UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI - UFSJ
DEPARTAMENTO DE ENGENHARIA ELÉTRICA - DEPEL
COORDENAÇÃO DE ENGENHARIA ELÉTRICA - COELE

GABRIEL AUGUSTO SILVA BATISTA

TRABALHO 3

São João del-Rei – MG
Setembro de 2023

Gauss Seidel com relaxação

```python
import numpy as np

def gaussSeidel(A, b, inicial, max_ite, tol, lamb):
    ite = 0
    while ite < max_ite:
        inicial_aux = np.copy(inicial)
        for i in range(len(A)):
            x = b[i]
            for j in range(len(A)):
                if i != j:
                    x -= A[i][j] * inicial[j]
            x /= A[i][i]
            inicial[i] = (1 - lamb) * inicial_aux[i] + lamb * x
        ite += 1
        if np.all(np.abs(inicial - inicial_aux) < tol):
            break
    return inicial, ite
```

Nesse código foi utilizado a biblioteca numpy para facilitar as operações.

Na linha 3 defini uma função para realizar o método de Gauss-Seidel com relaxação, que tem como objetivo resolver sistemas lineares e a relaxação é usado para acelerar ou desacelerar a convergência do método. Quando lambda é igual a 1.0, não há relaxação, e o método é o Gauss-Seidel clássico. Quando lambda é diferente de 1.0, é aplicada a sobrerrelaxação (ou sub-relaxação) para acelerar a convergência. Portanto, a combinação do Método de Gauss-Seidel com a sobrerrelaxação é o que foi implementado no código.

Na linha 5 criei um loop que itera até que o número máximo de iterações seja atingido ou até que a convergência seja alcançada. Em cada iteração, a função calcula uma nova estimativa para as variáveis do sistema linear. O loop continua até que a diferença entre a estimativa atual e a anterior seja menor que a tolerância especificada, indicando convergência.

```python
A = np.array([[4, -3, 1], [2, 4, -2], [4, 3, 3]], dtype=float)
b = np.array([29, -18, 3], dtype=float)

inicial = np.zeros(len(b))
max_ite = 1000
tol = 1e-9
```

Aqui é definido a matriz e os parâmetros usados na função, temos o valor inicial que é uma matriz constituída por zeros, temos também o valor máximo de iterações e a tolerância.

```python
res, ite = gaussSeidel(A, b, inicial, max_ite, tol, 1)
print(f"Sem relaxação:")
print(res)
print("Número de iterações:", ite)
print("\n")

v_lambda = [0.2, 0.5, 1.5]
print(f"Com relaxação:")
for i in v_lambda:
    inicial = np.zeros(len(b))
    res, ite = gaussSeidel(A, b, inicial, max_ite, tol, i)
    print(f"Lambda = {i}:")
    print(res)
    print("Número de iterações:", ite)
    print("\n")
```

Aqui é chamado a função e é passado os parâmetros. Primeiro, o método de Gauss-Seidel é aplicado sem relaxação (usando lambda igual a 1), e os resultados são impressos, incluindo a solução encontrada e o número de iterações necessárias.

Em seguida, o código testa o método com diferentes valores de relaxação (lambda), como 0.2, 0.5 e 1.5. Isso permite ao usuário observar como a convergência é afetada pela variação do fator de relaxação. Cada solução e o número de iterações correspondentes são impressos para análise.

```
Sem relaxação:
[ 3. -5.  2.]
Número de iterações: 67


Com relaxação:
Lambda = 0.2:
[ 3. -5.  2.]
Número de iterações: 143


Lambda = 0.5:
[ 3. -5.  2.]
Número de iterações: 55


Lambda = 1.5:
[-7.71940943e+232  1.00164769e+233 -8.22288102e+231]
Número de iterações: 1000
```

Aqui temos os resultados da execução do programa.

Na execução sem relaxação o algoritmo convergiu em 67 iterações, já quando foi aplicado um fator de relaxação de 0.2 o algoritmo precisou de 143 iterações, ele demorou mais para convergir, com lambda igual a 0.5 o algoritmo convergiu mais rápido, gastando apenas 55 iterações, por fim quando foi utilizado lambda igual 1.5 o algoritmo não convergiu.

Em resumo, o código fornece uma implementação do método de Gauss-Seidel com a capacidade de testar diferentes fatores de relaxação.