Universidade Federal do Amazonas

Instituto de Ciências Exatas

Departamento de Física

Implementação de 2 algoritmos para Eliminação Gaussiana com pivoteamento parcial na linguagem Python 3.7

Relatório para a disciplina de Cálculo Numérico

por

Aluno: Micael Davi Lima de Oliveira

Orientador: Prof. Dr. José Francisco de Magalhães Neto

Manaus, Outubro de 2019

Micael Davi Lima de Oliveira

Implementação de 2 algoritmos para Eliminação Gaussiana com pivoteamento parcial na linguagem Python 3.7

Relatório como requisito de nota parcial para a disciplina de Cálculo Numérico apresentado ao Curso de Bacharelado em Física na Universidade Federal do Amazonas.

Orientador: Prof. Dr. José Francisco de Magalhães Neto

Manaus, AM

2019

Agradecimentos

Agradeço muito a Deus, por me dar forças e esperança em viver. Ainda que eu não mereça tamanho amor, me fortalece nos dias de tristeza.

Aos meus pais, pelo carinho e apoio.

Ao professor José Reginaldo, cujo conhecimento transmitido foi imprescindível para a realização deste trabalho.

E por fim, aos autores dos algoritmos originais pelo livro: Applied Numerical Methods for Engineers Using MATLAB and C(1999)

Resumo

Neste trabalho será implementado 2 algoritmos para o método de Eliminação Gaussiana, importante para a solução de sistemas lineares. O princípio deste método será em transformar o sistema dado numa matriz triangular superior ou matriz identidade. O algoritmo também irá considerar as situações em que o elemento pivô é nulo, sendo portanto, necessário o tratamento de exceção via pivoteamento parcial da matriz. A linguagem de alto nível adotada e onde os algoritmos foram compilados foi o Python 3 na versão 3.7.5 em arquitetura de 64-bit no sistema operacional Ubuntu 19.10. O programa construído apenas oferece suporte a matrizes quadradas, e portanto, sistemas lineares onde o número de equações é igual ao número de incógnitas. O usuário irá fornecer 3 dados de entrada, o número de equações, a matriz A contendo os coeficientes do sistema de equações, e o vetor B constante, que é associado ao produto interno $Ax = b$. Todo o processamento do cálculo será mostrado ao usuário, para cada iteração será impresso o valor assumido pela matriz A e pelo vetor B. No fim, será apresentado as coordenadas do vetor solução, assim como também, o gráfico de correlação da matriz A por intermédio da biblioteca Matplotlib. O objetivo principal deste relatório consiste em avaliar ambos os algoritmos implementados, buscando entender o porquê da importância do pivoteamento em certas circunstância, a consistência e a eficiência para a resolução de sistemas lineares.

Palavras-chave: Eliminação gaussiana, pivoteamento parcial, sistemas lineares.

Lista de Figuras

Lista de Tabelas

Sumário

Introdução

A eliminação de Gauss foi um dos primeiros métodos para a solução de equações simultâneas. Permanece entre os algoritmos mais importantes, e ainda atualmente é a base para a solução de equações lineares em muitos pacotes de softwares comerciais. É um método numérico para sistemas $Ax = b$, onde assume-se que a matriz A é quadrada $n \times n$, e x e b são ambos vetores n-dimensionais. Durante o processamento, o sistema de equações $Ax = b$ é reduzido mediante a Eliminação de Gauss para uma sistema triangular superior, onde $Ux = y$ podendo ser solucionado via retro-substituição. [1,2]

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = f_1 \tag{1.1}$$

$$\begin{aligned} a_{22}x_2 + a_{22}x_2 + \ldots + a_{2n}x_n &= f_2 \\ &\vdots \end{aligned} \tag{1.2}$$

$$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = f_n \tag{1.3}$$

Desta forma, ao considerar uma matriz $A : m \times n$, como uma função que a cada vetor $x \in \mathbf{R}^n$ associa-se um vetor $b \in \mathbf{R}^m$. Portanto, dado uma matriz A, definimos o conjunto Imagem de A por: [6]

$$Im(A) = \{y \in \mathbf{R}^m | \exists x \in \mathbf{R}^n | y = Ax\} \tag{1.4}$$

E sendo assim, o conjunto $Im(A)$ é um subespaço vetorial do $\mathbf{R}^m$. E desta forma, resolver o sistema linear $Ax = b$ implica em obter os escalares $x_1, x_2, \ldots, x_n$ e que permitem escrever o vetor b de $\mathbf{R}^m$ como combinação linear das n colunas da matriz A. [6]

O princípio da eliminação é subtrair adequados múltiplos da primeira equação das demais de tal maneira a eliminar x_1 das equações posteriores. Mais precisamente, subtraímos da seguinte forma: [4]

$$\frac{a_{i1}}{a_{11}}(a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = f_1) \tag{1.5}$$

da i-ésima equação para cada $i = 2, \ldots, n$. Isso converte o quadro original de expressões para uma matriz triangular superior. A chave para reduzir o produto interno $A \cdot x = b$ para a forma triangular $U \cdot x = k$ deve-se porque sistemas triangulares são menos difíceis de resolver. Em particular, sistemas triangulares superiores podem ser resolvidos pelo algoritmo da retro-substituição. [4]

Lema 1: Suponha que A seja uma matriz triangular superior, além de ser uma matriz com nenhuma entrada nula nas diagonais. Logo, A é invertível, e A^{-1} também será uma matriz triangular superior. [4]

Mediante as operações elementares em uma matriz, poderemos chegar à solução do sistema linear. Por meio de uma matriz estendida, onde $m \times (n+1)$. A linha vertical extra destaca que a última coluna tem um papel especial: [4]

$$M_{ext} = (A|b) = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{12} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right] \tag{1.6}$$

Contudo, um requisito muito importante para o elemento pivô, é de que este seja não nulo. Caso contrário, será necessário o uso de pivoteamento da matriz. É importante atentar que a Eliminação de Gauss Regular apenas pode solucionar sistemas onde o número de equações n é idêntico ao número de incógnitas n. Denomina-se uma matriz regular quando o algoritmo é capaz de reduzi-la a uma matriz triangular superior. [4]

1.1 Descrição do método da Eliminação de Gauss

O método, em sua essência, consiste em transformar o sistema linear original em um sistema equivalente, na forma de matriz triangular superior. Seja o produto interno $A \cdot x = b$ um sistema linear, serão efetuadas uma sequência de operações elementares: [6]

Teorema 1:

i) trocar duas equações;

ii) multiplicar uma equação por uma constante não nula;

iii) adicionar um múltiplo de uma equação a uma outra equação.

Obtém-se assim, um novo sistema $\hat{A} \cdot x = \hat{b}$, que é equivalente ao sistema inicial $A \cdot x = b$. Utiliza-se a notação $a_{ij}^{(k)}$ para denotar o coeficiente da linha i e coluna j no final da k-ésima etapa. Assim como também, denota-se $b_i^{(k)}$ para o i-ésimo elemento do vetor constante ao final da etapa k. [6]

1.2 Pivoteamento e permutações

Até o momento foi considerado apenas as matrizes quadradas singulares. Contudo, um simples caso onde um dos elementos na diagonal principal é nulo, impede o processo em transformar-se numa matriz triangular superior. É impossível realizar operações sobre um pivô nulo. E mesmo adotasse o elemento pivô como tendendo a zero, isto levaria a resultados muito imprecisos, pois calculadores e computadores trabalham com aritmética de precisão finita. E por isso, a estratégia de pivoteamento e tão importante para contornar esta dificuldade. [6]

1.3 Pivoteamento parcial

Este processo consiste em: i) no início da etapa k da fase de eliminação, adotar um elemento pivô de maior módulo entre os coeficientes: $a_{ik}^{(k-1)}, i = k, k+1, \ldots, n$. ii) trocar as linhas k e i caso necessário. [6]

Pode-se perceber que a escolha do maior elemento em módulo durante as operações sob o pivô, consequentemente, faz com que os multiplicadores, em módulo, estejam no intervalo entre zero e um, prevenindo que ocorram ampliações nos erros de aproximação. [6]

Estrutura computacional

Neste capítulo, será tratado as características do ambiente computacional onde os 2 algoritmos foram compilados. Primeiramente, deve-se destacar que a linguagem de programação adotada foi o Python 3 na versão 3.7.3. Sendo portanto, alto-nível, de paradigma imperativo e orientada a objetos. A arquitetura computacional foi de 64 bits executada no sistema operacional com kernel Linux 5.3 sob a distribuição Ubuntu 19.10 x64 com ambiente gráfico Gnome 3.34.1. Adotou-se como IDE(Integrated development environment), o Visual Studio Code na versão 1.39.2.

As operações matriciais, como produto interno, foram auxiliadas pela biblioteca Numpy 1.17.3. Além disso, a plotagem da matriz de correlação foi mediante a biblioteca Matplotlib 3.1.1. Em relação ao hardware do sistema, os programas foram executados num processador Intel Core i3 6300 com clock de 3.8Ghz, com 4 processadores lógicos(threads). Além disso, um dual-channel de 2x Memória-RAM Kingston HyperX 2133Mhz de 4GB. O dispositivo de armazenamento foi um SSD Intel 120GB com velocidade de leitura até 500MB/s e escrita sequencial de até 450MB/s.

Tabela 1: *Parâmetros computacionais onde os algoritmos foram compilados.*

Parâmetro	**Característica**
Sistema Operacional	Ubuntu 19.04 x64
Kernel	Linux 5.3
Linguagem de Programação	Python 3.7.3
Ambiente gráfico do sistema	Gnome 3.34.1
IDE	Visual Studio Code 1.39.2
Biblioteca para matrizes	Numpy 1.17.3
Biblioteca para plotagem	Matplotlib 3.1.1

Desenvolvimento dos algoritmos

3.1 Algoritmo I: Eliminação Gaussiana sem pivoteamento

```
# Aluno: Micael Davi Lima de Oliveira - Bacharelado em Física - 21851626 - FB01
# Professor: José Francisco
# Disciplina: Cálculo Numérico 2019/2

'''
Parte I: Eliminação gaussiana sem pivoteamento

Este algoritmo teve como principal referência o código em MATLAB/Python encontrado no
seguinte site:

Link: https://learnche.org/3E4/Assignment_2_-_2010_-_Solution/Bonus_question

'''

import numpy as np
import matplotlib.pyplot as plt

def elimination(A, b, n):
    """
    Função para realizar a eliminação do elemento pertencente à próxima linha.
    """
    print("\n")
    print("+--------------------+")
    print("+   Processamento    +")
    print("+--------------------+")
```

```
26      print("\n")
27
28      for row in range(0, n-1):
29          for i in range(row+1, n):
30              factor = A[i,row] / A[row,row]
31              for j in range(row, n):
32                  A[i,j] = A[i,j] - factor * A[row,j]
33
34              b[i] = b[i] - factor * b[row]
35
36          print("A" + str(row+1) + "= \n")
37          print('%s' % (A))
38
39          print("\nB" + str(row+1) + "= \n")
40          print('%s' % (b))
41          print("---------------------")
42
43      return A, b
44
45  def substitution(a, b, n):
46      """
47      Função que efetua a substituição do elemento da linha anterior.
48      """
49      x = np.zeros((n,1))
50      x[n-1] = b[n-1] / a[n-1, n-1]
51      for row in range(n-2, -1, -1):
52          sums = b[row]
53          for j in range(row+1, n):
54              sums = sums - a[row,j] * x[j]
55          x[row] = sums / a[row,row]
56      return x
57
58  def gauss(A, b):
59      """
60      Esta função efetua a eliminação gaussiana sem o pivoteamento
61      """
62      n = A.shape[0]
63
64      # Verificação de elementos nulos nas diagonais.
```

```
65      if any(np.diag(A)==0):
66          print("\n Risco de uma divisao por zero, pois nao ha suporte ao
        pivoteamento.")
67
68      A, b = elimination(A, b, n)
69      return substitution(A, b, n)
70
71  # Rotina principal do algoritmo
72  while (True):
73      n_eq = int(input("\n - Digite o numero total de equacoes: "))
74
75      A = np.array(eval(input("\n - Insira a matriz A que representa o
        sistema de equacoes: ")))
76      b = np.array(eval(input("\n - Insira o vetor B: ")))
77
78      x = gauss(A, b)
79
80      print("\n")
81      print("+--------------------+")
82      print("+    Resultados      +")
83      print("+--------------------+")
84      print('\n Coordenadas referentes a solucao do sistema: \n\n%s' %x)
85
86      plt.matshow(A)
87      fig = plt.gcf()
88      fig.canvas.set_window_title('Matriz de Correlação')
89
90      plt.show()
91
92      op = int(input("\n\n Efetuar um novo calculo? \n\n (1) Sim  \n (2)
        Nao \n \n Opcao: "))
93
94      if (op == 2):
95          break
```

Listing 3.1: *Algoritmo I: Neste código não houve a implementação do pivoteamento parcial. Encontra-se apenas o algoritmo para eliminação gaussiana em matrizes singulares. Portanto, este código é incapaz de resolver sistemas não quadrados, o que possuam algum elemento nulo na diagonal principal.*

3.2 Algoritmo II: Eliminação Gaussiana com pivoteamento parcial

```
# Aluno: Micael Davi Lima de Oliveira - Bacharelado em Física - 21851626 - FB01
# Professor: José Francisco
# Disciplina: Cálculo Numérico 2019/2

"""
Parte II: Eliminação de Gauss_Jordan com pivoteamento parcial

Este algoritmo é uma adaptação do Algorithm 2.2.1
encontrado no livro de Métodos Numéricos aplicados
à Engenharia(1999) by Schilling and Harris.

"""

from numpy import*
from copy import*

import matplotlib.pyplot as plt

def GaussJordan(A,b):
    """
    Esta função retorna o vetor "x" presente
    no produto interno A.x=b.

       assumido que "A" é uma matriz quadrada "n x n"
    onde, "m" e "b" constituem os elementos da matriz.
    """

    print("\n")
    print("+--------------------+")
    print("+   Processamento    +")
    print("+--------------------+")
    print("\n")

    n,m = A.shape
    # C constitui uma matriz auxiliar de segurança, e portanto, será importante
```

```
36      # armazenar a matriz inicial e ser modificada com as iteraç es .
37      C = zeros((n,m+1),float)
38      C[:,0:n],C[:,n] = A, b
39
40      for j in range(n):
41          # Primeiro, é efetuado o pivoteamento parcial.
42          p = j # O elemento da diagonal atual será escolhido como piv .
43          # Busca por um piv  alternativo que seja o maior elemento da
        coluna.
44          for i in range(j+1,n):
45              if abs(C[i,j]) > abs(C[p,j]): p = i
46          if abs(C[p,j]) < 1.0e-16:
47              print ("A matriz apresenta a propriedade de singularidade.")
48              return b
49          #  Haverá uma troca de linha, para encontrar o maior elemento da
         diagonal.
50          C[p,:],C[j,:] = copy(C[j,:]),copy(C[p,:])
51          # Agora, haverá a eliminação do termo
52          pivot = C[j,j]
53          C[j,:] = C[j,:] / pivot
54          for i in range(n):
55              if i == j: continue
56              C[i,:] = C[i,:] - C[i,j]*C[j,:]
57          print("A" + str(j+1) + "= \n")
58          print(C[:,0:n])
59
60          print("\nB" + str(j+1) + "= \n")
61          print(C[:,n])
62
63          print("--------------------")
64      I,x = C[:,0:n],C[:,n]
65      return x
66
67  # Rotina principal do programa
68  while (True):
69
70      n_eq = int(input("\n - Digite o numero total de equacoes: "))
71
72      A = array(eval(input("\n - Insira a matriz A que representa o
```

```
    sistema de equacoes: ")))
73   b = array(eval(input("\n - Insira o vetor B: ")))
74
75   x = GaussJordan(A,b)
76
77   print("\n")
78   print("+--------------------+")
79   print("+    Resultados      +")
80   print("+--------------------+")
81   print('\n Coordenadas referentes a solucao do sistema:\n\n')
82
83   print ("x=", x)
84   print ("Ax=", dot(A,x))
85
86   plt.matshow(A)
87   fig = plt.gcf()
88   fig.canvas.set_window_title('Matriz de Correlação')
89
90   plt.show()
91
92   op = int(input("\n\n Efetuar um novo calculo? \n\n (1) Sim  \n (2)
    Nao \n \n Opcao: "))
93
94   if (op == 2):
95       break
```

Listing 3.2: *Algoritmo II: Neste algoritmo foi implementado a função de pivoteamento parcial, abrangendo assim, os casos onde os elementos na diagonal principal são nulos. Contudo, assim como no algoritmo anterior, é incapaz de solucionar matrizes que não do tipo $n \times n$.*

Resultados e Discussões

Percebeu-se que os valores de saída apesar de possuírem um erro associado, ainda sim, em inúmeras vezes foram iguais aos valores reais dos escalares que constituem a solução do sistema de equações. Corroborando que a Eliminação de Gauss é um método direto, ao invés, de um método iterativo. O erro associado aos resultados é devido principalmente à presença, em alguns caso, de dízimas periódicas durante o processamento, acarretando em aproximações que levam a um certo distanciamento do valor real.

Outro ponto notável, foi a observação de que determinados casos de teste $n \times n$ o algoritmo I foi incapaz de solucionar, sendo apontado com o seguinte erro: RuntimeWarning: divide by zero encountered in long_scalars. Desta forma, percebe-se a importância do algoritmo II, onde executou-se uma função de pivoteamento parcial para eliminar o elemento nulo na diagonal principal. O algoritmo II foi capaz de solucionar todos os problemas não resolvidos pelo algoritmo I.

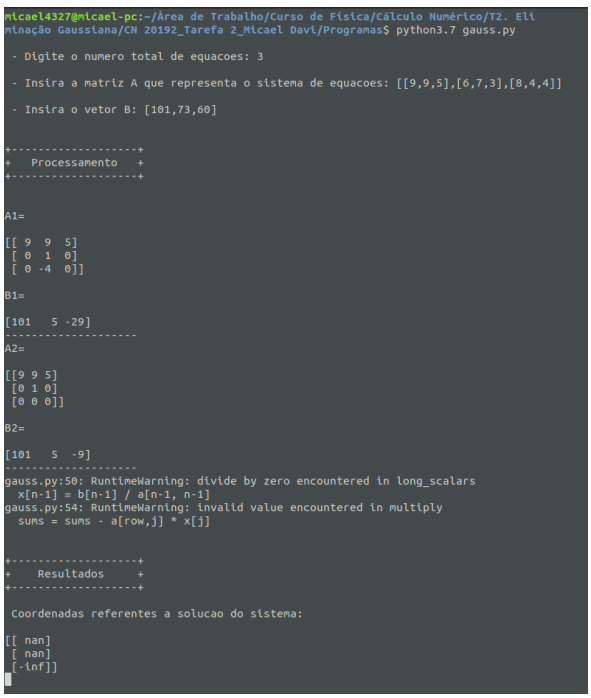

Figura 1: *Algoritmo I compilado no terminal do Ubuntu 19.10.*

Foram fornecido um total de 8 casos de testes. Destes, apenas 3 foram solucionados pelo algoritmo I. Isto porque, grande parte dos casos apresentam sistemas, que apesar de serem $n \times n$, requerem pivoteamento parcial para a resolução. Por outro lado, o algoritmo II foi capaz de solucionar todos os 8 casos propostos, e além disso, observou-se a formação de uma matriz identidade. Por fim, será apresentado alguns resultados obtidos dentre os 8 casos de teste, tanto pelo algoritmo I ou II.

$$M_1 = (A_1|b_1) = \left[\begin{array}{ccc|c} 9 & 9 & 5 & 101 \\ 6 & 7 & 3 & 73 \\ 8 & 4 & 4 & 60 \end{array}\right] \rightarrow x_1 = (*, *, *) \tag{4.1}$$

(Algoritmo I)

$$M_1 = (A_1|b_1) = \left[\begin{array}{ccc|c} 9 & 9 & 5 & 101 \\ 6 & 7 & 3 & 73 \\ 8 & 4 & 4 & 60 \end{array}\right] \rightarrow x_1 = (2, 7, 4) \tag{4.2}$$

(Algoritmo II)

Percebe-se que o algoritmo I foi incapaz de solucionar o primeiro caso de teste, por outro lado, o algoritmo II foi capaz de fornecer uma solução ao sistema. Mostrando a importância de levar em consideração os casos em que o elemento pivô é nulo.

$$M_7 = (A_7|b_7) = \left[\begin{array}{ccccc|c} -1 & 0 & -1 & 0 & -2 & 22 \\ 8 & 9 & -5 & 3 & -1 & 44 \\ -7 & 7 & -6 & -1 & 7 & -14 \\ 8 & 0 & -4 & -7 & 8 & -134 \\ 3 & 9 & 0 & 4 & 1 & 42 \end{array}\right] \rightarrow x_7 = (-3.334, 7.778, 2.889, 1.445, -10.778) \tag{4.3}$$

(Algoritmo I)

$$M_7 = (A_7|b_7) = \left[\begin{array}{ccccc|c} -1 & 0 & -1 & 0 & -2 & 22 \\ 8 & 9 & -5 & 3 & -1 & 44 \\ -7 & 7 & -6 & -1 & 7 & -14 \\ 8 & 0 & -4 & -7 & 8 & -134 \\ 3 & 9 & 0 & 4 & 1 & 42 \end{array}\right] \rightarrow x_7 = (-3.000, 4.000, -1.000, 6.000, -9.000) \quad (4.4)$$

(Algoritmo II)

Percebe-se que ambos os algoritmos foram capazes de solucionar o sistema, embora tenham apresentado resultados diferentes. Ao substituir as coordenadas obtidas pelo algoritmo I no sistema observou-se um erro relativamente grande nos resultados. Apenas o algoritmo II apresentou soluções consistente. Ainda permanece sem explicação o porquê do algoritmo I apresentar tantos erros. É provável que haja algum erro lógico dentro do algoritmo, o que implica nos resultados inconsistentes.

Conclusões

Após a construção dos algoritmos I e II, percebeu-se que apenas o último apresentou resultados consistentes e que fato eram solução do sistema de equações. Dentre os 8 casos de teste, o algoritmo I solucionou apenas 3, e com incoerências na solução proposta. Percebe-se então, que o primeiro algoritmo vem apresentado falhas internas graves. Contudo, o problema lógico ainda permanece desconhecido. Apenas dos resultados falhos para o algoritmo I, o algoritmo com pivoteamento parcial mostrou-se bastante promissor para a solução de sistemas lineares $n \times n$. Observou-se resultante bastante consistentes para o algoritmo II, e com erros relativamente baixos.

Como perspectivas futuras, seria preciso aprimorar os algoritmos para que apresentem erros cada vez menores, tenham menor nível de complexidade, e sejam capazes de solucionar sistemas onde o número de incógnitas difere do número de equações.

Bibliografia

[1] Burden R. L. et al. Numerical Analysis. 9 ed. Brooks/Cole Cengage Learning, 2010.

[2] Chapra, Steven C. Numerical methods for engineers. University of Michigan - Seventh edition.

[3] Gauss elimination without pivoting. Acesso em: 27 de outubro de 2019.

Disponível em: https://learnche.org/3E4/Assignment_2_-_2010_-_Solution/Bonus_question

[4] Introduction to Gaussian Elimination Algorithm. Acesso em: 27 de outubro de 2019.

Disponível em: https://sites.engineering.ucsb.edu/~hpscicom/projects/gauss/introge.pdf

[5] Numerical analysis lecture notes. Acesso em: 28 de outubro de 2019.

Disponível em: http://www-users.math.umn.edu/~olver/num_/lng.pdf

[6] Ruggiero, Márcia A. Cálculo Numérico: Aspectos Teóricos e Computacionais. São Paulo: Pearson Education, 2ª edição, 2000.