Curso de Optimización (DEMAT)

Tarea 5

Leslie Janeth Quincosa Ramírez

Descripción:	Fechas
Fecha de publicación del documento:	**Marzo 6, 2022**
Fecha límite de entrega de la tarea:	**Marzo 13, 2022**

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 del Classroom.
- Exportar el notebook a un archivo PDF y anexarlo en la tarea del Classroom 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 (6 puntos)

Programar el método de descenso máximo con tamaño de paso fijo y probarlo.

El algoritmo recibe como parámetros la función gradiente $g(x)$ de la función objetivo, un punto inicial x_0, el valor del tamaño de paso α, un número máximo de iteraciones N, la tolerancia $\tau > 0$. Fijar $k = 0$ y repetir los siguientes pasos:

1. Calcular el gradiente g_k en el punto x_k, $g_k = g(x_k)$.
2. Si $\|g_k\| < \tau$, hacer $res = 1$ y terminar.
3. Elegir la dirección de descenso como $p_k = -g_k$.
4. Calcular el siguiente punto de la secuencia como

$$x_{k+1} = x_k + \alpha p_k$$

5. Si $k + 1 \geq N$, hacer $res = 0$ y terminar.
6. Si no, hacer $k = k + 1$ y volver el paso 1.
7. Devolver el punto x_k, g_k, k y res.

De acuerdo con la proposición vista en la clase 12, para que el método de máximo descenso con paso fijo para funciones cuadráticas converja se requiere que el tamaño de paso α cumpla con

$$0 < \alpha < \frac{2}{\lambda_{\max}(A)} = \alpha_{\max},$$

donde $\lambda_{\max}(A)$ es el eigenvalor más grande de A.

1. Escriba una función que implementa el algoritmo de descenso máximo con paso fijo.

2. **Programe las funciones cuadráticas y sus gradientes**

$$f_i(x) = \frac{1}{2} x^\top \mathbf{A}_i x - \mathbf{b}_i^\top x, \quad i = 1,2$$

donde

$$\mathbf{A}_1 = \begin{bmatrix} 1.18 & 0.69 \\ 0.69 & 3.01 \end{bmatrix}, \qquad \mathbf{b}_1 = \begin{pmatrix} -0.24 \\ 0.99 \end{pmatrix}.$$

$$\mathbf{A}_2 = \begin{bmatrix} 6.36 & -3.07 & -2.8 & -3.42 & -0.68 \\ -3.07 & 10.19 & 0.74 & 0.5 & 0.72 \\ -2.8 & 0.74 & 4.97 & -1.48 & 1.93 \\ -3.42 & 0.5 & -1.48 & 4.9 & -0.97 \\ -0.68 & 0.72 & 1.93 & -0.97 & 3.21 \end{bmatrix}, \qquad \mathbf{b}_2 = \begin{pmatrix} 0.66 \\ 0.37 \\ -2.06 \\ 0.14 \\ 1.36 \end{pmatrix}.$$

3. **Fije el número máximo de iteraciones** $N = 2000$ **y la tolerancia** $\tau = \sqrt{\epsilon_m}$ **, donde** ϵ_m **es el épsilon de la máquina. Para cada función cuadrática, calcule** $\alpha_{\max}$ **de la matriz** A_i **. Pruebe con los tamaños de paso** α **igual a** $1.1\alpha_{\max}$ **y** $0.9\alpha_{\max}$ **. Use el punto inicial**

$$\mathbf{x}_0 = \begin{pmatrix} -38.12 \\ -55.87 \end{pmatrix} \quad \text{para} \quad f_1$$

$$\mathbf{x}_0 = \begin{pmatrix} 4.60 \\ 6.85 \\ 4.31 \\ 4.79 \\ 8.38 \end{pmatrix} \quad \text{para} \quad f_2$$

4. **En cada caso imprima** x_k **,** $\|g_k\|$ **, el número de iteraciones** k **y el valor de** res **.**

Solución:

In [32]:

```
# En esta celda puede poner el código de las funciones
# o poner la instrucción para importarlas de un archivo .py
import numpy as np
from numpy import linalg as LA
from numpy.linalg import eigvals
import sys

#g(x) de la función objetivo, un punto inicial  x0 , el valor del tamaño de paso  α , un
número máximo de iteraciones  N , la tolerancia  τ>0 .
def DescMAxPasoFijo (g, x0, alpha, N, tol):
    xk = x0
    for k in range(N):
```

```
        gk = g(xk)
        norm = LA.norm(gk)
        if norm < tol:
            res = 1
            break
        else:
            pk = -gk
            xk = xk + alpha*pk
            if k+1 >= N:
                res = 0
                break
    return xk, gk, k, res
```

In [33]:

```
# Pruebas del algoritmo
eps = sys.float_info.epsilon

def Probar(f, g, x0, A, b):
    N = 2000
    tol = eps**(1/3)
    eigval = np.linalg.eigvals(A)
    eigmax = eigval.max()
    alphamax = 2/eigmax
    alphas = [0.9*alphamax, 1.1*alphamax]
    for alpha in alphas:
        xk, gk, k, res = DescMAxPasoFijo (g, x0, alpha, N, tol)
        print('res:', res)
        print('Valor k:', k)
        print('xk:', xk)
        print('k:', k)
        print('||gk||:', LA.norm(gk))
        print('\n')
```

In [34]:

```
A1 = np.array([[1.18, 0.69], [0.69, 3.01]])
b1 = np.array([-0.24, 0.99])


A2 = np.array([[6.36, -3.07, -2.8, -3.42, -0.68], [-3.07, 10.19, 0.74, 0.5, 0.72], [-2.8
, 0.74, 4.97, -1.48, 1.93], [-3.42, 0.5, -1.48, 4.9, -0.97], [-0.68, 0.72, 1.93, -0.97,
3.21]])

b2 = np.array([0.66, 0.37, -2.06, 0.14, 1.36])

x01 = np.array([-38.12, -55.87])
x02 = np.array([4.60, 6.85, 4.31, 4.79, 8.38])

def f1(x):
    return (1/2)*x.T@A1@x-b1.T@x

def g1(x):
    return A1@x-b1

def f2(x):
    return (1/2)*x.T@A2@x-b2.T@x

def g2(x):
    return A2@x-b2

print('Método del descenso máximo con tamaño de paso fijo para f1')
Probar(f1, g1, x01, A1, b1)

print('Método del descenso máximo con tamaño de paso fijo para f2')
Probar(f2, g2, x02, A2, b2)
```

```
Método del descenso máximo con tamaño de paso fijo para f1
res: 1
Valor k: 78
xk: [-0.45696972  0.43365567]
k: 78
```

```
||gk||: 5.846955574558156e-06


res: 0
Valor k: 1999
xk: [-4.77996787e+159 -1.42775834e+160]
k: 1999
||gk||: inf


Método del descenso máximo con tamaño de paso fijo para f2
res: 1
Valor k: 719
xk: [-2.77191498 -0.52190225 -3.05956961 -2.57611286  1.01464394]
k: 719
||gk||: 6.005936793437383e-06


res: 0
Valor k: 1999
xk: [-7.41437598e+158  9.63088421e+158  3.28353818e+158  2.91033121e+158
  1.57034121e+158]
k: 1999
||gk||: inf
```

Comentarios sobre las trayectorias:

Notemos que al elegir un α menor al máximo, de acuerdo a la proposición, entonces las funciones convergieron en pocos pasos.

Por otro lado, si tomamos un α mayor al máximo, entonces el método falla, ya que el tamaño de paso es grande y no logra converger y se cicla.

Ejercicio 2 (4 puntos)

Pruebe el método de descenso máximo con paso fijo aplicado a la función de Rosenbrock.

Encuentre un valor adecuado para α para que el algoritmo converja. Use como punto inicial el punto $(-12, 10)$.

Imprima x_k, $\|g_k\|$, el número de iteraciones k y el valor de res.

Solución:

In [35]:

```
# En esta celda puede poner el código de las funciones
# o poner la instrucción para importarlas de un archivo .py
def Backtraking(f, fk, gk, xk, pk, a0, rho, c):
    a = a0
    while f(xk + a*pk) > fk +c*a*gk.T@pk:
        a = rho*a
    return a

def DescMAxPasoFijoBack (f, g, x0, alpha, N, tol):
    xk = x0
    for k in range(N):
      gk = g(xk)
```

```
        fk = f(xk)
        norm = LA.norm(gk)
        if norm < tol:
            res = 1
            break
        else:
            pk = -gk
            rho = 0.02
            c = 0.0001
            alpha = Backtraking(f, fk, gk, xk, pk, alpha, rho, c)
            xk = xk + alpha*pk
            if k+1 >= N:
                res = 0
                break
    return xk, gk, k, res

def function(x):
    xT = x.T
    return 100 * (xT[1] - xT[0]**2)**2 + (1-xT[0])**2

def gradient(x):
    xT = x.T
    return np.array([-400*xT[0]*(xT[1]- xT[0]**2) -2*(1-xT[0]), 200*(xT[1]-xT[0]**2)])

def Hessian(x):
    xT = x.T
    return np.array([[1200*xT[0]-400*xT[1]+2, -400*xT[0]], [-400*xT[0], 200]])

x03 = np.array([-12, 10])
x04 = np.array([-1.2, 1.0])
```

In [36]:

```
# Pruebas realizadas a la función de Rosenbrock
print('Función de Rosenbrock con método de Descenso Máximo de Paso fijo, eligiendo una al
pha adecuada')

def Probar2(f, g, x0, alpha):
    N = 50000
    for tol in [eps**(1/3), 1e-3]:
      xk, gk, k, res = DescMAxPasoFijoBack (f, g, x0, alpha, N, tol)
      print('res:', res)
      print('Valor k:', k)
      print('xk:', xk)
      print('k:', k)
      print('||gk||:', LA.norm(gk))
      print('fxk:', f(xk))
      print('\n')

Probar2(function, gradient, x03, eps)
Probar2(function, gradient, x04, eps)

def Probar3(f, g, x0, alpha):
    N = 50000
    for tol in [eps**(1/3), 1e-3]:
      xk, gk, k, res = DescMAxPasoFijo(g, x0, alpha, N, tol)
      print('res:', res)
      print('Valor k:', k)
      print('xk:', xk)
      print('k:', k)
      print('||gk||:', LA.norm(gk))
      print('fxk:', f(xk))
      print('\n')

print('Función de Rosenbrock con método de Descenso Máximo de Paso fijo con un alpha cual
quiera')
Probar3(function, gradient, x03, 1e-3)
Probar3(function, gradient, x04, 1e-3)
```

```
Función de Rosenbrock con método de Descenso Máximo de Paso fijo, eligiendo una alpha ade
cuada
res: 0
```

```
Valor k: 49999
xk: [-11.99999286  10.0000003 ]
k: 49999
||gk||: 643782.8614331756
fxk: 1795764.398611263


res: 0
Valor k: 49999
xk: [-11.99999286  10.0000003 ]
k: 49999
||gk||: 643782.8614331756
fxk: 1795764.398611263


res: 0
Valor k: 49999
xk: [-1.2  1. ]
k: 49999
||gk||: 232.8676838583469
fxk: 24.199999396997907


res: 0
Valor k: 49999
xk: [-1.2  1. ]
k: 49999
||gk||: 232.8676838583469
fxk: 24.199999396997907


Función de Rosenbrock con método de Descenso Máximo de Paso fijo con un alpha cualquiera
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:35: RuntimeWarning: overflow encountered in double_scalars
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:35: RuntimeWarning: invalid value encountered in double_scalars
```

```
res: 0
Valor k: 49999
xk: [nan nan]
k: 49999
||gk||: nan
fxk: nan


res: 0
Valor k: 49999
xk: [nan nan]
k: 49999
||gk||: nan
fxk: nan


res: 1
Valor k: 27568
xk: [0.99999323 0.99998644]
k: 27568
||gk||: 6.053167442160728e-06
fxk: 4.587385991543984e-11


res: 1
Valor k: 14789
xk: [0.99888303 0.99776284]
k: 14789
||gk||: 0.000999938589393288
fxk: 1.2496137290730113e-06
```

Resultados:

- Primeramente intenté hacerlo con un α fijo y pequeño, pero no convergía al mínimo conocido $x^* = (1, 1)$.
- Por lo que utilice el método de Backtracking para encontrar un α adecuado. Con valores $\rho, c \in (0, 1)$, pero para alcanzar el mínimo tampoco encontró el valor.
- Con este método utilizando una tolerancia de $\varepsilon_m^{1/3}$, no convergen las iteraciones.
- Cuando cambiamos el x_0, entonces el método sí converge, tanto con el α del Backtraking como para una α cualquiera.
- Con una tolerancia más grande, sí converge y también se aproxima muy bien al mínimo conocido.