

Foro2_Sor

August 2, 2021

1 Solución al problema planteado por el compañero

```
[1]: import numpy as np
def sor(A,b,x0,w,tol,iteMax):
    delta = 1
    xi = x0
    Q = 1/w*np.diag(np.diag(A)) + np.tril(A,-1)
    for i in range(iteMax):
        r = b - A@xi
        delta = np.linalg.solve(Q,r)
        xi = xi + delta
        if np.linalg.norm(delta) < tol:
            break
    return xi, i
```

```
[5]: n=1000 # número de ecuaciones
A = np.zeros([n,n])
for i in range(n):
    A[i,i] = 7 # valores de la diagonal de la matriz A
    if i < n-1:
        A[i,i+1] = 3
        A[i+1,i] = 3
print('\t Matriz A \n',A,'\n')
b = 28*np.ones([n]) # valores de creación del vector de términos independientes b
b[0] = b[n-1] = 21 # Valores inicial y final de b
#print(b)
```

```
Matriz A
[[7. 3. 0. ... 0. 0. 0.]
 [3. 7. 3. ... 0. 0. 0.]
 [0. 3. 7. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 7. 3. 0.]
 [0. 0. 0. ... 3. 7. 3.]
 [0. 0. 0. ... 0. 3. 7.]]
```

```
[6]: w = 1
for k in range(9):
    x0 = np.zeros_like(b) # x0 = Vector inicial
    w += 0.1
    tol = 1e-4
    iteMax = 500
    [xsor, itsor] = sor(A,b,x0,w,tol,iteMax)
    if itsor != iteMax-1:
        print('\n Con w = ', '{0:.2f}'.format(w), '\t Se
↪ alcanzaron', itsor+1, 'iteraciones para el método Sor', '\n\n\t Las soluciones
↪ X', k+1, '\n\n', xsor.round(decimals=2))
    else:
        print('\n Con w = ', '{0:.2f}'.format(w), '\t Se alcanzaron', itsor+1, '
↪ iteraciones que es el número máximo que se consideró para el método
↪ Sor', '\n\n\t En la búsqueda de Las soluciones X', k+1)
```

Con $w = 1.10$ Se alcanzaron 18 iteraciones para el método Sor

Las soluciones X 1

[illegible]

Las soluciones X 2

[illegible]

[illegible]

[illegible]

[illegible]

Con $w = 1.60$ Se alcanzaron 53 iteraciones para el método Sor

Las soluciones X 6

[illegible]

[illegible]

Con $w = 1.70$ Se alcanzaron 75 iteraciones para el método Sor

Las soluciones X 7

[illegible]

[illegible]

