INSTITUTO FEDERAL MINAS GERAIS (IFMG) - CAMPUS BAMBUÍ
Cálculo Numérico – Sistemas Lineares – Métodos Iterativos
Prof. Marcos Roberto Ribeiro

Lista de Exercícios 09

Exercício 1:

Considere o código de geração de sistemas lineares mostrado no Apêndice A e faça o seguinte:

- Gere 10 sistemas lineares distintos usando sua matrícula como semente acrescida de um múltiplo de 100.
- Por exemplo, para uma matrícula igual a 5000, podemos usar como semente os números 5000, 5100, 5200, ...
- Salve cada sistema gerado em um arquivo CSV, pode ser usado valor da semente como nome do arquivo.
- A Figura 1 mostra um exemplo de utilização do gerador, usando a semente 10 e salvando o arquivo como 10.csv.
- Para cada sistema gerado, faça o teste de convergência e execute os métodos Jacobi e Gauss-Seidel para resolvê-los.
- Utilize zero como valor inicial para todas as variáveis, tolerância de 1^{-5} e o máximo 50 iterações.
- O Apêndice B mostra como deve ser ser a resposta de execução dos métodos.

Figura 1: Exemplo de utilização do gerador de sistemas lineares

```
python3 gerador.py -p4 -e -s10 -a 10.csv
[[  39.2143    7.4755   -7.0714   -6.6673   -2.      -10.      174.7692]
 [  -2.2224   24.1633    6.        1.        3.4347    6.6673  107.1676]
 [  -5.051     1.4143   25.6184    3.4347    9.4959    3.      -82.5819]
 [  -8.2837    3.       -6.2633   34.6898    2.        7.8796 -146.5613]
 [  -5.8592    4.6469   -3.8388   -7.4755   40.3163   -9.4959 -188.8096]
 [  -9.0918    1.       -7.0714   -1.        5.       32.1633 -198.335 ]]
```

Apêndices

A Gerador de sistemas lineares

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse
import csv
import numpy as np

def gera_sistema_linear(num_var, precisao=None):
    '''Gerador de sistema lineares aleatórios'''
    # Gera lista de números para compor a matriz
    lista_int = np.asarray(range(-10, 11))
    lista_frac = np.linspace(-9.9, 9.9)
    # lista_frac = []
    lista_num = np.concatenate((lista_int, lista_frac))
    # Gera matriz com números aleatórios
    mat_a = np.random.choice(lista_num, (num_var, num_var))
    # Garante dominância diagonal
    for i in range(num_var):
        mat_a[i, i] = np.sum(np.abs(mat_a[i, :])) + 1
    # Gera vetor com a solução do sistema
    vet_x = np.random.choice(lista_num, num_var)
    # Calcula verto de termos independentes
    vet_b = mat_a.dot(vet_x)
    # Monta a matriz estendida
    mat = np.append(mat_a, vet_b[:, None], axis=1)
    if precisao is not None:
        mat = np.round(mat, precisao)
    return mat

def  salva_csv(mat, arq_csv):
    '''Salva matriz em arquivo CSV'''
    with open(arq_csv, 'w', newline='', encoding='utf8') as arq_csv:
        writer = csv.writer(arq_csv)
        for row in mat:
            writer.writerow(row)

def argumentos():
    '''Lê argumentos da linha de comando'''
    parser = argparse.ArgumentParser(
        'Gerador de sistemas lineares')
    parser.add_argument('-v', '--variaveis', type=int,
                        help='Número de variáveis')
    parser.add_argument('-p', '--precisao', type=int,
                        help='Precisão em casas decimais')
    parser.add_argument('-s', '--semente', type=int,
                        help='Semente para geração de números aleatórios')
    parser.add_argument('-e', '--exibe', action='store_true',
                        help='Exibe matriz gerada')
    parser.add_argument('-a', '--arquivo',
                        help='Arquivo CSV para salvar a matriz')
    args = parser.parse_args()
```

```
52      return args
53
54  def principal():
55      '''Função principal'''
56      args = argumentos()
57      # Inicializa semente para geração de números aleatórios
58      if args.semente:
59          np.random.seed(args.semente)
60      # Verifica número de variáveis
61      if not args.variaveis or args.variaveis < 2:
62          args.variaveis = np.random.randint(5, 10)
63      # Gera matriz do sistema linear
64      mat = gera_sistema_linear(args.variaveis, args.precisao)
65      if args.exibe:
66          print(mat)
67      # Salva em arquivo
68      if args.arquivo:
69          salva_csv(mat, args.arquivo)
70
71  if __name__ == '__main__':
72      principal()
```

B Exemplo de resolução usado os métodos de Jacobi e Gauss-Seidel

```
================================================
Matriz estendida
================================================
[[  39.2143    7.4755   -7.0714   -6.6673   -2.      -10.      174.7692]
 [  -2.2224   24.1633    6.        1.        3.4347    6.6673  107.1676]
 [  -5.051     1.4143   25.6184    3.4347    9.4959    3.      -82.5819]
 [  -8.2837    3.       -6.2633   34.6898    2.        7.8796 -146.5613]
 [  -5.8592    4.6469   -3.8388   -7.4755   40.3163   -9.4959 -188.8096]
 [  -9.0918    1.       -7.0714   -1.        5.       32.1633 -198.335 ]]

================================================
Método de Jacobi
================================================
Matriz de iteração:
------------------------------------------------
[[-0.         -0.19063199  0.18032707  0.17002216  0.0510018   0.25500901]
 [ 0.09197419 -0.         -0.24831045 -0.04138508 -0.14214532 -0.27592672]
 [ 0.19716298 -0.05520641 -0.         -0.1340716  -0.37066718 -0.11710333]
 [ 0.23879354 -0.08648075  0.18055163 -0.         -0.05765383 -0.22714458]
 [ 0.1453308  -0.11526107  0.09521707  0.18542128 -0.          0.23553501]
 [ 0.28267622 -0.03109134  0.21985928  0.03109134 -0.15545669 -0.        ]]
------------------------------------------------
Raio espectral: 0.34947212441942566
------------------------------------------------
O sistema convergirá!
------------------------------------------------
Aproximação inicial, x = [0. 0. 0. 0. 0. 0.]
Iteração   0: x=[ 4.45677215  4.43513924 -3.22353855 -4.22491049 -4.68320754 -6.16650033], Erro=+6.16650033
Iteração   1: x=[ 0.50030785  8.18753203  0.43479336 -2.45553944 -7.08945111 -5.15661831], Erro=+3.95646430
Iteração   2: x=[ 0.88031658  6.90539464 -0.01599852 -3.15496956 -7.18267477 -5.1582866 ], Erro=+1.28213739
Iteração   3: x=[ 0.91934438  7.09493958  0.25823137 -3.02898329 -7.15267252 -5.11736855], Erro=+0.27422989
Iteração   4: x=[ 0.96604722  7.00966597  0.22265844 -2.99756709 -7.1097382  -5.05268451], Erro=+0.08527361
Iteração   5: x=[ 0.99991453  6.99754342  0.20887308 -3.00263094 -7.07544867 -5.05035015], Erro=+0.03428953
Iteração   6: x=[ 1.00122275  6.99877275  0.20391526 -2.9984914  -7.07083118 -5.04891858], Erro=+0.00495782
Iteração   7: x=[ 1.00139874  6.99890147  0.20167113 -2.99977186 -7.07015007 -5.05026614], Erro=+0.00224413
Iteração   8: x=[ 1.00044292  6.9998029   0.20177574 -2.99987932 -7.07090783 -5.05085948], Erro=+0.00095582
Iteração   9: x=[ 1.00008172  6.9999649   0.20190229 -2.99998817 -7.07130036 -5.05102024], Erro=+0.00039253
Iteração  10: x=[ 0.99999414  7.00000491  0.20200104 -3.00000644 -7.07141752 -5.05104192], Erro=+0.00011716
Iteração  11: x=[ 0.99998971  6.99999572  0.20202998 -3.0000013  -7.07143395 -5.05102856], Erro=+0.00002894
Iteração  12: x=[ 1.00000012  6.99998657  0.20203346 -2.99999843 -7.07142668 -5.05102045], Erro=+0.00001041
Iteração  13: x=[ 1.00000542  6.99998327  0.20203198 -2.99999679 -7.07142134 -5.0510175 ], Erro=+0.00000534

Solução aproximada encontrada
x = [ 1.00000542  6.99998327  0.20203198 -2.99999679 -7.07142134 -5.0510175 ]

================================================
Método de Gauss-Seidel
================================================
Matriz de iteração:
------------------------------------------------
[[-0.         -0.19063199  0.18032707  0.17002216  0.0510018   0.25500901]
 [-0.         -0.01753322 -0.23172502 -0.02574742 -0.13745447 -0.25247247]
 [-0.         -0.03661762  0.04834653 -0.09912811 -0.35302314 -0.0528869 ]
 [-0.         -0.05061677  0.07182974  0.02492911 -0.09732667 -0.15396488]
 [-0.         -0.03855585  0.07083813  0.02286083 -0.02840498  0.26811182]
 [-0.         -0.0569727   0.06002927  0.02428872 -0.05753505  0.02184027]]
------------------------------------------------
Raio espectral: 0.23775723186719663
------------------------------------------------
O sistema convergirá!
------------------------------------------------
Aproximação inicial, x = [0. 0. 0. 0. 0. 0.]
Iteração   0: x=[ 4.45677215  4.84504726 -2.61230577 -4.05132152 -5.59388397 -4.88800984], Erro=+5.59388397
Iteração   1: x=[ 0.84148252  7.47273792 -0.28113382 -3.28818328 -7.20999533 -5.20417257], Erro=+3.61528963
Iteração   2: x=[ 0.72763551  7.16879044  0.24694759 -3.02874797 -7.16758769 -5.10932704], Erro=+0.52808141
Iteração   3: x=[ 0.95126408  7.01529515  0.23790403 -2.98769397 -7.08830491 -5.05437711], Erro=+0.22362857
Iteração   4: x=[ 1.00393078  6.99425389  0.20812305 -2.99572737 -7.06960817 -5.04853921], Erro=+0.05266670
```

```
Iteração   5: x=[ 1.00364803  6.99768679  0.20134093 -2.99972028 -7.07005606 -5.0502715 ], Erro=+0.00678212
Iteração   6: x=[ 1.00062713  6.99979991  0.20153287 -3.00017044 -7.07121186 -5.05098326], Erro=+0.00302090
Iteração   7: x=[ 0.99994193  7.00006854  0.20195506 -3.00005276 -7.07144802 -5.0510521 ], Erro=+0.00068520
Iteração   8: x=[ 0.99995726  7.00001281  0.20204099 -2.99999951 -7.07143753 -5.05102712], Erro=+0.00008592
Iteração   9: x=[ 0.99999933  6.99998476  0.20203688 -2.99999406 -7.07142168 -5.05101755], Erro=+0.00004208
Iteração  10: x=[ 1.00000811  6.99998147  0.20203106 -2.99999581 -7.07141865 -5.05101677], Erro=+0.00000878

Solução aproximada encontrada
x = [ 1.00000811  6.99998147  0.20203106 -2.99999581 -7.07141865 -5.05101677]
```