Curso de Optimización

Tarea 6

Descripción:	Fechas
Fecha de publicación del documento:	**Marzo 25, 2025**
Fecha límite de entrega de la tarea:	**Abril 3, 2025**

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 comprimirlo y puede anexar sólo el notebook en la tarea
- Exportar el notebook a un archivo PDF y anexarlo en la tarea como un archivo independiente. **No lo incluya dentro del ZIP**, porque la idea que lo pueda accesar directamente para poner anotaciones y la calificación de cada ejercicio.

Ejercicio 1 (3 puntos)

Considere las siguientes funciones y los puntos $\mathbf{x}_0$:

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

$$f(\mathbf{x}) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2.$$

- $\mathbf{x}_0 = (2, 3)$

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

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

- $\mathbf{x}_0 = (2, 4)$

Función de Hartmann de dimensión 6 (Referencia): Para $\mathbf{x} = (x_1, x_2, \dots, x_6)$

$$f(\mathbf{x}) = -\frac{1}{1.94}\left[2.58 + \sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{i=1}^{6} a_{ij}(x_j - p_{ij})^2\right)\right],$$

donde

$$\alpha = (1.0, 1.2, 3.0, 3.2)$$

$$\mathbf{A} = [a_{ij}] = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix},$$

$$\mathbf{P} = [p_{ij}] = 10^{-4}\begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}.$$

- $\mathbf{x}_0 = (0, 0, 0, 0, 0, 0)$

NOTA 1: Esta función tiene 6 óptimos locales. El óptimo global es $\mathbf{x}_* = (0.20169, 0.15001, 0.476874, 0.275332, 0.311652, 0.6573)$, y $f(\mathbf{x}_*) = -3.0424$.

NOTA 2: Para esta función necesita calcular el gradiente y programar esta función para usarla en los siguientes ejercicios.

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

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right] \quad n \geq 2.$$

- $\mathbf{x}_0 = (-1.2, 1.0) \in \mathbb{R}^2$
- $\mathbf{x}_0 = (-1.2, 1.0, \dots, -1.2, 1.0) \in \mathbb{R}^{200}$
- $\mathbf{x}_0 = (-1.2, 1.0, \dots, -1.2, 1.0) \in \mathbb{R}^{600}$

---

1. Calcule y programe el Hessiano de cada función.
2. Escriba una función que recibe como entrada una matriz simétrica. La función hace lo siguiente:

- Calcula los eigenvalores de la matriz. Puede usar la función `numpy.linalg.eigvalsh()` para calcular los eigenvalores.
- Devuelve el eigenvalor más pequeño y el eigenvalor más grande.

3. Escriba una función que reciba el eigenvalor más pequeño λ_1 y el eigenvalor más grande λ_n de una matriz simétrica. La función imprime λ_1, λ_n y un mensaje de acuerdo a las siguientes condiciones:

- Si $\lambda_1 > 0$, imprima `Matriz definida positiva`.
- Si $\lambda_n < 0$, imprima `Matriz definida negativa`.
- Si $\lambda_1 < 0$ y $\lambda_n > 0$, imprima `Matriz indefinida`.

4. Evalúe el Hessiano $\mathbf{H}_0$ en los puntos $\mathbf{x}_0$ dados y use la funciones de los Punto 2 y 3 para indicar si la matriz es definida positiva o no.

Solución:

In [ ]:

```

```

Ejercicio 2 (4 puntos)

Programar el método de Newton con tamaño de paso α_k inexacto usando el algoritmo de backtracking usando la condición de descenso suficiente.

> La función que implementa el método de Newton recibe como parámetros:
>
> - la función objetivo $f : \mathbb{R}^n \to \mathbb{R}$,
> - el gradiente $g(\mathbf{x})$ de la función objetivo,
> - la Hessiana $H(\mathbf{x})$ de la función f,
> - el punto inicial $\mathbf{x}_0$,
> - un número máximo de iteraciones N,
> - la tolerancia $\tau > 0$, y
> - los parámetros α_{ini}, c_1, ρ y el número máximo de iteraciones N_b para el algoritmo de backtracking.
>
> Fijar $m = 0$.
>
> Para $k = 0, 1, \ldots, N-1$ repetir los siguientes pasos:
>
> 1. Calcular el gradiente $\mathbf{g}_k$ en el punto $\mathbf{x}_k$: $\mathbf{g}_k = g(\mathbf{x}_k)$.
> 2. Si $\|\mathbf{g}_k\| < \tau$, hacer $res = 1$ y terminar el ciclo.

> 3. Si no se cumple el criterio de paro, calcular la Hessiana en el punto $\mathbf{x}_k$: $\mathbf{H}_k = H(\mathbf{x}_k)$.
> 4. Intentar calcular la factorización de Cholesky de $\mathbf{H}_k$. Si la factorización es exitosa, $\mathbf{H}_k = \mathbf{L}\mathbf{L}^\top$, resolver el sistema de ecuaciones $\mathbf{L}\mathbf{L}^\top\mathbf{p}_k = -\mathbf{g}_k$ para obtener la dirección de descenso $\mathbf{p}_k$.
> 5. Si falla el cálculo de la factorización, definir la dirección de descenso como $\mathbf{p}_k = -\mathbf{g}_k$ y hacer $m = m + 1$.
> 6. Calcular el tamaño de paso α_k usando el algoritmo de backtracking con la condición de descenso suficiente.
> 7. Calcular el siguiente punto de la secuencia como $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$.
>
> Devolver el punto $\mathbf{x}_k$, $\mathbf{g}_k$, k, m y res.
>
> .

Nota: Para calcular la factorización de Cholesky y resolver el sistema de ecuaciones puede usar las funciones `scipy.linalg.cho_factor` y `scipy.linalg.cho_solve`. Si la matriz no es definida positiva, la función `cho_factor` lanza la excepción `scipy.linalg.LinAlgError`. Puede manejar este error mediante una excepción en la que se defina la dirección de descenso como $-\mathbf{g}_k$.

1. Programe la función que implementa el algoritmo del método de Newton. Si la dimensión de las variables es 2, almacene los puntos $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k$ que genera el algoritmo y haga que la función devuelva esta lista para usarlos para graficar la trayectoria.

2. Pruebe el algoritmo con las funciones del Ejercicio 1 usando los puntos iniciales $\mathbf{x_0}$ indicados, el número de iteraciones máximas $N = 10000$ y la tolerancia $\tau = \sqrt{n\epsilon_m}$, donde ϵ_m es el épsilon de máquina. Para el algoritmo del backtraking use $\alpha_{ini} = 1$, $c_1 = 0.1$, $\rho = 0.6$ y el máximo de iteraciones $N_b = 100$.

   Imprima los valores siguientes:

- $f(\mathbf{x}_0)$.
- Imprimir el número de iteraciones k.
- Imprimir el número m de veces en que $\mathbf{H}_k$ no fue definida positiva.
- El valor de res.
- La norma $\|\mathbf{g}_k\|$.
- Si $n <= 6$, imprimir $\mathbf{x}_k$. En caso contrario, imprimir las primeras y las últimas 3 componentes de $\mathbf{x}_k$.

- $f(\mathbf{x}_k)$.

  Además, si la dimensión de la variable es $n = 2$, grafique los contornos de nivel de la función y la trayectoria definida por los puntos $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k$.

3. Revise el valor del contador m y escriba un comentario sobre si el algoritmo se comporta más como el método de descenso máximo o como el método de Newton, y la necesidad de modificar la matriz Hessiana.

Solución:

In [ ]:

```

```

_

Ejercicio 3 (3 puntos)

Modifique el algoritmo del método de Newton del Ejercicio 2 que implementa el algoritmo del método de Newton reemplazando el Paso 4 (Intentar calcular la factorización de Cholesky) y el Paso 5 (Si falla el cálculo de la factorización) para que quede de la siguiente manera:

> La función que implementa el método de Newton recibe como parámetros:
>
> - la función objetivo $f : \mathbb{R}^n \to \mathbb{R}$,
> - el gradiente $g(\mathbf{x})$ de la función objetivo,
> - la Hessiana $H(\mathbf{x})$ de la función f,
> - el punto inicial $\mathbf{x}_0$,
> - un número máximo de iteraciones N,
> - la tolerancia $\tau > 0$,
> - **un incremento** $\delta > 0$,y
> - los parámetros α_{ini}, c_1, ρ y el número máximo de iteraciones N_b para el algoritmo de backtracking.
>
> Definir $\epsilon = \sqrt{\epsilon_m}$.

> Para $k = 0, 1, \ldots, N-1$ repetir los siguientes pasos:
>
> 1. Calcular el gradiente $\mathbf{g}_k = g(\mathbf{x}_k)$ en el punto $\mathbf{x}_k$.
> 2. Si $\|\mathbf{g}_k\| < \tau$, hacer $res = 1$ y terminar el ciclo.
> 3. Si no se cumple el criterio de paro, calcular la Hessiana $\mathbf{H}_k = H(\mathbf{x}_k)$ en $\mathbf{x}_k$.
> 4. Usar la función del Punto 2 del Ejercicio 1 para obtener el eigenvalor más pequeño λ_1 de $\mathbf{H}_k$.
> 5. Si $\lambda_1 < \epsilon$, modificar la Hessiana sumando $\delta + |\lambda_1|$ a los elementos de la diagonal. Es decir se reemplaza $\mathbf{H}_k$ por la matriz $\mathbf{H}_k + (\delta + |\lambda_1|)\mathbf{I}$, donde $\mathbf{I}$ es la matriz identidad, lo cual hace que todos los eigenvalores de la matriz sean positivos y mayores que δ.
> 6. Calcular la factorización de Cholesky $\mathbf{L}\mathbf{L}^\top$ de la matriz modificada.
> 7. Resolver el sistema de ecuaciones $\mathbf{L}\mathbf{L}^\top \mathbf{p}_k = -\mathbf{g}_k$ para obtener la dirección de descenso $\mathbf{p}_k$.
> 8. Calcular el tamaño de paso α_k usando el algoritmo de backtracking con la condición de descenso suficiente.
> 9. Calcular $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$.
>
> Devolver el punto $\mathbf{x}_k$, $\mathbf{g}_k$, k y res.
>
> .

1. Programe la función que implementa el algoritmo anterior.
2. Repita las pruebas del Ejercicio 2 usando $\delta = 0.001$ y $\delta = 10.0$.
3. Repita la prueba para la función de Rosenbrock con $\delta = 100.0$.
4. En este algoritmo se tiene un costo adicional por el cálculo de los eigenvalores de la matriz Hessiana, el cual se incrementa conforme aumenta el valor de n. Explique si conviene hacer este cambio y qué tan sensible es el algoritmo al cambiar el valor de δ.

Solución:

```
In [ ]:
```