ESCUELA POLITÉCNICA NACIONAL
MÉTODOS NUMÉRICOS

Leandro Bravo GR1CC

Tarea 11 - Gauss-Jacobi y Gauss-Seidel

```
%load_ext autoreload
import numpy as np
from src import gauss_jacobi,gauss_jacobi_iteraciones
from src import gauss_seidel, gauss_seidel_iteraciones,es_diagonal_estrictamente_dominante
```

Conjunto de Ejercicios

1. Encuentre las primeras dos iteraciones del método de Jacobi para los siguientes sistemas lineales, por medio de $x^{(0)} = 0$:

a.
$$\begin{aligned} 3x_1 - x_2 + x_3 &= 1,\\ 3x_1 + 6x_2 + 2x_3 &= 0,\\ 3x_1 + 3x_2 + 7x_3 &= 4 \end{aligned}$$

```
%autoreload 2
A = [[3,-1,1],[3,6,2],[3,3,7]]
b = np.array([1,0,4],dtype=float)
x0=np.zeros(len(b))
max_iter = 2
tol = 10e-6

x = gauss_jacobi_iteraciones(A, b, x0, tol, max_iter)
```

```
Iteración 1 , solución aproximada: [0.33333333 0.         0.57142857]
Iteración 2 , solución aproximada: [0.33333333 0.         0.57142857]
```

b.
$$\begin{aligned} 10x_1 - x_2 &= 9,\\ -x_1 + 10x_2 - 2x_3 &= 7,\\ -2x_2 + 10x_3 &= 6 \end{aligned}$$

```
%autoreload 2
A = [[10,-1,0],[-1,10,-2],[0,-2,10]]
b = np.array([9,7,6],dtype=float)
x0=np.zeros(len(b))
max_iter = 2
tol = 10e-6

x = gauss_jacobi_iteraciones(A, b, x0, tol, max_iter)
```

```
Iteración 1 , solución aproximada: [0.9 0.7 0.6]
Iteración 2 , solución aproximada: [0.9 0.7 0.6]
```

c.
$$\begin{aligned} 10x_1 - 5x_2 &= 6,\\ 5x_1 + 10x_2 - 4x_3 &= 25,\\ -4x_2 + 8x_3 - x_4 &= -11,\\ -x_3 + 5x_4 &= -11 \end{aligned}$$

```
%autoreload 2
A = [[10,-5,0,0],[5,10,-4,0],[0,-4,8,-1],[0,0,-1,5]]
b = np.array([16,25,-11,-11],dtype=float)
x0=np.zeros(len(b))
max_iter = 2
tol = 10e-6

x = gauss_jacobi_iteraciones(A, b, x0, tol, max_iter)
```

```
Iteración 1 , solución aproximada: [ 1.6    2.5   -1.375 -2.2  ]
Iteración 2 , solución aproximada: [ 1.6    2.5   -1.375 -2.2  ]
```

d.

$$
\begin{aligned}
4x_1 + x_2 + x_3 \quad\quad + x_5 &= 6,\\
-x_1 - 3x_2 + x_3 + x_4 \quad\quad &= 6,\\
2x_1 + x_2 + 5x_3 - x_4 - x_5 &= 6,\\
-x_1 - x_2 - x_3 + 4x_4 \quad\quad &= 6,\\
2x_2 - x_3 + x_4 + 4x_5 &= 6
\end{aligned}
$$

```
%autoreload 2
A = [[4,1,1,0,1],[-1,-3,1,1,0],[2,1,5,-1,-1],[-1,-1,-1,4,0],[0,2,-1,1,4]]
b = np.array([6,6,6,6,6],dtype=float)
x0=np.zeros(len(b))
max_iter = 2
tol = 10e-6

x = gauss_jacobi_iteraciones(A, b, x0, tol, max_iter)
```

```
Iteración 1 , solución aproximada: [ 1.5 -2.   1.2  1.5  1.5]
Iteración 2 , solución aproximada: [ 1.5 -2.   1.2  1.5  1.5]
```

2. Repita el ejercicio 1 usando el método de Gauss-Siedel

```
%autoreload 2
print("\nLiteral a:\n")
A = [[3,-1,1],[3,6,2],[3,3,7]]
b = np.array([1,0,4],dtype=float)
x0=np.zeros(len(b))
max_iter = 2
tol = 10e-6

A = np.array(A, dtype=float)

gauss_seidel_iteraciones(A,b,x0,tol,max_iter)

print("\nLiteral b:\n")
A = [[10,-1,0],[-1,10,-2],[0,-2,10]]
b = np.array([9,7,6],dtype=float)
x0=np.zeros(len(b))
max_iter = 2
tol = 10e-6

A = np.array(A, dtype=float)

gauss_seidel_iteraciones(A,b,x0,tol,max_iter)

print("\nLiteral c:\n")
A = [[10,-5,0,0],[5,10,-4,0],[0,-4,8,-1],[0,0,-1,5]]
b = np.array([16,25,-11,-11],dtype=float)
x0=np.zeros(len(b))
max_iter = 2
tol = 10e-6

A = np.array(A, dtype=float)

gauss_seidel_iteraciones(A,b,x0,tol,max_iter)

print("\nLiteral d:\n")
A = [[4,1,1,0,1],[-1,-3,1,1,0],[2,1,5,-1,-1],[-1,-1,-1,4,0],[0,2,-1,1,4]]
b = np.array([6,6,6,6,6],dtype=float)
x0=np.zeros(len(b))
max_iter = 2
tol = 10e-6

A = np.array(A, dtype=float)

gauss_seidel_iteraciones(A,b,x0,tol,max_iter)
```

```
Literal a:

Iteración 1 , solución aproximada: [ 0.33333333 -0.16666667  0.5       ]
Iteración 2 , solución aproximada: [ 0.33333333 -0.16666667  0.5       ]

Literal b:

Iteración 1 , solución aproximada: [0.9   0.79  0.758]
Iteración 2 , solución aproximada: [0.9   0.79  0.758]

Literal c:
```

```
Iteración 1 , solución aproximada: [ 1.6    1.7   -0.525 -2.305]
Iteración 2 , solución aproximada: [ 1.6    1.7   -0.525 -2.305]

Literal d:

Iteración 1 , solución aproximada: [ 1.5     -2.5      1.1      1.525    2.64375]
Iteración 2 , solución aproximada: [ 1.5     -2.5      1.1      1.525    2.64375]
```

3. Utilice el método de Jacobi para resolver los sistemas lineales en el ejercicio 1, con TOL = 10-3

```
%autoreload 2
print("Literal a:\n")
A = [[3,-1,1],[3,6,2],[3,3,7]]
b = np.array([1,0,4],dtype=float)
x0=np.zeros(len(b))
max_iter = 100
tol = 10e-3

try:
    x = gauss_jacobi(A, b, x0, tol, max_iter)
    print("Se llego a la solución aproximada en el rango deseado con \nx =",x)
except ValueError as e:
    print(e)

print("\nLiteral b:\n")
A = [[10,-1,0],[-1,10,-2],[0,-2,10]]
b = np.array([9,7,6],dtype=float)
x0=np.zeros(len(b))
max_iter = 100
tol = 10e-3

try:
    x = gauss_jacobi(A, b, x0, tol, max_iter)
    print("Se llego a la solución aproximada en el rango deseado con \nx =",x)
except ValueError as e:
    print(e)

print("\nLiteral c:\n")
A = [[10,-5,0,0],[5,10,-4,0],[0,-4,8,-1],[0,0,-1,5]]
b = np.array([16,25,-11,-11],dtype=float)
x0=np.zeros(len(b))
max_iter = 100
tol = 10e-3

try:
    x = gauss_jacobi(A, b, x0, tol, max_iter)
    print("Se llego a la solución aproximada en el rango deseado con \nx =",x)
except ValueError as e:
    print(e)

print("\nLiteral d:\n")
A = [[4,1,1,0,1],[-1,-3,1,1,0],[2,1,5,-1,-1],[-1,-1,-1,4,0],[0,2,-1,1,4]]
b = np.array([6,6,6,6,6],dtype=float)
x0=np.zeros(len(b))
max_iter = 100
tol = 10e-3

try:
    x = gauss_jacobi(A, b, x0, tol, max_iter)
    print("Se llego a la solución aproximada en el rango deseado con \nx =",x)
except ValueError as e:
    print(e)
```

```
Literal a:

Se llego a la solución aproximada en el rango deseado con
x = [ 0.03490444 -0.23975543  0.6547619 ]

Literal b:

Se llego a la solución aproximada en el rango deseado con
x = [0.99555 0.95725 0.7911 ]

Literal c:

Se llego a la solución aproximada en el rango deseado con
x = [ 2.09023438  0.9784625  -1.18959961 -2.4369875 ]
```

```
Literal d:

Se llego a la solución aproximada en el rango deseado con
x = [ 0.7850751  -0.99873844  1.8646296   1.91522095  1.98538479]
```

4. Utilice el método de Gauss-Siedel para resolver los sistemas lineales en el ejercicio 1, con TOL = 10-3.

```
%autoreload 2
print("Literal a:\n")
A = [[3,-1,1],[3,6,2],[3,3,7]]
b = np.array([1,0,4],dtype=float)
x0=np.zeros(len(b))
max_iter = 100
tol = 10e-3

A = np.array(A, dtype=float)

try:
    x = gauss_seidel(A, b, x0, tol, max_iter)
    print("Se llego a la solución aproximada en el rango deseado con \nx =",x)
except ValueError as e:
    print(e)

print("\nLiteral b:\n")
A = [[10,-1,0],[-1,10,-2],[0,-2,10]]
b = np.array([9,7,6],dtype=float)
x0=np.zeros(len(b))
max_iter = 100
tol = 10e-3

A = np.array(A, dtype=float)

try:
    x = gauss_seidel(A, b, x0, tol, max_iter)
    print("Se llego a la solución aproximada en el rango deseado con \nx =",x)
except ValueError as e:
    print(e)

print("\nLiteral c:\n")
A = [[10,-5,0,0],[5,10,-4,0],[0,-4,8,-1],[0,0,-1,5]]
b = np.array([16,25,-11,-11],dtype=float)
x0=np.zeros(len(b))
max_iter = 100
tol = 10e-3

A = np.array(A, dtype=float)

try:
    x = gauss_seidel(A, b, x0, tol, max_iter)
    print("Se llego a la solución aproximada en el rango deseado con \nx =",x)
except ValueError as e:
    print(e)

print("\nLiteral d:\n")
A = [[4,1,1,0,1],[-1,-3,1,1,0],[2,1,5,-1,-1],[-1,-1,-1,4,0],[0,2,-1,1,4]]
b = np.array([6,6,6,6,6],dtype=float)
x0=np.zeros(len(b))
max_iter = 100
tol = 10e-3

A = np.array(A, dtype=float)

try:
    x = gauss_seidel(A, b, x0, tol, max_iter)
    print("Se llego a la solución aproximada en el rango deseado con \nx =",x)
except ValueError as e:
    print(e)
```

```
Literal a:

Se llego a la solución aproximada en el rango deseado con
x = [ 0.0361492  -0.23660752  0.65733928]

Literal b:

Se llego a la solución aproximada en el rango deseado con
x = [0.9957475  0.95787375 0.79157475]
```

```
Literal c:

Se llego a la solución aproximada en el rango deseado con
x = [ 2.08980938  0.97914391 -1.19017501 -2.438035  ]

Literal d:

Se llego a la solución aproximada en el rango deseado con
x = [ 0.78616258 -1.00240703  1.86606999  1.91245638  1.98960692]
```

5. El sistema lineal

$$\begin{aligned} 2x_1 - x_2 + x_3 &= -1, \\ 2x_1 + 2x_2 + 2x_3 &= 4, \\ -x_1 - x_2 + 2x_3 &= -5, \end{aligned}$$

tiene la solución (1,2,-1).

a. Muestre que el método de Jacobi con x(0) = 0 falla al proporcionar una buena aproximación después de 25 iteraciones

```
%autoreload 2
A = [[2,-1,1],[2,2,2,],[-1,-1,2]]
b = [1,4,-5]
x0 = np.zeros(len(b))
tol = 3
max_iter = 25

try:
    x = gauss_jacobi(A, b, x0, tol, max_iter)
    print("La solució obtenida con Gauss-Jacobi es de \nx =",x)
except ValueError as e:
    print(e)
```

```
La solució obtenida con Gauss-Jacobi es de
x = [ 0.5  2.  -2.5]
```

b. Utilice el método de Gauss-Siedel con $x^{(0)} = 0$: para aproximar la solución para el sistema lineal dentro de 10^{-5}.

```
%autoreload 2
tol = 10e-5
A = np.array(A, dtype=float)

try:
    x = gauss_seidel(A, b, x0, tol, max_iter)
    print("La solución obtenida con Gauss-Seidel es de \nx =",x)
except ValueError as e:
    print(e)
```

```
La solución obtenida con Gauss-Seidel es de
x = [ 1.66669655  1.33329964 -1.00000191]
```

6. El sistema lineal

$$\begin{aligned} x_1 \qquad - x_3 &= 0.2, \\ -\frac{1}{2}x_1 + x_2 - \frac{1}{4}x_3 &= -1.425, \\ x_1 - \frac{1}{2}x_2 + x_3 &= 2, \end{aligned}$$

tiene la solución (0.9,-0.8, 0.7).

a. ¿La matriz de coeficientes

$$A = \begin{bmatrix} 1 & 0 & -1 \\ \frac{1}{2} & 1 & -\frac{1}{4} \\ 1 & -\frac{1}{2} & 1 \end{bmatrix}$$

tiene diagonal estrictamente dominante?

```
%autoreload 2
A = np.array([
    [1, 0, -1],
    [1/2,1,-1/4],
    [1,-1/2,1]
], dtype=float)
```

```
        if es_diagonal_estrictamente_dominante(A):
            print("La matriz es estrictamente diagonal dominante.")
        else:
            print("La matriz NO es estrictamente diagonal dominante.")
```

```
La matriz NO es estrictamente diagonal dominante.
```

b. Utilice el método iterativo de Gauss-Siedel para aproximar la solución para el sistema lineal con una tolerancia de 10^{-22} y un máximo de 300 iteraciones.

```
        %autoreload 2
        b = [0.2,-1.425,2]
        x0 = np.zeros(len(b))
        tol = 10e-22
        max_iter = 300

        try:
            x = gauss_seidel(A, b, x0, tol, max_iter)
            print("La solución obtenida con Gauss-Seidel es de \nx =",x)
        except ValueError as e:
            print(e)
```

```
El método de Gauss-Seidel no convergió.
```

c. ¿Qué pasa en la parte b) cuando el sistema cambia por el siguiente?

$$\begin{aligned} x_1 \quad\quad - 2x_3 &= 0.2, \\ -\frac{1}{2}x_1 + x_2 - \frac{1}{4}x_3 &= -1.425, \\ x_1 - \frac{1}{2}x_2 + x_3 &= 2, \end{aligned}$$

```
        %autoreload 2
        A_mod = np.array([
            [1, 0, -2],
            [1/2,1,-1/4],
            [1,-1/2,1]
        ], dtype=float)

        try:
            x = gauss_seidel(A_mod, b, x0, tol, max_iter)
            print("La solución obtenida con Gauss-Seidel es de \nx =",x)
        except ValueError as e:
            print(e)
```

```
El método de Gauss-Seidel no convergió.
```

7. Repita el ejercicio 6 usando el método de Jacobi.

```
        %autoreload 2
        print("\nLiteral b:\n")
        A = np.array([
            [1, 0, -1],
            [1/2,1,-1/4],
            [1,-1/2,1]
        ], dtype=float)
        b = [0.2,-1.425,2]
        x0 = np.zeros(len(b))
        tol = 10e-22
        max_iter = 300

        try:
            x = gauss_jacobi(A, b, x0, tol, max_iter)
            print("La solución obtenida con Gauss-Seidel es de \nx =",x)
        except ValueError as e:
            print(e)

        print("\nLiteral c:\n")
        A_mod = np.array([
            [1, 0, -2],
            [1/2,1,-1/4],
            [1,-1/2,1]
        ], dtype=float)

        try:
            x = gauss_jacobi(A_mod, b, x0, tol, max_iter)
            print("La solución obtenida con Gauss-Seidel es de \nx =",x)
```

```
        except ValueError as e:
            print(e)
```

```
Literal b:

El método de Gauss-Jacobi no convergió.

Literal c:

El método de Gauss-Jacobi no convergió.
```

8. Un cable coaxial está formado por un conductor interno de 0.1 pulgadas cuadradas y un conductor externo de 0.5 pulgadas cuadradas. El potencial en un punto en la sección transversal del cable se describe mediante la ecuación de Laplace. Suponga que el conductor interno se mantiene en 0 volts y el conductor externo se mantiene en 110 volts. Aproximar el potencial entre los dos conductores requiere resolver el siguiente sistema lineal.

$$\begin{bmatrix} 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 4 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 4 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 4 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 4 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 4 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \\ w_{10} \\ w_{11} \\ w_{12} \end{bmatrix} = \begin{bmatrix} 220 \\ 110 \\ 110 \\ 220 \\ 110 \\ 110 \\ 110 \\ 110 \\ 220 \\ 110 \\ 110 \\ 220 \end{bmatrix}.$$

image.png

```
        import numpy as np

        # Matriz de coeficientes A
        A = np.array([
            [4, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0],
            [-1, 4, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, -1, 4, -1, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, -1, 4, 0, -1, 0, 0, 0, 0, 0, 0],
            [-1, 0, 0, 0, 4, 0, -1, 0, 0, 0, 0, 0],
            [0, 0, 0, -1, 0, 4, 0, -1, 0, 0, 0, 0],
            [0, 0, 0, 0, -1, 0, 4, 0, -1, 0, 0, 0],
            [0, 0, 0, 0, 0, -1, 0, 4, 0, 0, 0, -1],
            [0, 0, 0, 0, 0, 0, -1, 0, 4, -1, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, -1, 4, -1, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 4, -1],
            [0, 0, 0, 0, 0, -1, 0, 0, 0, 0, -1, 4]
        ], dtype=float)

        # Vector de términos independientes b
        b = np.array([220, 110, 110, 220, 110, 110, 110, 110,220, 110, 110, 220], dtype=float)
```

a. ¿La matriz es estrictamente diagonalmente dominante?

```
        if es_diagonal_estrictamente_dominante(A):
            print("La matriz es estrictamente diagonal dominante.")
        else:
            print("La matriz NO es estrictamente diagonal dominante.")
```

```
La matriz es estrictamente diagonal dominante.
```

b. Resuelva el sistema lineal usando el método de Jacobi con $x(0) = 0$ y $TOL = 10^{-2}$

```
        x0 = np.zeros(len(b))
        tol = 10e-2
        max_iter = 300

        try:
            x = gauss_jacobi(A, b, x0, tol, max_iter)
            print("La solución obtenida con Gauss-Seidel es de \nx =",x)
```

```
        except ValueError as e:
            print(e)
```

```
La solución obtenida con Gauss-Seidel es de
x = [87.92837143 65.92839241 65.92839241 87.92837143 65.92839241 65.92839241
 65.92839241 65.92839241 87.92837143 65.92839241 65.92839241 87.92837143]
```

c. Repita la parte b) mediante el método de Gauss-Siedel.

```
        try:
            x = gauss_seidel(A, b, x0, tol, max_iter)
            print("La solución obtenida con Gauss-Seidel es de \nx =",x)
        except ValueError as e:
            print(e)
```

```
La solución obtenida con Gauss-Seidel es de
x = [87.98217949 65.98985217 65.99375664 87.99604191 65.98985217 65.9974727
 65.99375664 65.99838442 87.99604191 65.9974727  65.99838442 87.99896428]
```

Link del repositorio:

https://github.com/ElAlfa3007/M-todos_Num-ricos.git