# Plantilla de Referencia: Descenso de Gradiente

Material de Apoyo para la Práctica Dirigida

27 de octubre de 2025

## Introducción

Este documento es una plantilla paso a paso para resolver el Ejercicio 1. Contiene todo el código necesario para implementar y visualizar el Descenso de Gradiente para un problema de ejemplo.

**Tu tarea es adaptar este código para el problema de la práctica dirigida.** Los lugares que debes modificar están marcados con comentarios como `<-- REEMPLAZAR ...`.

## 1. El Problema de Ejemplo

Para esta plantilla, usaremos la función simple $f(x, y) = x^2 + y^2$. Su mínimo está en $(0, 0)$ y su gradiente es $\nabla f(x, y) = [2x, 2y]$.

## 2. Implementación en Python (Plantilla)

### 2.1. Paso 1: Definir la Función y el Gradiente

Aquí es donde traduces las matemáticas de tu ejercicio a código.

```
import numpy as np

# <-- REEMPLAZAR LA FORMULA CON LA DEL EJERCICIO 1
def f(punto):
    """Calcula el valor de la funcion en un punto [x, y]."""
    x = punto[0]
    y = punto[1]
    # Formula de ejemplo: x^2 + y^2
```

```
    return x**2 + y**2
```

Listing 1: Define aquí tu función f(x,y).

```
# <-- REEMPLAZAR LAS FORMULAS CON LAS DERIVADAS DEL EJERCICIO 1
def grad_f(punto):
    """Calcula el gradiente [df/dx, df/dy] en un punto [x, y]."""
    x = punto[0]
    y = punto[1]

    # Derivada de ejemplo df/dx = 2x
    df_dx = 2 * x
    # Derivada de ejemplo df/dy = 2y
    df_dy = 2 * y

    return np.array([df_dx, df_dy])
```

Listing 2: Define aquí tu función gradiente.

## 2.2. Paso 2: Implementar el Algoritmo de Descenso de Gradiente

Esta función es el "motor" de la optimización. **No necesitas modificar esta función**, ya que es genérica.

```
def descenso_gradiente(punto_inicial, alpha, n_iteraciones):
    """Ejecuta el algoritmo de Descenso de Gradiente."""
    trayectoria = [punto_inicial]
    punto_actual = punto_inicial.copy()

    for i in range(n_iteraciones):
        # Llama a la funcion gradiente que definiste en el Paso 1
        grad = grad_f(punto_actual)

        # Actualiza el punto
        punto_actual = punto_actual - alpha * grad

        # Guarda el nuevo punto
        trayectoria.append(punto_actual)

    return np.array(trayectoria)
```

Listing 3: Función genérica para el Descenso de Gradiente.

## 2.3. Paso 3: Ejecutar y Visualizar

Esta es la sección principal donde defines tus parámetros y generas el gráfico. Sigue las instrucciones en los comentarios para adaptar el código.

```
import matplotlib.pyplot as plt

# --- 1. DEFINE TUS PARAMETROS ---
# <-- REEMPLAZAR con los valores pedidos en la practica
punto_inicial = np.array([4.0, 4.0])
alpha = 0.1
n_iteraciones = 20
minimo_teorico = np.array([0.0, 0.0]) # <-- REEMPLAZAR con el minimo de
    tu ejercicio

# --- 2. EJECUTA EL ALGORITMO (sin cambios) ---
trayectoria = descenso_gradiente(punto_inicial, alpha, n_iteraciones)

# --- 3. PREPARA LA MALLA PARA EL GRAFICO DE CONTORNO ---
# <-- AJUSTA los limites de x e y para que tu trayectoria se vea bien
x_vals = np.linspace(-5, 5, 100)
y_vals = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x_vals, y_vals)
Z = f([X, Y]) # Llama a la funcion f que definiste en el Paso 1

# --- 4. CREA EL GRAFICO (Plantilla de Visualizacion) ---
plt.figure(figsize=(12, 10))

# Dibuja las curvas de nivel
contour = plt.contour(X, Y, Z, levels=20, cmap='viridis', alpha=0.6)
plt.clabel(contour, inline=True, fontsize=8)

# Dibuja la trayectoria con flechas
for i in range(len(trayectoria) - 1):
    x_start, y_start = trayectoria[i]
    dx = trayectoria[i+1, 0] - x_start
    dy = trayectoria[i+1, 1] - y_start
    plt.arrow(x_start, y_start, dx, dy,
              head_width=0.2, head_length=0.3,
              fc='red', ec='red', length_includes_head=True,
              label='Trayectoria' if i == 0 else "")

# Dibuja los puntos clave
plt.plot(punto_inicial[0], punto_inicial[1], 'go', markersize=10,
         label=f'Inicio: ({punto_inicial[0]}, {punto_inicial[1]})')
plt.plot(trayectoria[-1, 0], trayectoria[-1, 1], 'mo', markersize=10,
```

```
41          label=f'Final: ({trayectoria[-1, 0]:.2f}, {trayectoria[-1,
    1]:.2f})')
42 plt.plot(minimo_teorico[0], minimo_teorico[1], 'k*', markersize=15,
43          label=f'Minimo teorico: ({minimo_teorico[0]}, {minimo_teorico
    [1]})')
44
45 # Configura el grafico
46 plt.title(f'Descenso de Gradiente (alpha = {alpha})', fontsize=14)
47 plt.xlabel('x', fontsize=12)
48 plt.ylabel('y', fontsize=12)
49 plt.grid(True)
50 plt.legend()
51 plt.colorbar(contour, label='f(x,y)')
52
53 plt.show()
```

Listing 4: Código principal para ejecutar y graficar.