerate(funciones):
                if abs((m * x + b) - y) < 0.5:
                    simbolo = str(idx + 1)
            if x == 0 and y == 0:
                simbolo = '+'
            elif x == 0 and simbolo == ' ':
                simbolo = '|'
            elif y == 0 and simbolo == ' ':
                simbolo = '-'
            linea += simbolo
        print(linea)
    print("   +" + "-" * 17)


if __name__ == "__main__":
    funciones_usuario = pedir_funciones_texto()
    graficar_combinado(funciones_usuario)
if __name__ == "__main__":
    funciones_usuario = pedir_funciones_texto()
        graficar_combinado(funciones_usuario)
```


10|
9|
8|
7|
6|
5|
4|
3|
2|
1|
0|
-1|
-2|

4. Una startup de hardware dispone de un máximo de 50 unidades de componentes electrónicos. Para ensamblar un dispositivo tipo A se necesitan 5 unidades y para un dispositivo tipo B se necesitan 10 unidades. Determine cuántos dispositivos de cada tipo puede ensamblar sin exceder las 50 unidades de componentes. Formule el problema, resuélvalo gráficamente y explique las posibles combinaciones de producción.

```
def extraer_coeficientes(expr):
    expr = expr.replace(" ", "").lower()
    if 'x' not in expr:
        return 0.0, float(expr)
    if expr.startswith('x'): expr = '1' + expr
    if expr.startswith('-x'): expr = '-1' + expr
    x_index = expr.find('x')
    m_str = expr[:x_index]
    b_str = expr[x_index+1:] if x_index+1 < len(expr) else '0'
    m = float(m_str)
    b = float(b_str) if b_str else 0.0
    return m, b

def pedir_funciones_texto():
    funciones = []
    print("Ejercicio 5: Startup de Hardware")
    print("Ingrese dos funciones en formato y = mx + b (ej: -0.5x+5)")
    for i in range(2):
        while True:
            texto = input(f"Ingrese la función {i+1}: ")
            try:
                funciones.append(extraer_coeficientes(texto))
                break
            except Exception as e:
                print(f"Error en el formato: {e}. Inténtelo de nuevo.")
    return funciones

def cumple_restricciones_ej5(x, y):
    return (5*x + 10*y <= 50) and (x >= 0) and (y >= 0)

def graficar_combinado(funciones):
    print("\n" + "="*60)
    print("Visualización Combinada: Funciones (1, 2) y Región Factible (*)")
    print("="*60 + "\n")
    for y in range(10, -3, -1):
        linea = f"{y:3d}| "
        for x in range(-2, 15):
            simbolo = ' '
            if cumple_restricciones_ej5(x, y):
```

```
            simbolo = '*'
        for idx, (m, b) in enumerate(funciones):
            if abs((m * x + b) - y) < 0.5:
                simbolo = str(idx + 1)
        if x == 0 and y == 0:
            simbolo = '+'
        elif x == 0 and simbolo == ' ':
            simbolo = '|'
        elif y == 0 and simbolo == ' ':
            simbolo = '-'
        linea += simbolo
    print(linea)
print("   +" + "-" * 17)

if __name__ == "__main__":
    funciones_usuario = pedir_funciones_texto()
    graficar_combinado(funciones_usuario)
```

```
 10|    |   1           2
  9|    |  1          2
  8|    | 1          2
  7|    1           2
  6|   1*         2
  5|  1 **       2
  4|    ****    2
  3|    ***** 2
  2|    *****2*
  1|    ****2***
  0|  --+**2******-----
 -1|    |  2
 -2|    | 2
     +-----------------
```