Universidade Federal de São Carlos

Campus Sorocaba

Projeto 2 de Cálculo Numérico

Lucas Granja Toniello RA: 726560

Sumário

1) Os métodos propostos foram todos feitos usando a linguagem *python*, na versão 3.4.3 e usando como base os algoritmos passados no livro texto [1]. Para maior facilidade, também foi usada a biblioteca *math* [2] da linguagem, dela foram utilizadas as seguintes funções matemáticas:

- math.pow(a, b): retorna o valor de “a” elevado a “b”.
- math.sqrt(x): retorna a raiz quadrada de x.

Também foi usado um código externo para transposição de matriz [3].

Para a correta execução dos programas é necessário que o usuário coloque os seguintes valores de entrada no começo dele:

- Grau da matriz quadrada.
- Todos os valores de entrada da matriz (denominada A), da linha 1 até a linha n ([1, 1], [1, 2], … [1, n], [2, 1], [2, 2]… [2, n], ... [m, n]).
- Todos os valores do vetor de coeficientes (denominado B) de 1 até n ([1, 1], [1, 2], … [1, n]).

Para Gauss-Jacobi e Gauss-Seidel também são necessários:

- Chute inicial (denominado x0) de 1 até n ([1, 1], [1, 2], … [1, n]).
- Precisão ε.

As imagens com os resultados também possuem as os valores de entrada usados na execução do programa para melhor entendimento e, os códigos enviados por e-mail também estão comentados para melhor entendimento deste.

2) O sistema linear 3x3 utilizado para os seguintes exercícios foi:

$$A=\begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & 2 \\ 0 & 2 & 3 \end{bmatrix} \qquad B=\begin{bmatrix} 3 \\ 5 \\ -4 \end{bmatrix}$$

a) A solução analítica do sistema 3x3, feita pela Eliminação de Gauss:

Eliminação da coluna 1:

$$L2 \rightarrow L2-\left(\frac{-1}{4}\right)L1$$

$$A=\left[\begin{array}{ccc|c} 4 & -1 & 0 & 3 \\ 0 & \frac{15}{4} & 2 & \frac{23}{4} \\ 0 & 2 & 3 & -4 \end{array}\right]$$

Eliminação da coluna 2:

$$L3 \rightarrow L3-\left(\frac{8}{15}\right)L2$$

$$A=\left[\begin{array}{ccc|c} 4 & -1 & 0 & 3 \\ 0 & \frac{15}{4} & 2 & \frac{23}{4} \\ 0 & 0 & \frac{29}{15} & \frac{-106}{15} \end{array}\right]$$

Agora, é possível extrair os valores do vetor x:

$$x3=\frac{\frac{-106}{15}}{\frac{29}{15}}=\frac{-106}{29} \qquad x3=3.655172413793103$$

$$x2=\frac{\frac{23}{4}-2(x3)}{\frac{15}{4}} \rightarrow \frac{\frac{23}{4}-2\left(\frac{-106}{29}\right)}{\frac{15}{4}}=\frac{101}{29} \qquad x2=3.482758620689655$$

$$x1=\frac{3-(-1(x2))-0(x3)}{4} \rightarrow \frac{3-(-1(\frac{101}{29}))}{4}=\frac{47}{29} \qquad x1=1.620689655172414$$

Agora, faremos o cálculo do Fator de Cholesky:

$$g1,1=\sqrt{(a1,1)}\rightarrow\sqrt{(4)}=2$$

$$g2,1=\frac{a2,1}{g1,1}\rightarrow\frac{-1}{2}=\frac{-1}{2}$$

$$g3,1=\frac{a3,1}{g1,1}\rightarrow\frac{0}{2}=0$$

$$g2,2=\sqrt{(a2,2-g^2 2,1)}\rightarrow\sqrt{(4-(\frac{-1}{2})^2)}=\frac{\sqrt{(15)}}{2}$$

$$g3,2=\frac{a3,2-(g3,1-g2,1)}{g2,2}\rightarrow\frac{2-(0(\frac{-1}{2}))}{\frac{\sqrt{(15)}}{2}}=\frac{4\sqrt{(15)}}{15}$$

$$g3,3=\sqrt{(a3,3-g^2 3,1-g^2 3,2)}\rightarrow\sqrt{(3-(0^2)-(\frac{4\sqrt{(15)}}{15})^2)}=\frac{\sqrt{(435)}}{15}$$

Portanto nossa matriz G resultante da Fatoração de Cholesky será:

$$G=\begin{bmatrix} 2 & 0 & 0 \\ \frac{-1}{2} & \frac{\sqrt{(15)}}{2} & 0 \\ 0 & \frac{4\sqrt{(15)}}{15} & \frac{\sqrt{(435)}}{15} \end{bmatrix}$$

A partir dessa, podemos resolver o sistema a partir das equações:

$$Gy=B$$
$$G^T x=y$$

Efetuando os cálculos chegaremos aos mesmos resultados obtidos da Eliminação de Gauss.

b)

Método de Eliminação de Gauss:

```
Digite o grau do sistema: 3
Digite os valores da matriz A:
4 -1 0
-1 4 2
0 2 3
Digite os valores do vetor constante (B):
3 5 -4
Resolução do sistema:
x1 = 1.6206896551831154
x2 = 3.482758620732461
x3 = -3.655172413873365
```

Fatoração de Cholesky

```
Digite o grau do sistema: 3
Digite os valores da matriz A:
4 -1 0
-1 4 2
0 2 3
Digite os valores do vetor constante (B):
3 5 -4

Fator de Cholesky:
[2.0, 0, 0]
[-0.5, 1.9364916731037083, 0]
[0.0, 1.0327955589886446, 1.3904435743076138]

Resolução do sistema:
x1 = 1.6206896551724141
x2 = 3.482758620689657
x3 = -3.6551724137931054
```

Método Iterativo de Gauss-Jacobi

```
Digite o grau do sistema: 3
Digite os valores da matriz A:
4 -1 0
-1 4 2
0 2 3
Digite os valores do vetor constante (B):
3 5 -4
Digite os valores do vetor x0
0.75 1.25 -1.33
Digite a precisão e: 0.00001

Resolução do sistema após 23 iteracoes:
x1 = 1.6206687902015364
x2 = 3.482707027014177
x3 = -3.655116773870764
```

```
Digite o grau do sistema: 3
Digite os valores da matriz A:
4 -1 0
-1 4 2
0 2 3
Digite os valores do vetor constante (B):
3 5 -4
Digite os valores do vetor x0
0.75 1.25 -1.33
Digite a precisão e: 0.00001

Resolução do sistema após 13 iteracoes:
x1 = 1.6206774868261258
x2 = 3.4827393541413656
x3 = -3.655159569427577
```

c) Podemos perceber que as soluções obtidas para todos os métodos foram bem próximas das originais, se diferindo apenas em torno da décima casa decimal para os métodos diretos. Para os métodos iterativos os resultados se diferiram apenas na sexta casa decimal, fruto da precisão escolhida, nestes também percebemos que o método de Gauss-Seidel sempre precisou de menos iterações que o de Gauss-Jacobi, como esperado. Porém, ainda sim foram encontradas dificuldades durante a construção dos algoritmos.

No método da Eliminação de Gauss um problema constante ocorria nas eliminações das colunas durante as operações de subtração. Devido ao sistema de ponto flutuante, frequentemente, os valores se diferenciavam por valores extremamente pequenos, mas que rapidamente se acumulavam e futuramente levavam a problemas, principalmente durante operações de divisão. Para a correção deste problema, o resultado de toda operação de subtração nas linhas era arredondado em 10 casas decimais.

```
Digite o grau do sistema: Digite os valores da matriz A:
Digite os valores do vetor constante (B):
Resolução do sistema:
x1 = 1.6206896551831154
x2 = 3.482758620732461
x3 = -3.655172413873365

Lucas@DESKTOP-8DINTHR:/mnt/c/Users/Lucas/Desktop/jooj/pit
Digite o grau do sistema: Digite os valores da matriz A:
Digite os valores do vetor constante (B):
Resolução do sistema:
x1 = 1.6987179487179487
x2 = 3.7948717948717947
x3 = -4.615384615384615
```

Comparação: Em cima com arredondamento de 10 casas decimais, em baixo sem

Também é preciso notar que o sistema escolhido satisfazia tanto o Critério das Linhas como o de Sassenfield o que fez com que os métodos iterativos. convergissem, outros exemplos em que os critérios não eram satisfeitos foram testados, o que fazia com que os valores rapidamente crescessem para longe da solução.

d) O sistema linear 6x6 utilizado para o seguinte exercício foi:

$$A=\begin{bmatrix} 16 & -4 & 1 & 0 & 2 & 0 \\ -4 & 20 & 2 & 3 & 0 & 3 \\ 1 & 2 & 12 & 2 & 4 & 1 \\ 0 & 3 & 2 & 14 & 1 & 7 \\ 2 & 0 & 4 & 1 & 18 & 2 \\ 0 & 3 & 1 & 7 & 2 & 24 \end{bmatrix} \qquad B=\begin{bmatrix} 8 \\ 10 \\ 8 \\ -6 \\ 14 \\ 4 \end{bmatrix}$$

Método de Eliminação de Gauss

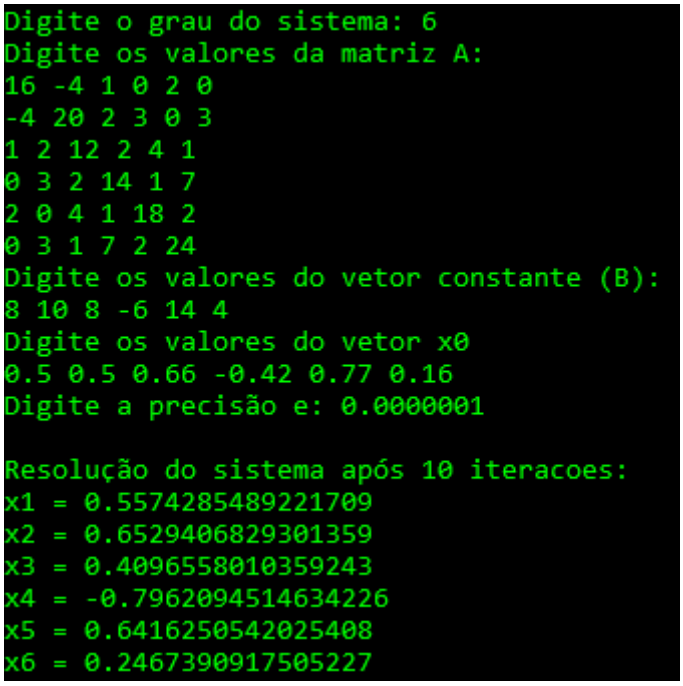
```
Digite o grau do sistema: 6
Digite os valores da matriz A:
16 -4 1 0 2 0
-4 20 2 3 0 3
1 2 12 2 4 1
0 3 2 14 1 7
2 0 4 1 18 2
0 3 1 7 2 24
Digite os valores do vetor constante (B):
8 10 8 -6 14 4
Resolução do sistema:
x1 = 0.5574285515004402
x2 = 0.652940683888454
x3 = 0.40965580745244895
x4 = -0.7962094547140117
x5 = 0.6416250520471622
x6 = 0.24673909248993933
```

Fatoração de Cholesky

```
Digite o grau do sistema: 6
Digite os valores da matriz A:
16 -4 1 0 2 0
-4 20 2 3 0 3
1 2 12 2 4 1
0 3 2 14 1 7
2 0 4 1 18 2
0 3 1 7 2 24
Digite os valores do vetor constante (B):
8 10 8 -6 14 4

Fator de Cholesky:
[4.0, 0, 0, 0, 0, 0]
[-1.0, 4.358898943540673, 0, 0, 0, 0]
[0.25, 0.516185401208764, 3.416292234510822, 0, 0, 0]
[0.0, 0.6882472016116854, 0.4814391536796536, 3.6461667722113185, 0, 0]
[0.5, 0.1147078669352809, 1.1169388365367965, 0.10512808857952027, 4.059339579992121, 0]
[0.0, 0.6882472016116854, 0.1887241482424242, 1.7649923871589608, 0.3756051926550549, 4.4982687333979205]

Resolução do sistema:
x1 = 0.5574285514998081
x2 = 0.6529406838884915
x3 = 0.40965580745559027
x4 = -0.7962094547195102
x5 = 0.6416250520507236
x6 = 0.24673909249225248
```

Método Iterativo de Gauss-Jacobi

```
Digite o grau do sistema: 6
Digite os valores da matriz A:
16 -4 1 0 2 0
-4 20 2 3 0 3
1 2 12 2 4 1
0 3 2 14 1 7
2 0 4 1 18 2
0 3 1 7 2 24
Digite os valores do vetor constante (B):
8 10 8 -6 14 4
Digite os valores do vetor x0
0.5 0.5 0.66 -0.42 0.77 0.16
Digite a precisão e: 0.0000001

Resolução do sistema após 32 iteracoes:
x1 = 0.5574285503345722
x2 = 0.6529406972933558
x3 = 0.4096558260935598
x4 = -0.7962094306014479
x5 = 0.6416250637962416
x6 = 0.246739109472745
```



Novamente, todos os critérios para a aplicação dos métodos foram aplicados que fez com que os resultados todos ficassem bem próximos, se diferindo apenas em torno da décima casa decimal para os métodos diretos e na oitava para os iterativos (por causa da aproximação). Novamente, também tivemos que o Método de Gauss-Seidel foi muito superior ao de Gauss-Jacobi.

Referências

[1] Cálculo Numérico: Aspectos Teóricos e Computacionais, 2ª edição, Márcia A. Gomes Ruggiero e Vera Lúcia da Rocha Lopes
[2] https://docs.python.org/3/library/math.html
[3] https://www.geeksforgeeks.org/transpose-matrix-single-line-python/

Anexos

Método de Eliminação de Gauss, código fonte em python 3.4.3:

```
def eliminacao(A, B):
    n = len(A)

    for k in range(0, n):
        for i in range(k + 1, n):

            m = A[i][k] / A[k][k]
            A[i][k] = 0

            for j in range (k + 1, n):
                A[i][j] = round(A[i][j] - m * A[k][j], 10)

            B[i] = round(B[i] - m * B[k], 10)

def resolucaoSistemaSuperior(A, B):
    x = [0 for i in range(0, len(A))]
    n = len(A) - 1

    x[n] = B[n] / A[n][n]

    for i in range(n - 1, -1, -1):
        soma = 0

        for j in range(i + 1, n + 1):
            soma = soma + A[i][j] * x[j]

        x[i] = (B[i] - soma) / A[i][i]

    return x

def pprint(x):
    i = 1

    for prt in x:
        print("x{} = {}" .format(i, prt))
        i = i+1

    print()

A = []
B = []
```

```
grau = (int)(input("Digite o grau do sistema: "))
print("Digite os valores da matriz A:")

for i in range(0, grau):
    new = list(map(float, input().split()))
    A.append(new)

print("Digite os valores do vetor constante (B):")
B = list(map(float, input().split()))

eliminacao(A, B)
x = resolucaoSistemaSuperior(A, B)
print("Resolução do sistema:")
pprint(x);
```

Fatoração de Cholesky, código fonte em python 3.4.3:

```
import math
import sys

def fatoracaoCholesky(A):
    n = len(A)
    G = [[0 for i in range(0, n)] for i in range(0, n)]

    for k in range(0, n):
        soma = 0

        for j in range(0, k):
            soma = soma + math.pow(G[k][j], 2)

        try:
            G[k][k] = math.sqrt(A[k][k] - soma)
        except:
            print("\n\nERRO: A matriz não é definida positiva\nEncerrando
programa")
            sys.exit()

        for i in range(k, n):
            soma = 0

            for j in range(0, k):
                soma = soma + G[i][j] * G[k][j]
```

```
            G[i][k] = (A[i][k] - soma) / G[k][k]

    return G

def resolucaoSistemaSuperior(A, B):
    x = [0 for i in range(0, len(A))]
    n = len(A) - 1

    x[n] = B[n] / A[n][n]

    for i in range(n - 1, -1, -1):
        soma = 0

        for j in range(i + 1, n + 1):
            soma = soma + A[i][j] * x[j]

        x[i] = (B[i] - soma) / A[i][i]

    return x

def resolucaoSistemaInferior(A, B):
    x = [0 for i in range(0, len(A))]
    n = len(A) - 1

    x[0] = B[0] / A[0][0]

    for i in range(1, n + 1):
        soma = 0

        for j in range(i - 1, -1, -1):
            soma = soma + A[i][j] * x[j]

        x[i] = (B[i] - soma) / A[i][i]

    return x

def transposta(A):
    T = [[A[j][i] for j in range(len(A))] for i in range(len(A[0]))]
    return T

def pprint(x):
    i = 1

    if type(x[0]) == list:
        for prt in x:
```

```
            print(prt)

    else:
        for prt in x:
            print("x{} = {}" .format(i, prt))
            i = i+1

    print()

A = []
B = []

grau = (int)(input("Digite o grau do sistema: "))
print("Digite os valores da matriz A:")

for i in range(0, grau):
    new = list(map(float, input().split()))
    A.append(new)

print("Digite os valores do vetor constante (B):")
B = list(map(float, input().split()))

G = fatoracaoCholesky(A)
print("\nFator de Cholesky: ")
pprint(G)

Y = resolucaoSistemaInferior(G, B)
x = resolucaoSistemaSuperior(transposta(G), Y)

print("Resolução do sistema:")
pprint(x)
```

Método de Gauss-Jacobi

```
def modulo(x):
    if x < 0:
        return (-1) * x
    else:
        return x

def maxModulo(x):
    maior = 0.0000000001

    for i in range(0, len(x)):
        if modulo(x[i]) > maior:
```

```
            maior = modulo(x[i])

    return maior

def distanciaAbsoluta(x0, x1):
    n = len(x0)
    d = [0 for i in range(0, n)]

    for i in range(0, n):
        d[i] = x1[i] - x0[i]

    return maxModulo(d)

def distanciaRelativa(x0, x1):
    return distanciaAbsoluta(x0, x1) / maxModulo(x1)

def iteracao(A, B, x0):
    n = len(A)
    x1 = [0 for i in range(0, n)]

    for i in range(0, n):
        x1[i] = B[i]

        for j in range(0, n):
            if i != j:
                x1[i] = x1[i] - (A[i][j] * x0[j])

        x1[i] = x1[i] / A[i][i]

    return x1

def gaussJacobi(A, B, x0, e):
    x1 = []
    iteracoes = 0

    while True:
        x1 = iteracao(A, B, x0)
        iteracoes = iteracoes + 1

        if distanciaRelativa(x0, x1) < e:
            return x1, iteracoes
        else:
            x0 = x1

def pprint(x):
```

```
    i = 1

    for prt in x:
        print("x{} = {}" .format(i, prt))
        i = i+1

A = []
B = []
x0 = []

grau = (int)(input("Digite o grau do sistema: "))
print("Digite os valores da matriz A:")

for i in range(0, grau):
    new = list(map(float, input().split()))
    A.append(new)

print("Digite os valores do vetor constante (B):")
B = list(map(float, input().split()))

print("Digite os valores do vetor x0")
x0 = list(map(float, input().split()))

e = (float)(input("Digite a precisão e: "))

x, iteracoes = gaussJacobi(A, B, x0, e)
print("\nResolução do sistema após {} iteracoes:" .format(iteracoes))
pprint(x)
```

Método de Gauss-Seidel

```
def modulo(x):
    if x < 0:
        return (-1) * x
    else:
        return x

def maxModulo(x):
    maior = 0.0000000001

    for i in range(0, len(x)):
        if modulo(x[i]) > maior:
            maior = modulo(x[i])

    return maior
```

```
def distanciaAbsoluta(x0, x1):
    n = len(x0)
    d = [0 for i in range(0, n)]

    for i in range(0, n):
        d[i] = x1[i] - x0[i]

    return maxModulo(d)

def distanciaRelativa(x0, x1):
    return distanciaAbsoluta(x0, x1) / maxModulo(x1)

def iteracao(A, B, x0):
    n = len(A)
    x1 = [0 for i in range(0, n)]

    for i in range(0, n):
        x1[i] = B[i]

        for j in range(0, n):
            if j < i:
                x1[i] = x1[i] - (A[i][j] * x1[j])
            elif j > i:
                x1[i] = x1[i] - (A[i][j] * x0[j])
            else:
                pass

        x1[i] = x1[i] / A[i][i]

    return x1

def gaussSeidel(A, B, x0, e):
    x1 = []
    iteracoes = 0

    while True:
        x1 = iteracao(A, B, x0)
        iteracoes = iteracoes + 1

        if distanciaRelativa(x0, x1) < e:
            return x1, iteracoes
        else:
            x0 = x1
```

```
def pprint(x):
    i = 1

    for prt in x:
        print("x{} = {}" .format(i, prt))
        i = i+1

A = []
B = []
x0 = []

grau = (int)(input("Digite o grau do sistema: "))
print("Digite os valores da matriz A:")

for i in range(0, grau):
    new = list(map(float, input().split()))
    A.append(new)

print("Digite os valores do vetor constante (B):")
B = list(map(float, input().split()))

print("Digite os valores do vetor x0")
x0 = list(map(float, input().split()))

e = (float)(input("Digite a precisão e: "))

x, iteracoes = gaussSeidel(A, B, x0, e)
print("\nResolução do sistema após {} iteracoes:" .format(iteracoes))
pprint(x)
```