# optim_tarea03

February 17, 2024

# 1 Curso de Optimización I (DEMAT/CIMAT)

# 2 Tarea 3

| Descripción: | Fechas |
|---|---|
| Fecha de publicación del documento: | **Febrero 11, 2024** |
| Fecha límite de entrega de la tarea: | **Febrero 18, 2024** |

## 2.1 Indicaciones

Puede escribir el código de los algoritmos que se piden en una celda de este notebook o si lo prefiere, escribir las funciones en un archivo `.py` independiente e importar la funciones para usarlas en este notebook. Lo importante es que en el notebook aparezcan los resultados de la pruebas realizadas y que:

- Si se requieren otros archivos para poder reproducir los resultados, para mandar la tarea cree un archivo ZIP en el que incluya el notebook y los archivos adicionales.
- Si todos los códigos para que se requieren para reproducir los resultados están en el notebook, no hace falta comprimir el notebook y puede anexar este archivo en la tarea del Classroom.
- Exportar el notebook a un archivo PDF y anexarlo en la tarea del Classroom como un archivo independiente. **No incluya el PDF dentro del ZIP**, porque la idea que lo pueda accesar directamente para poner anotaciones y la calificación de cada ejercicio.

En la descripción de los ejercicios se nombran algunas variables para el algoritmo, pero sólo es para facilitar la descripción. En la implementación pueden nombrar sus variables como gusten.

En los algoritmos se describen las entradas de las funciones. La intención es que tomen en cuenta lo que requiere el algoritmo y que tiene que haber parámetros que permitan controlar el comportamiento del algoritmo, evitando que dejen fijo un valor y que no se puede modificar para hacer diferentes pruebas. Si quieren dar esta información usando un tipo de dato que contenga todos los valores o usar variables por separado, etc., lo pueden hacer y no usen variables globales si no es necesario.

Lo mismo para los valores que devuelve una función. Pueden codificar como gusten la manera en que regresa los cálculos. El punto es que podamos tener acceso a los resultados para poder usarlos, y por eso no es conveniente que la función sólo imprima los valores sin devolverlos.

Para los ejercicios teóricos puede escribir en la celda la solución, o si escribio la solución en una hoja, puede insertar una(s) foto(s) en la que se vea clara la solución. Si le es más facíl insertar la

imagen en un procesador de texto como Word, lo puede utilizar y exportar el documento a PDF y subir el archivo. No lo compacte para que se pueda escribir anotaciones en el PDF.

—

---

## 2.2 Ejercicio 1 (4 puntos)

Programe el Algoritmo 3 de la Clase 6 de descenso máximo con backtracking para calcular el tamaño de paso $\alpha_k$ de manera inexacta.

1. Programar la función que implementa el algoritmo de backtracking (Algoritmo 2 de la Clase 6) que usa la condición de descenso suficiente (condición de Armijo) para seleccionar el tamaño de paso. La función recibe como entrada:

- el valor inicial $\alpha_{ini}$,
- el valor $\rho \in (0, 1)$,
- la constante $c_1 \in (0, 1)$ para la condición de Armijo,
- el punto $\mathbf{x}_k$,
- la función $f$,
- el valor $f_k = f(\mathbf{x}_k)$,
- el valor del gradiente en el punto $\mathbf{x}_k$, $\nabla f_k$,
- la dirección de descenso $\mathbf{p}_k$, y
- el número máximo de iteraciones $N_b$.

La función devuelve - el tamaño de paso $\alpha_k$, - el número $i_k$ de iteraciones realizadas por el algoritmo de backtracking

2. Programar la función que implementa el algoritmo de descenso máximo con backtracking. Ésta recibe como entrada:

- La función $f(\mathbf{x})$,
- el gradiente $\nabla f(\mathbf{x})$ de la función $f$,
- un punto inicial $\mathbf{x}_0$,
- las tolerancia $\tau > 0$,
- el número máximo de iteraciones $N$ para el algoritmo de descenso máximo,
- el valor inicial $\alpha_{ini}$,
- el valor $\rho \in (0, 1)$,
- la constante $c_1 \in (0, 1)$ para la condición de Armijo y
- el número máximo de iteraciones $N_{gs}$ para el método de la sección dorada.

La función devuelve - El último punto $\mathbf{x}_k$ generado por el algoritmo, - el número $k$ de iteraciones realizadas y - Una variable indicadora que es $True$ si el algoritmo termina por cumplirse la condición de paro ($\|\alpha_k \mathbf{p}_k\| < \tau$) o $False$ si termina porque se alcanzó el número máximo de iteraciones. - Si $n \neq 2$, devuelve un arreglo vació. En caso contrario, devuelve un arreglo que contiene las

componentes de los puntos de la secuencia, el tamaño de paso y la cantidad de iteraciones que hizo el algoritmo de backtracking en cada iteración:

$$\begin{array}{cccc} x_1^{(0)} & x_2^{(0)} & \alpha_0 & i_0 \\ x_1^{(1)} & x_2^{(1)} & \alpha_0 & i_1 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(k)} & x_2^{(k)} & \alpha_k & i_k \end{array}$$

3. Para probar el algoritmo, programe las siguientes funciones, calcule su gradiente de manera analítica y programe la función correspondiente. Use cada punto $\mathbf{x}_0$ como punto inicial del algoritmo.

**Función de Himmelblau:** Para $\mathbf{x} = (x_1, x_2)$

$$f(\mathbf{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2.$$
$$\mathbf{x}_0 = (2., 4.)$$
$$\mathbf{x}_0 = (0., 0.)$$

**Función de Beale :** Para $\mathbf{x} = (x_1, x_2)$

$$f(\mathbf{x}) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2.$$
$$\mathbf{x}_0 = (2., 3.)$$
$$\mathbf{x}_0 = (2., 4.)$$

**Función de Rosenbrock:** Para $\mathbf{x} = (x_1, x_2, ..., x_n)$

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \quad n \geq 2.$$
$$\mathbf{x}_0 = (-2.1, 4.5)$$
$$\mathbf{x}_0 = (-1.2, 1.0)$$
$$\mathbf{x}_0 = (-2.1, 4.5, -2.1, 4.5, -2.1, 4.5, -2.1, 4.5, -2.1, 4.5)$$
$$\mathbf{x}_0 = (-1.2, 1.0, -1.2, 1.0, -1.2, 1.0, -1.2, 1.0, -1.2, 1.0)$$

En la página Test functions for optimization pueden ver las gráficas de estas funciones y sus mínimos locales.

Use - la tolerancia $\tau = \sqrt{n}\epsilon_m^{1/2}$, donde $\epsilon_m$ es el épsilon de la máquina, - $\alpha_{ini} = 1.0$, - $\rho = 0.8$, - $c_1 = 0.1$, - el número de iteraciones máximas $N = 30000$ para el descenso máximo - el número de iteraciones máximas $N_b = 600$ para el método de backtracking.

En cada caso imprima los resultados: - El número de iteraciones realizadas $k$ - El punto $\mathbf{x}_k$ obtenido - $f(\mathbf{x}_k)$ - $\|\nabla f(\mathbf{x}_k)\|$ - La variable que indica si el algoritmo terminó porque se cumplió el criterio de paro o no. Además, si $n = 2$, imprima - El valor promedio de los tamaños de paso $\alpha_0, \alpha_1, ..., \alpha_k$. - El valor promedio de las iteraciones $i_0, i_1, ..., i_k$ realizadas por el algoritmo de backtracking. - La gráfica de los contornos de nivel de la función y la trayectoria de los puntos $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_k$.

3. Nocedal sugiere que la constante $c_1$ sea del orden de 0.0001. Use $c_1 = 0.0001$ y repeta la prueba con la función de Beale y explique en qué casos conviene usar un valor grande o pequeño de $c_1$.

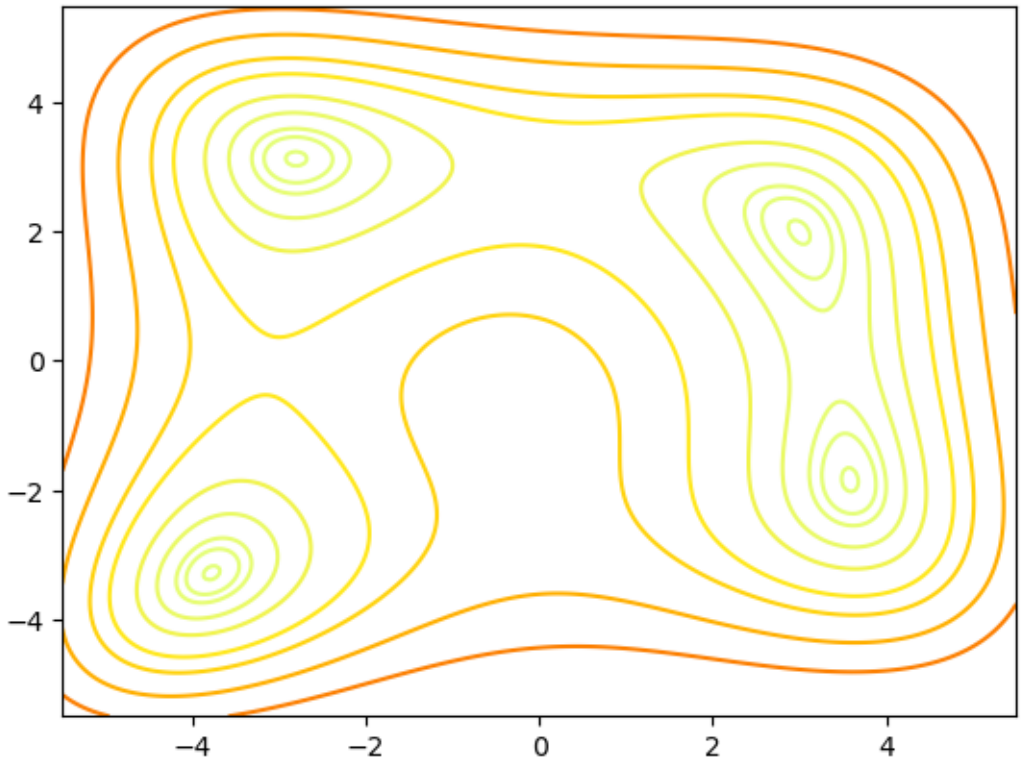
### 2.2.1 Solución:

[1]:
```
import matplotlib.pyplot as plt
import numpy as np

def fncHimmelblau(x, fparam=None):
    return (x[0]**2 + x[1] - 11)**2 + (x[0] + x[1]**2 - 7)**2;


def contornosFnc2D(fncf, xleft, xright, ybottom, ytop, levels):
    # Crea una discretización uniforme del intervalo [xleft, xright]
    ax = np.linspace(xleft, xright, 250)
    # Crea una discretización uniforme del intervalo [ybottom, ytop]
    ay = np.linspace(ybottom, ytop, 250)
    # La matriz mX que tiene las abscisas
    mX, mY = np.meshgrid(ax, ay)
    # Se crea el arreglo mZ con los valores de la función en cada nodo
    vz = np.zeros( len(ax)*len(ay) )
    for i,xy in enumerate(zip(mX.flatten(), mY.flatten())):
        vz[i] = fncf(xy)
    mZ = vz.reshape( len(ay), len(ax) )
    # Grafica de las curvas de nivel
    fig, ax = plt.subplots()
    CS = ax.contour(mX, mY, mZ, levels, cmap='Wistia')

contornosFnc2D(fncHimmelblau, xleft=-5.5, xright=5.5, ybottom=-5.5, ytop=5.5,
               levels=[0.5, 5, 10, 25, 50, 100, 150, 250, 400])
```




—

---

### 2.3 Ejercicio 2 (3 puntos)

Reprograme el Algoritmo 3 de la Clase 6 de descenso máximo para calcular el tamaño de paso inicial del algoritmo de backtracking.

1. Modifique la función del Punto 2 del Ejercicio 1 de modo que en la iteración $k = 0$ se invoque a la función que ejecuta el backtracking usando el valor $\alpha_{ini}$ dado, es decir,

   $backtracking(f, \mathbf{x}_k, f_k, \nabla f_k, \mathbf{p}_k, \alpha_{ini}, \rho, c_1, N_b)$

   y para $k > 0$ se calcule el valor inicial del tamaño de paso como

$$\bar{\alpha} = \alpha_{k-1} \min \left\{ 100, \ \frac{\nabla f_{k-1}^{\top} \mathbf{p}_{k-1}}{\nabla f_k^{\top} \mathbf{p}_k} \right\}$$

y se use este valor al ejecutar backtracking: $backtracking(f, \mathbf{x}_k, f_k, \nabla f_k, \mathbf{p}_k, \bar{\alpha}, \rho, c_1, N_b)$.

2. Repita las pruebas del Ejercicio 1, imprimiendo los mismos resultados:

- El número de iteraciones realizadas $k$
- El punto $\mathbf{x}_k$ obtenido
- $f(\mathbf{x}_k)$
- $\|\nabla f(\mathbf{x}_k)\|$
- La variable que indica si el algoritmo terminó porque se cumplió el criterio de paro o no. Además, si $n = 2$, imprima
- El valor promedio de los tamaños de paso $\alpha_0, \alpha_1, ..., \alpha_k$.
- El valor promedio de las iteraciones $i_0, i_1, ..., i_k$ realizadas por el algoritmo de backtracking.
- La gráfica de los contornos de nivel de la función y la trayectoria de los puntos $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_k$.

3. Con base en el valor promedio de las iteraciones realizadas por el algoritmo de backtracking, el valor $k$ y las gráficas, escriba un comentario sobre si el cambio realizado ayuda al desempeño del método. ¿Hay alguna diferencia entre importante entre calcular el tamaño de paso de manera exacta con respecto a la búsqueda inexacta?

#### 2.3.1 Solución:

[ ]:

—

---

### 2.4 Ejercicio 3 (1.5 puntos)

Sea $f(\mathbf{x}) = f(x_1, x_2) = 5 + x_1^2 + x_2^2$. Si $\mathbf{x}_0 = (-1, 1)^\top$, $\mathbf{p}_0 = (1, 0)$ y $c_1 = 10^{-4}$. Verifique de $\mathbf{p}_0$ es una dirección de descenso y encuentre el valor más grande $\alpha > 0$ que satisface la condición de descenso suficiente:

$$f(\mathbf{x}_0 + \alpha \mathbf{p}_0) \leq f(\mathbf{x}_0) + c_1 \alpha \mathbf{p}_0^\top \nabla f(\mathbf{x}_0).$$

#### 2.4.1 Solución:

[ ]:

—

## 2.5 Ejercicio 4 (1.5 puntos)

Sea $f : \mathbb{R}^n \to \mathbb{R}$ y $\mathbf{S}$ una matriz no singular de tamaño $n \times n$. Si $\mathbf{x} = \mathbf{S}\mathbf{y}$ para $\mathbf{y} \in \mathbb{R}^n$ y definimos $g(\mathbf{y}) = f(\mathbf{S}\mathbf{y})$, aplicando la regla de la cadena muestre que

$$\nabla g(\mathbf{y}) = \mathbf{S}^\top \nabla f(\mathbf{x}).$$

Entonces aplicando el método de máximo descenso a la función $g$ es

$$\mathbf{y}_{k+1} = \mathbf{y}_k - \alpha_k \mathbf{S}^\top \nabla f(\mathbf{S}\mathbf{y}_k).$$

Multiplicando por $\mathbf{S}$ ambos miembros de la ecuación y usando la notación $\mathbf{x}_k = \mathbf{S}\mathbf{y}_k$:

$$\mathbf{x}_{k+1} = \mathbf{S}\mathbf{y}_{k+1} = \mathbf{S}\mathbf{y}_k - \alpha_k \mathbf{S}\mathbf{S}^\top \nabla f(\mathbf{S}\mathbf{y}_k) = \mathbf{x}_k - \alpha_k \mathbf{S}\mathbf{S}^\top \nabla f(\mathbf{x}_k).$$

Si $\mathbf{D} = \mathbf{S}\mathbf{S}^\top$, obtenemos el método de *gradiente escalado*:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{D} \nabla f(\mathbf{x}_k).$$

Muestre que $-\mathbf{D}\nabla f(\mathbf{x}_k)$ es una dirección de descenso.

### 2.5.1 Solución:

[ ]:

—