Gráfica de Funciones y Ejercicios Resueltos

Alumno:
Ronald Wilder Incacutipa Muñuico

Profesor:
Ing. Fred Torres Cruz

Clase: 25-09-27
25 de septiembre de 2025

Código básico para graficar funciones lineales

```python
import math

def evaluar(expr, x):
    try:
        return eval(expr, {"x": x, "math": math})
    except:
        return None

def graficar(f1, f2):
    xs = list(range(-10, 11))   # valores de x de -10 a 10
    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(" x | f1(x) | f2(x)")
    print("-"*20)
    for x, y1, y2 in zip(xs, ys1, ys2):
        print(f"{x:2} | {y1:6} | {y2:6}")

    # --- Grafico en consola ---
    vals = [v for v in ys1+ys2 if v is not None]
    ymin, ymax = min(vals), max(vals)

    alto = 20
    ancho = len(xs)

    def fila(y):
        return int((ymax-y)*(alto-1)/(ymax-ymin))

    grid = [[" "]*ancho for _ in range(alto)]

    # Ejes
    if ymin <= 0 <= ymax:
        r0 = fila(0)
        for c in range(ancho): grid[r0][c] = "-"
    if 0 in xs:
        c0 = xs.index(0)
        for r in range(alto): grid[r][c0] = "|"

    # Funciones
    for i,(y1,y2) in enumerate(zip(ys1,ys2)):
        if y1 is not None: grid[fila(y1)][i] = "x"
        if y2 is not None: grid[fila(y2)][i] = "o"
        if y1 is not None and y2 is not None and fila(y1)==fila(y2):
            grid[fila(y1)][i] = "*"

    print("\nGrafico:")
```

```
    for fila_ in grid:
        print("".join(fila_))

# --- Programa principal ---
f1 = input("Funcion 1: ")
f2 = input("Funcion 2: ")
graficar(f1,f2)
```

Código adaptado para resolver los ejercicios

```
import numpy as np
import matplotlib.pyplot as plt
from itertools import combinations

def graficar_restricciones(restricciones, limites=(0, 20, 0, 20))
    :
    """
    restricciones: lista de tuplas (a, b, c, tipo)
        a*x + b*y <= c    ---> tipo = "<="
        a*x + b*y >= c    ---> tipo = ">="
    limites: (xmin, xmax, ymin, ymax)
    """
    xmin, xmax, ymin, ymax = limites
    x = np.linspace(xmin, xmax, 400)

    plt.figure(figsize=(7, 7))

    # Dibujar las rectas de las restricciones
    for a, b, c, tipo in restricciones:
        if b != 0:
            y = (c - a * x) / b
            plt.plot(x, y, label=f"{a}x + {b}y {tipo} {c}")
        else:
            x_line = np.full_like(x, c / a)
            plt.plot(x_line, x, label=f"{a}x {tipo} {c}")

    # Ejes
    plt.axhline(0, color="black", linewidth=0.8)
    plt.axvline(0, color="black", linewidth=0.8)

    # ---------------------------
    # Encontrar puntos de interseccion
    # ---------------------------
    puntos = []
    for (a1, b1, c1, _), (a2, b2, c2, _) in combinations(
       restricciones, 2):
        det = a1 * b2 - a2 * b1
        if det != 0:  # hay interseccion
            x_int = (c1 * b2 - c2 * b1) / det
            y_int = (a1 * c2 - a2 * c1) / det
```

```
            if xmin <= x_int <= xmax and ymin <= y_int <= ymax:
                puntos.append((x_int, y_int))

    # Agregar esquinas del grafico
    puntos += [(xmin, ymin), (xmin, ymax), (xmax, ymin), (xmax, ymax)]

    # Filtrar puntos factibles
    factibles = []
    for px, py in puntos:
        valido = True
        for a, b, c, tipo in restricciones:
            if tipo == "<=" and not (a * px + b * py <= c + 1e-6):
                valido = False
            if tipo == ">=" and not (a * px + b * py >= c - 1e-6):
                valido = False
        if valido:
            factibles.append((px, py))

    # Region factible
    if factibles:
        factibles = np.array(factibles)
        from scipy.spatial import ConvexHull
        hull = ConvexHull(factibles)
        pts = factibles[hull.vertices]
        plt.fill(pts[:, 0], pts[:, 1],
                 "skyblue", alpha=0.4,
                 label="Region factible")

    # Configuracion final
    plt.xlim(xmin, xmax)
    plt.ylim(ymin, ymax)
    plt.xlabel("x")
    plt.ylabel("y")
    plt.legend()
    plt.grid(True, linestyle="--", alpha=0.6)
    plt.title("Restricciones y Region Factible")
    plt.show()

# Aqui van las restricciones dependiendo del problema
```

Ejercicios resueltos

Ejercicio 1. (Tiempo de desarrollo)
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 del sprint.

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

Restricciones (formulación)

Variables:

$$x = \text{horas en front-end}, \quad y = \text{horas en back-end}$$

Restricciones:

$$x \geq 5, \quad x + y \leq 15, \quad x \geq 0, \; y \geq 0$$

Combinaciones posibles (horas enteras)

Para horas enteras, x puede tomar valores $5, 6, \ldots, 15$. Para cada x, y puede ser cualquier entero desde 0 hasta $15 - x$.

Las combinaciones (x, y) factibles son:

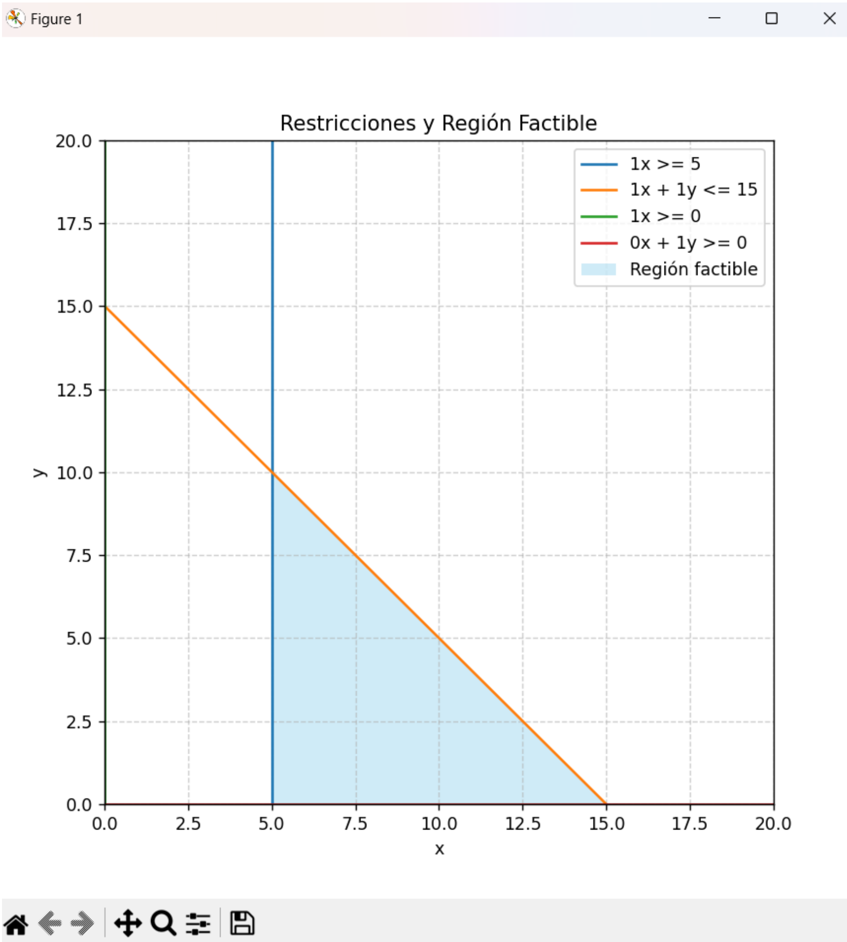
$$\begin{aligned}
x = 5 &: (5,0),(5,1),(5,2),(5,3),(5,4),(5,5),(5,6),(5,7),(5,8),(5,9),(5,10) \quad (11)\\
x = 6 &: (6,0),(6,1),(6,2),(6,3),(6,4),(6,5),(6,6),(6,7),(6,8),(6,9) \quad (10)\\
x = 7 &: (7,0),(7,1),(7,2),(7,3),(7,4),(7,5),(7,6),(7,7),(7,8) \quad (9)\\
x = 8 &: (8,0),(8,1),(8,2),(8,3),(8,4),(8,5),(8,6),(8,7) \quad (8)\\
x = 9 &: (9,0),(9,1),(9,2),(9,3),(9,4),(9,5),(9,6) \quad (7)\\
x = 10 &: (10,0),(10,1),(10,2),(10,3),(10,4),(10,5) \quad (6)\\
x = 11 &: (11,0),(11,1),(11,2),(11,3),(11,4) \quad (5)\\
x = 12 &: (12,0),(12,1),(12,2),(12,3) \quad (4)\\
x = 13 &: (13,0),(13,1),(13,2) \quad (3)\\
x = 14 &: (14,0),(14,1) \quad (2)\\
x = 15 &: (15,0) \quad (1)
\end{aligned}$$

Total de combinaciones enteras factibles: $11 + 10 + \cdots + 1 = 66$. **Nota:** Agregar este código en la parte final del código adaptado para los ejercicios:

```
restricciones = [
    (1, 0, 5, ">="),     # x >= 5
    (1, 1, 15, "<="),    # x + y <= 15
    (1, 0, 0, ">="),     # x >= 0
    (0, 1, 0, ">=")      # y >= 0
]
```

```
graficar_restricciones(restricciones, limites=(0, 20, 0, 20))
```

Representación gráfica del problema



Ejercicio 2. (Servidores en la nube)
Un ingeniero de datos administra dos tipos de servidores en la nube: *Servidor A* (x) y *Servidor B* (y). 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.

Sistema de ecuaciones

$$3x + 5y = 20$$

Variables:

$$x = \text{horas de Servidor A}, \quad y = \text{horas de Servidor B}$$

Solución (puntos de intersección)

Para encontrar las combinaciones posibles:
- Si $x = 0$:

$$5y = 20 \quad \Rightarrow \quad y = 4$$

- Si $y = 0$:

$$3x = 20 \quad \Rightarrow \quad x = \frac{20}{3} \approx 6{,}67$$

Por lo tanto, la recta $3x + 5y = 20$ pasa por los puntos $(0, 4)$ y $\left(\frac{20}{3}, 0\right)$.

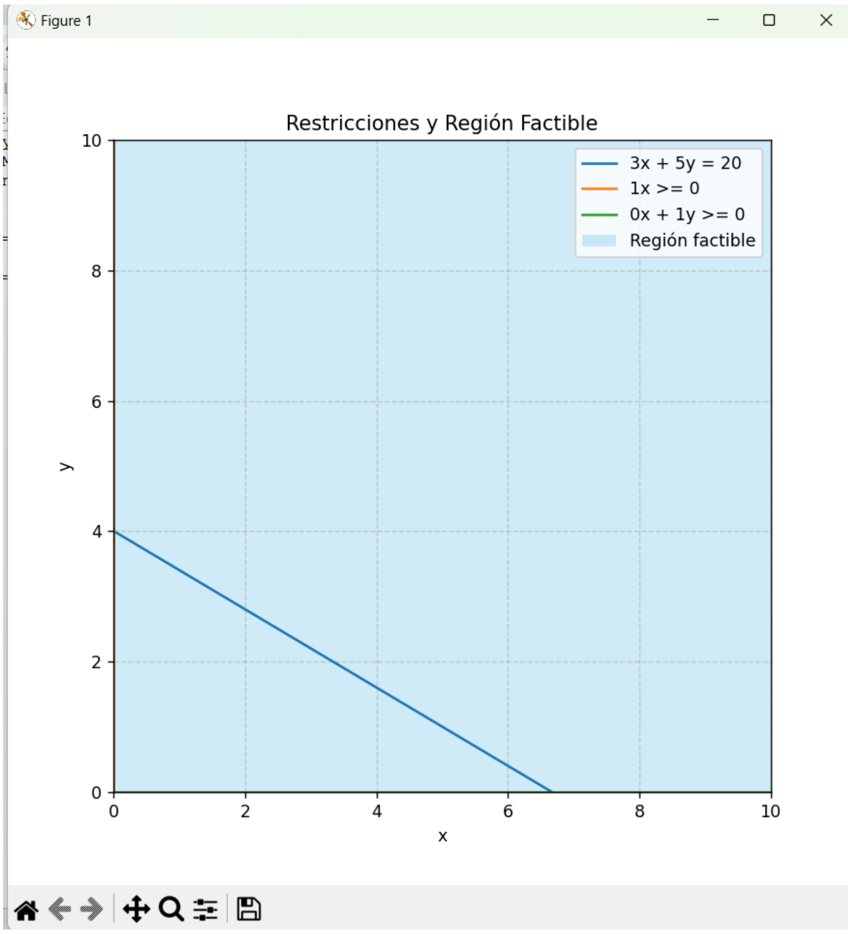
Código en Python

Nota: Agregar este código en la parte final del código adaptado para los ejercicios:

```python
restricciones = [
    (3, 5, 20, "="),     # 3x + 5y = 20
    (1, 0, 0, ">="),     # x >= 0
    (0, 1, 0, ">=")      # y >= 0
]

graficar_restricciones(restricciones, limites=(0, 10, 0, 10))
```

Representación gráfica del problema



Ejercicio 3. (Organización de tiempo)
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.

Restricciones (formulación)

Variables:

$$x = \text{horas en reuniones}, \quad y = \text{horas en documentación}$$

Restricciones:

$$x \geq 4, \quad y \geq 6, \quad x + y \leq 12, \quad x \geq 0,\ y \geq 0$$

Región factible (interpretación)

La región factible está definida por las tres condiciones principales:

- $x \geq 4$ (línea vertical en $x = 4$, tomar la parte hacia la derecha),
- $y \geq 6$ (línea horizontal en $y = 6$, tomar la parte hacia arriba),
- $x + y \leq 12$ (línea recta $y = 12 - x$, tomar la parte debajo de la recta).

Geométricamente la región factible es el polígono (en este caso un triángulo o trapecio reducido) limitado por las intersecciones de esas rectas en el primer cuadrante.

Vértices importantes (intersecciones):

$$(4, 6), \quad (4, 8) \text{ (intersección de } x = 4 \text{ con } y = 12 - x), \quad (6, 6) \text{ (intersección de } y = 6 \text{ con } x + y = 12).$$

Combinaciones posibles (horas enteras)

Como las horas se pueden considerar enteras, x puede tomar los valores $4, 5, 6$ (porque $x \geq 4$ y $x + y \leq 12$ con $y \geq 6$ limita x a como máximo 6). Para cada x:

$$\begin{aligned} x = 4 &: (4,6), (4,7), (4,8) \quad (3) \\ x = 5 &: (5,6), (5,7) \quad (2) \\ x = 6 &: (6,6) \quad (1) \end{aligned}$$

Total de combinaciones enteras factibles: $3 + 2 + 1 = 6$.

Código en Python

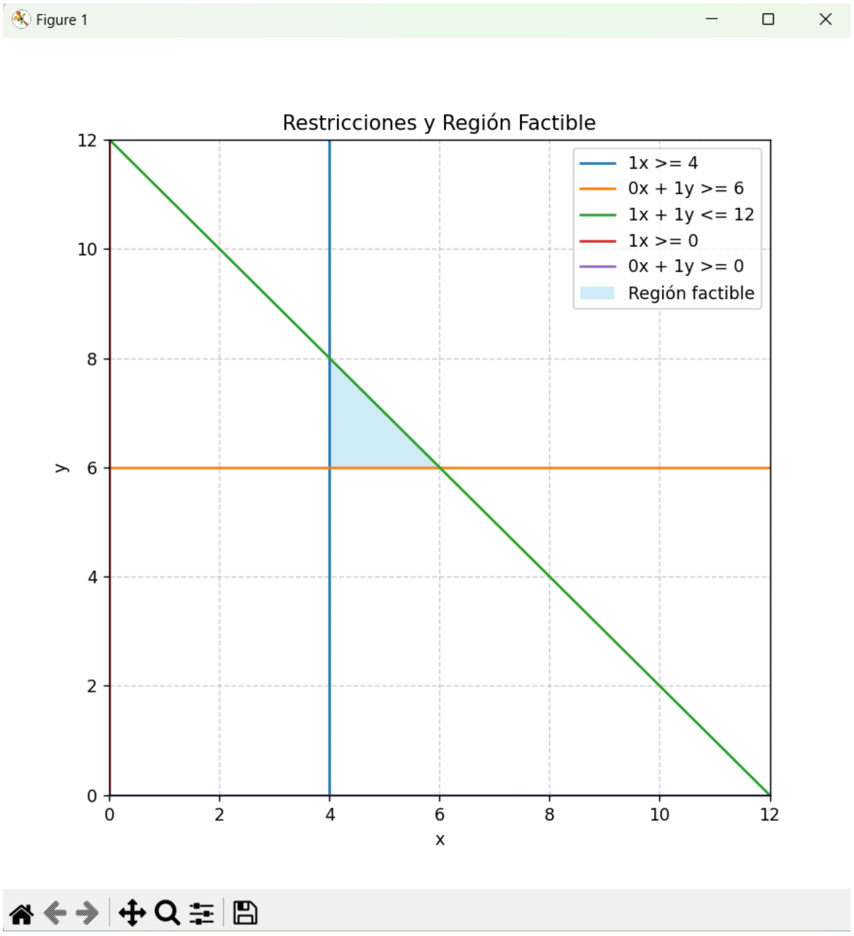
Nota: Agregar este código en la parte final del código adaptado para los ejercicios:

```
restricciones = [
    (1, 0, 4, ">="),     # x >= 4
    (0, 1, 6, ">="),     # y >= 6
    (1, 1, 12, "<="),    # x + y <= 12
    (1, 0, 0, ">="),     # x >= 0
```

```
    (0, 1, 0, ">=")      # y >= 0
]

graficar_restricciones(restricciones, limites=(0, 12, 0, 12))
```

Representación gráfica del problema



Ejercicio 4. (Producción de assets)
Una empresa de desarrollo de videojuegos produce dos tipos de assets: Modelos 3D (P_1) y Texturas (P_2). 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.

Restricciones (formulación)

Variables:

$$P_1 = \text{cantidad de Modelos 3D}, \quad P_2 = \text{cantidad de Texturas}$$

Restricciones:

$$2P_1 + 3P_2 \leq 18, \quad P_1 \geq 0, \; P_2 \geq 0$$

Combinaciones posibles (valores enteros)

Para valores enteros, P_1 puede tomar valores $0, 1, \ldots, 9$ (porque $2P_1 \leq 18 \Rightarrow P_1 \leq 9$). Para cada P_1 entero, P_2 puede ser cualquier entero desde 0 hasta $\left\lfloor \frac{18 - 2P_1}{3} \right\rfloor$.

Las combinaciones (P_1, P_2) factibles (agrupadas por P_1) son:

$$
\begin{aligned}
P_1 &= 0 : (0,0), (0,1), (0,2), (0,3), (0,4), (0,5), (0,6) \quad (7) \\
P_1 &= 1 : (1,0), (1,1), (1,2), (1,3), (1,4), (1,5) \quad (6) \\
P_1 &= 2 : (2,0), (2,1), (2,2), (2,3), (2,4) \quad (5) \\
P_1 &= 3 : (3,0), (3,1), (3,2), (3,3), (3,4) \quad (5) \\
P_1 &= 4 : (4,0), (4,1), (4,2), (4,3) \quad (4) \\
P_1 &= 5 : (5,0), (5,1), (5,2) \quad (3) \\
P_1 &= 6 : (6,0), (6,1), (6,2) \quad (3) \\
P_1 &= 7 : (7,0), (7,1) \quad (2) \\
P_1 &= 8 : (8,0) \quad (1) \\
P_1 &= 9 : (9,0) \quad (1)
\end{aligned}
$$

Total de combinaciones enteras factibles: $7+6+5+5+4+3+3+2+1+1 = 37$.

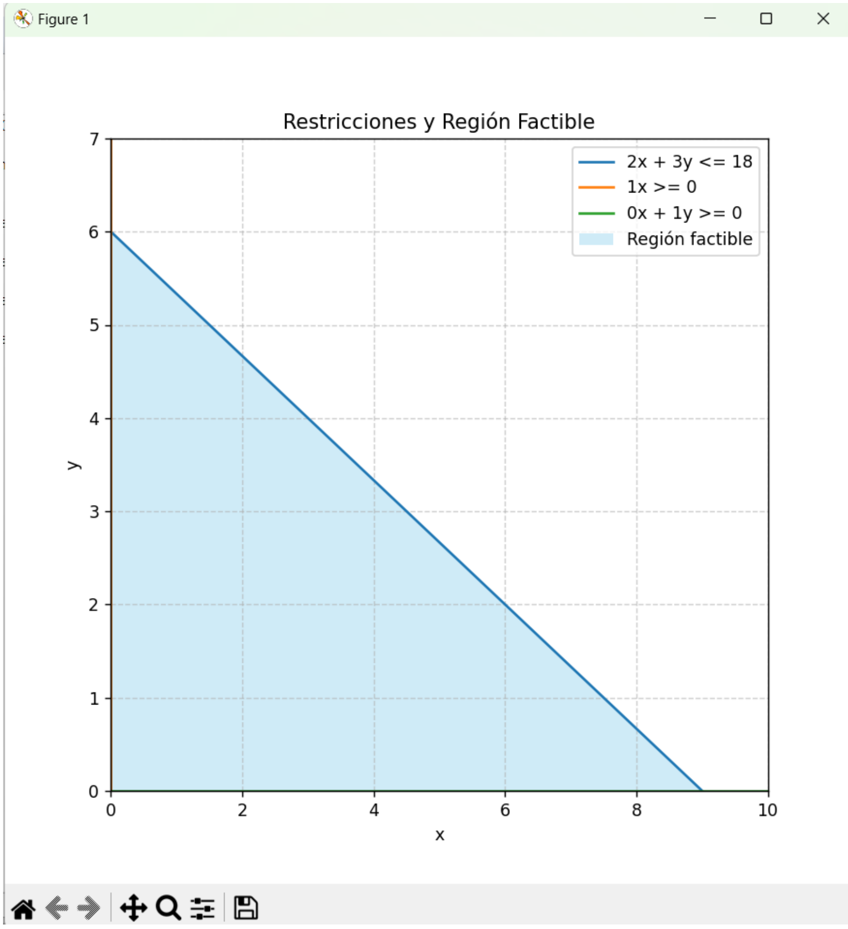
Código en Python

Nota: Agregar este código en la parte final del código adaptado para los ejercicios:

```python
restricciones = [
    (2, 3, 18, "<="),    # 2P1 + 3P2 <= 18
    (1, 0, 0, ">="),     # P1 >= 0
    (0, 1, 0, ">=")      # P2 >= 0
]

graficar_restricciones(restricciones, limites=(0, 10, 0, 7))
```

Representación gráfica del problema



Ejercicio 5. (Componentes para ensamblaje)
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.

Restricciones (formulación)

Variables:

$$A = \text{cantidad de dispositivos tipo A}, \quad B = \text{cantidad de dispositivos tipo B}$$

Restricciones:

$$5A + 10B \leq 50, \quad A \geq 0, \; B \geq 0$$

Combinaciones posibles (valores enteros)

Para valores enteros, A puede tomar valores $0, 1, \ldots, 10$ (ya que $5A \leq 50 \Rightarrow A \leq 10$). Para cada A, B puede ser cualquier entero desde 0 hasta $\left\lfloor \dfrac{50 - 5A}{10} \right\rfloor$.

Las combinaciones (A, B) factibles son:

$$\begin{aligned}
A = 0 &: (0,0), (0,1), (0,2), (0,3), (0,4), (0,5) \quad (6) \\
A = 1 &: (1,0), (1,1), (1,2), (1,3), (1,4) \quad (5) \\
A = 2 &: (2,0), (2,1), (2,2), (2,3), (2,4) \quad (5) \\
A = 3 &: (3,0), (3,1), (3,2), (3,3) \quad (4) \\
A = 4 &: (4,0), (4,1), (4,2), (4,3) \quad (4) \\
A = 5 &: (5,0), (5,1), (5,2) \quad (3) \\
A = 6 &: (6,0), (6,1), (6,2) \quad (3) \\
A = 7 &: (7,0), (7,1) \quad (2) \\
A = 8 &: (8,0), (8,1) \quad (2) \\
A = 9 &: (9,0) \quad (1) \\
A = 10 &: (10,0) \quad (1)
\end{aligned}$$

Total de combinaciones enteras factibles: $6+5+5+4+4+3+3+2+2+1+1 = 36$.

Código en Python

Nota: Agregar este código en la parte final del código adaptado para los ejercicios:

```
restricciones = [
    (5, 10, 50, "<="),  # 5A + 10B <= 50
    (1, 0, 0, ">="),    # A >= 0
    (0, 1, 0, ">=")     # B >= 0
]

graficar_restricciones(restricciones, limites=(0, 12, 0, 6))
```

Representación gráfica del problema