Laboratório 3 - Sistemas Lineares

Instituto Tecnológico de Aeronáutica - ITA
Matemática Computacional CCI - 22
Nicholas Scharan Cysne
Turma 22.1

1. Introdução

Temos como objetivo deste laboratório a implementação dos Métodos de Gauss-Jacobi e Gauss-Siedel para a aproximações de soluções de sistemas lineares, com posterior teste da questão 1 da Listex 3.

2. Resultados

Os dois algoritmos feitos têm por objetivo resolver qualquer sistema linear de ordem n, dado os coeficientes de tal sistema linear, um vetor inicial de "chute" e a precisão ε desejada. Neste respectivo laboratório desejou-se resolver o seguinte sistema linear.

$$\begin{cases} 3x_1 - 0{,}1x_2 - 0{,}2x_3 = 7{,}85 \\ 0{,}1x_1 + 7x_2 - 0{,}3x_3 = -19{,}3 \\ 0{,}3x_1 - 0{,}2x_2 + 10x_3 = 71{,}4 \end{cases}$$

Iniciou-se os algoritmos com o vetor de chute inicial $x = [0,\ 0,\ 0]$ e $\varepsilon = 0.05$.

O algoritmo de Gauss-Jacobi executou a tarefa em 3 iterações, chegando ao resultado de $x = [3.00080635\ \ -2.49973844\ \ 7.00020667]$ com precisão de $\varepsilon = 1.777963e-05$. A Figura 1 mostra o *output* do algoritmo com os resultados apresentados.

```
Número de iterações: 3
Solução do sistema: [ 3.00080635 -2.49973844  7.00020667]
Precisão da Aproximação: 1.7779637987459788e-05
```

Figura 1. Resultados do Gauss-Jacobi.

O algoritmo de Gauss-Siedel executou a tarefa em 2 iterações, chegando ao resultado de $x = [2.99055651\ \ -2.49962468\ \ 7.00029081]$ com precisão de $\varepsilon = 0.0021278$. A Figura 2 mostra o *output* do algoritmo com os resultados apresentados.

```
Número de iterações: 2
Solução do sistema: [ 2.99055651 -2.49962468  7.00029081]
Precisão da Aproximação: 0.002127804536453643
```

Figura 2. resultados do Gauss-Siedel.

3. Código

A implementação dos algoritmos foi feita utilizando Python3 utilizando a biblioteca *numpy* de representações numéricas, os códigos fonte estão em anexo no Apêndice A.

Cada algoritmo foi feito em seu respectivo arquivo, sendo o Método de Gauss-Jacobi documentado no arquivo “gauss-jacobi.py” e o Método de Gauss-Siedel documentado no arquivo “gauss-siedel”.

Para cada um deles é pedido ao usuário a introdução da seguintes informações sobre o problema:

- Ordem do Sistema Linear N
- Matriz de Coeficientes A
- Vetor de Chute Inicial X
- Vetor de Termos Independentes B
- Precisão ε

Após a introdução dos dados para algoritmo é executado e apresentam-se os resultados finais no terminal.

4. Conclusão

Os dois algoritmos obtiveram com poucas iterações a resolução do sistema pedido. O algoritmo de Gauss-Siedel obteve o resultado em menos iterações porém perdeu precisão, encontrando um valor maior que o algoritmo de Gauss-Jacobi. Contudo os dois algoritmos desempenharam bem se comparado a precisão encontrada com a devida.

O algoritmo de Gauss-Jacobi obteve um erro relativo do vetor de saída, calculando para x_1, x_2, x_3 individualmente, de $ER = [0.0002687 \;\; 0.0001046 \;\; 0.0000295]$, enquanto o algoritmo de Gauss-Siedel obteve $ER = [0.0031478 \;\; 0.0001501 \;\; 0.0000415]$.

APÊNDICE A

Arquivo “gauss-jacobi.py”:

```
#   Laboratório 3 - Sistemas Lineares
#
#   Nicholas Scharan Cysne      T22.1
#   Professor: Johnny
#   Data: 29/04/2019
#
#   Método de Gauss-Jacobi

import numpy as np
import math

# Identificação dos Parâmetros
print("Digite a ordem do sistema linear:")

order = int(input())                                # Ordem do Sistema Linear
A_coef = np.identity(order)                         # Matriz de Coeficientes A
C_coef = np.identity(order)                         # Matriz de Coeficientes C
X_vector = np.zeros(np.shape(A_coef[0]))            # Vetor X
X1_vector = np.zeros(np.shape(A_coef[0]))           # Vetor X1
dx_vector = np.zeros(np.shape(A_coef[0]))           # Vetor dx
b_vector = np.zeros(np.shape(A_coef[0]))            # Vetor b
g_vector = np.zeros(np.shape(A_coef[0]))            # Vetor g

while True:
   print("Digite, por linhas, os coeficientes da matriz de coeficientes:")
   for i in range (order):
       A_coef[i] = input().split()           # Ler N linhas de coeficientes
   print("Matrix de coeficientes:")
   print(A_coef)
   print("A matriz está correta? (S/N)")
   answer = input()
   if answer == "S":
       break

while True:
   print("Digite o vetor X inicial:")
```

```
    X_vector = input().split()                # Vetor X

    if np.size(X_vector) != order:
        print("O tamanho do vetor está errada. Digite Novamente:")
    else:
        for i in range (np.size(X_vector)):
            X_vector[i] = float(X_vector[i])
        print("Vetor inicial X:")
        print(X_vector)
        print("O vetor está correto? (S/N)")
        answer = input()
        if answer == "S":
            break

while True:
    print("Digite o vetor b dos termos independentes:")
    b_vector = input().split()
    if np.size(b_vector) != order:
        print("O tamanho do vetor está errada. Digite Novamente:")
    else:
        for i in range (np.size(b_vector)):
            b_vector[i] = float(b_vector[i])
        print("Vetor de termos independentes b:")
        print(b_vector)
        print("O vetor está correto? (S/N)")
        answer = input()
        if answer == "S":
            break

while True:
    print("Digite a precisão:")
    epsilon = float(input())
    print("Precisão de Aproximação:")
    print(epsilon)
    print("O valor está correto? (S/N)")
    answer = input()
    if answer == "S":
        break

# Aproximações Lineares
counter_k = 1                                 # Contador de iterações k
```

```
for i in range (order):
   for j in range (order):
       if i == j:
           C_coef[i][j] = 0
           g_vector[i] = b_vector[i]/A_coef[i][j]
       else:
           C_coef[i][j] = -A_coef[i][j]/A_coef[i][i]

while True:

   X1_vector = np.dot(C_coef, X_vector) + g_vector
   dx_vector = np.abs(X1_vector - X_vector)
   dr1 = math.fabs(np.sort(X1_vector)[0])
   dr2 = math.fabs(np.sort(dx_vector)[0])

   if dr1 != 0 and (dr2/dr1) < epsilon:
       break

   X_vector = np.copy(X1_vector)
   counter_k += 1

print("Número de iterações: {}".format(counter_k))
print("Solução do sistema: {}".format(X1_vector))
print("Precisão da Aproximação: {}".format(dr2/dr1))
```

Arquivo “gauss-jacobi.py”:

```
#   Laboratório 3 - Sistemas Lineares
#
#   Nicholas Scharan Cysne      T22.1
#   Professor: Johnny
#   Data: 29/04/2019
#
#   Método de Gauss-Siedel

import numpy as np
import math

# Identificação dos Parâmetros

print("Digite a ordem do sistema linear:")

order = int(input())                                # Ordem do Sistema Linear
A_coef = np.identity(order)                         # Matriz de Coeficientes A
C_coef = np.identity(order)                         # Matriz de Coeficientes C
X_vector = np.zeros(np.shape(A_coef[0]))            # Vetor X
X1_vector = np.zeros(np.shape(A_coef[0]))           # Vetor X1
dx_vector = np.zeros(np.shape(A_coef[0]))           # Vetor dx
b_vector = np.zeros(np.shape(A_coef[0]))            # Vetor b
g_vector = np.zeros(np.shape(A_coef[0]))            # Vetor g

while True:
   print("Digite, por linhas, os coeficientes da matriz de coeficientes:")
   for i in range (order):
       A_coef[i] = input().split()           # Ler N linhas de coeficientes

   print("Matrix de coeficientes:")
   print(A_coef)
   print("A matriz está correta? (S/N)")
   answer = input()                          # Resposta dada pelo Usuário
   if answer == "S":
       break

while True:
   print("Digite o vetor X inicial:")
```

```
    X_vector = input().split()                  # Vetor X

    if np.size(X_vector) != order:
        print("O tamanho do vetor está errada. Digite Novamente:")
    else:
        for i in range (np.size(X_vector)):
            X_vector[i] = float(X_vector[i])
        print("Vetor inicial X:")
        print(X_vector)
        print("O vetor está correto? (S/N)")
        answer = input()                        # Resposta dada pelo Usuário

        if answer == "S":
            break

while True:
    print("Digite o vetor b dos termos independentes:")
    b_vector = input().split()
    if np.size(b_vector) != order:
        print("O tamanho do vetor está errada. Digite Novamente:")
    else:
        for i in range (np.size(b_vector)):
            b_vector[i] = float(b_vector[i])
        print("Vetor de termos independentes b:")
        print(b_vector)
        print("O vetor está correto? (S/N)")
        answer = input()
        if answer == "S":
            break

while True:
    print("Digite a precisão:")
    epsilon = float(input())                    # Precisão da Aproximação

    print("Precisão de Aproximação:")
    print(epsilon)
    print("O valor está correto? (S/N)")
    answer = input()                            # Resposta dada pelo Usuário

    if answer == "S":
        break
```

```
# Aproximações Lineares
counter_k = 1                                    # Contador de iterações k

for i in range (order):
   for j in range (order):
       if i == j:
           C_coef[i][j] = 0
           g_vector[i] = b_vector[i]/A_coef[i][j]
       else:
           C_coef[i][j] = -A_coef[i][j]/A_coef[i][i]

while True:

   X1_vector = np.copy(X_vector)
   for i in range (order):
       X1_vector[i] = np.dot(C_coef[i], X1_vector) + g_vector[i]

   dx_vector = np.abs(np.array(X1_vector) - np.array(X_vector))
   dr1 = math.fabs(np.sort(X1_vector)[0])
   dr2 = math.fabs(np.sort(dx_vector)[0])

   if dr1 != 0 and (dr2/dr1) < epsilon:
       break

   X_vector = np.copy(X1_vector)
   counter_k += 1

print("Número de iterações: {}".format(counter_k))
print("Solução do sistema: {}".format(X1_vector))
print("Precisão da Aproximação: {}".format(dr2/dr1))
```