Trabalho 04 - Sistemas Lineares

Arthur Vieira Silva

28 de Maio de 2025

Sumário

1 Introdução

A solução de equações lineares é um problema recorrente em várias áreas, como engenheria, matemática, física, entre outras. Dessa forma, há várias técnicas numéricas que podem ser utilizadas para solucionar tais sistemas lineares, sendo assim, dois métodos iterativos serão implementados neste trabalho utilizando a linguagem *Python*: **Método de Jacobi** e **Método de Gauss-Seidel**. Além disso, será utilizado um critério de parada para determinar o quão próxima uma determinada solução $x_i^{(k)}$ está da solução exata, sem conhecermos a solução exata. Esse critério de parada será definido por: $\frac{||x_i^{(k+1)} - x_i^{(k)}||_\infty}{||x_i^{(k+1)}||_\infty} \leq 10^{-1} \Rightarrow i = 1, ..., n$, isto é, a norma infinita da diferença entre a solução atual e a solução anterior, dividida pela norma infinita da solução atual. Também definiu-se um número máximo de iterações igual a 20.

Para alguns casos especiais, os métodos iterativos são mais eficientes do que os métodos diretos na solução de sistemas lineares. Por exemplo, quando a matriz de coeficientes A possui grande quantidade de elementos nulos (matriz esparsa), não há necessidade de operar com os coeficientes nulos, desde que a convergência seja possível. Dessa forma, em um método iterativo, inicia-se com uma solução inicial $x^{(0)}$ (que pode ser ruim) e vamos melhorando essa aproximação através de sucessivas iterações.

Entretanto, é preciso verificar se um determinado sistema irá convergir ou não para a solução final, independentemente da solução inicial escolhida. Para isso, se um sistema for **diagonal dominante**, ou **diagonal estritamente dominante**, a sequência irá convergir para a solução final, tanto usando o Método de Jacobi quanto o Método de Gauss-Seidel. Uma matriz A é **estritamente diagonal dominante** quando $|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, i = 1, ..., n$. Já uma matriz A é **diagonal dominante** quando $|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|, i = 1, ..., n$ e para ao menos um i, a_{ii} é estritamente maior que a soma dos elementos fora da diagonal.

Se um sistema linear possuir diagonal pouco dominante, o processo iterativo poderá convergir oscilando, ou de maneira muito lenta, recomenda-se, nesse caso, a utilização de coeficientes de relaxação, visando reduzir o número de iterações do algoritmo. Geralmente, utiliza-se um coeficiente de relaxação $0 < \lambda < 2$.

Diante desse contexto, ambos os algoritmos implementados foram testados utilizando-se o sistema linear a seguir, no qual fornece uma matriz A de ordem 3. É possível verificar facilmente que o sistema não é **diagonalmente dominante** e nem **diagonal estritamente dominante**. Com isso, foram utilizados os coeficientes de relaxação $\lambda = 0.3$, $\lambda = 0.5$ e $\lambda = 1.5$. Tal sistema de equações, a sua matriz A e o seu vetor b podem ser visualizados a seguir.

$$\begin{cases} 4x_1 - 3x_2 + x_3 = 29 \\ 2x_1 + 4x_2 - 2x_3 = -18 \\ 4x_1 + 3x_2 + 3x_3 = 3 \end{cases}$$

$$A = \begin{bmatrix} 4 & -3 & 1 \\ 2 & 4 & -2 \\ 4 & 3 & 3 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 29 \\ -18 \\ 3 \end{bmatrix}$$

2 Solução numérica dos problemas propostos

2.1 Método de Jacobi

No primeiro método, particionamos a matriz A = L + D + U, onde:

1. D é a diagonal principal de A;
2. L é a matriz triangular inferior estrita, obtida apenas dos elementos infradiagonais de A; e
3. U é a matriz triangular superior estrita, obtida apenas dos elementos supradiagonais de A.

Com isso, considera-se C = D, E = L + U e temos a seguinte equação:

$$X^{(k+1)} = D^{-1}(B - (L + U) \times X^{(k)}) \tag{1}$$

Os elementos de D^{-1} podem ser obtidos de maneira trivial, tomando os recíprocos dos coeficientes da diagonal principal D de A. Então, a Equação 1, na forma desenvolvida, pode ser expressa por:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{j=1,j<i}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1,j>i}^{n} a_{ij}x_j^{(k)}) \tag{2}$$

Dessa maneira, o sistema foi resolvido tomando-se uma solução inicial $x^{(0)} = (0, 0, ..., 0)$ e, em seguida, isolando-se diretamente a i-ésima incógnita x_i na i-ésima equação i. A partir disso, a Equação 2 é aplicada somente com os coeficientes a_{ij} **não nulos** necessários. Com esses passos, o sistema de equações para este exemplo ficaria dessa forma:

$$\begin{cases} x_1 = (29 + 3x_2 - x_3)/4 \\ x_2 = (-18 - 2x_1 + 2x_3)/4 \\ x_3 = (3 - 4x_1 - 3x_2)/3 \end{cases}$$

A seguir, serão apresentadas as saídas obtidas na execução do Método de Jacobi, sem a aplicação de coeficientes de relaxação e com a aplicação de coeficientes de relaxação.

Saída sem a aplicação de coeficiente de relaxação:

Iteração	x^k	**Erro**
1	[7,25, −4,5, 1]	1,0
2	[3,625, −7,625, −4,1667]	0,6776
3	[2,5729, −8,3958, 3,7917]	0,9479
4	[0,0052, −3,8906, 5,9653]	0,7552
5	[2,8407, −1,5200, 4,8837]	0,5806
6	[4,8891, −3,4785, −1,2677]	1,2582

7	[4,9580, −7,5784, −2,0403]	0,5410
8	[2,0763, −7,9992, 1,9677]	0,5010
9	[0,7587, −4,5543, 6,2308]	0,6842
10	[2,2766, −1,7640, 4,5427]	0,6142
11	[4,7914, −3,3669, −0,2715]	1,0048
12	[4,7927, −7,0314, −2,0215]	0,5212
13	[2,4818, −7,9071, 1,6412]	0,4632
14	[0,9094, −4,9203, 5,5980]	0,7068
15	[2,1603, −2,1557, 4,7078]	0,5872
16	[4,4563, −3,2262, 0,2753]	0,9947
17	[4,7615, −6,5905, −1,7155]	0,5105
18	[2,7360, −7,7385, 1,2418]	0,3822
19	[1,1357, −5,2471, 5,0905]	0,7335
20	[2,0421, −2,5226, 4,7329]	0,5757

Saída com a aplicação de coeficiente de relaxação $\lambda = 0.3$:

Iteração	x^k	**Erro**
1	[2,175, −1,35, 0,3]	1,0
2	[3,3713, −2,5763, 0,045]	0,3637
3	[3,9518, −3,6523, −0,2441]	0,2723
4	[4,1378, −4,5360, −0,3559]	0,1948
5	[4,0776, −5,1993, −0,2435]	0,1276
6	[3,87772504, −5,63764959, 0,05831754]	0,07776

Saída com a aplicação de coeficiente de relaxação $\lambda = 0.5$:

Iteração	x^k	**Erro**
1	[3,625, −2,25, 0,5]	1,0
2	[4,5313, −4,1563, −0,5417]	0,4207
3	[4,3997, −5,5964, −0,7135]	0,2573
4	[3,8154, −6,3265, 0,0082]	0,1154
5	[3,1592, −6,3650, 1,1238]	0,1753

6	[2,6773, −5,9414, 2,1382]	0,1707
7	[2,4683, −5,3555, 2,7550]	0,1152
8	[2,5065, −4,8561, 2,9097]	0,1028
9	[2,6935, −4,5772, 2,7119]	0,0609

Saída com a aplicação de coeficiente de relaxação $\lambda = 1.5$:

Iteração	x^k	**Erro**
1	[10,875, −6,75, 1,5]	1,0
2	[−2,7188, −10,4063, −10,875]	1,25
3	[4,6055, −7,6641, 27,9844]	1,3886
4	[−10,5440, 14,6162, −10,2070]	2,6129
5	[36,4178, −13,8054, 5,7671]	1,2895
6	[−25,0277, −22,8354, −53,5111]	1,1483
7	[17,7657, −16,6949, 112,5639]	1,4754
8	[−59,0011, 72,6961, −65,2711]	2,4463
9	[146,6353, −47,8006, 43,0936]	1,4024
10	[−132,3784, −60,5060, −241,6166]	1,1784
11	[99,6011, −58,4257, 477,8241]	1,5057
12	[−283,8385, 306,1301, −348,9758]	2,3692
13	[628,0565, −208,6680, 284,4698]	1,4519
14	[−644,5810, −160,1061, −1083,8459]	1,2625
15	[559,4884, −256,1457, 2072,7440]	1,5229
16	[−1334,3120, 1256,2645, −1769,6303]	2,1713
17	[2754,9400, −961,3709, 1670,5424]	1,4843
18	[−3074,5907, −339,3627, −4901,5948]	1,3408
19	[3004,4853, −1207,3217, 9110,5228]	1,5380
20	[−6266,0506, 5176,4390, −8751,7495]	2,0410

2.2 Método de Gauss-Seidel

Assim como no Método de Jacobi, neste método também temos a partição de A na forma A = L + D + U, mas tomamos C = D + L, E = U e temos a seguinte equação:

$$X^{(k+1)} = (D + L)^{-1}(B - U \times X^{(k)}) \tag{3}$$

sua forma desenvolvida é:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{j=1, j<i}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1, j>i}^{n} a_{ij}x_j^{(k)}) \tag{4}$$

É possível observar que a Equação 4 é muito semelhante à Equação 2, com a exceção de que, no Método de Gauss-Seidel, utilizamos os valores de x_j **mais atualizados** possíveis a cada iteração, e não os valores da iteração anterior, como no Método de Jacobi.

Desse modo o sistema de equações também foi resolvido tomando-se uma solução inicial $x^{(0)} = (0, 0, ..., 0)$ e isolando a i-ésima incógnita x_i na i-ésima equação i, como mostrado anteriormente. Logo, o sistema fica como no primeiro método mas, na segunda equação, podemos usar o valor de x_1 calculado na primeira equação da iteração atual e, na terceira equação, também podemos usar os valores de x_1 e x_2 calculados nas duas primeiras equações da iteração atual.

A seguir, serão apresentadas as saídas obtidas na execução do Método de Gauss-Seidel, sem a aplicação de coeficientes de relaxação e com a aplicação de coeficientes de relaxação.

Saída sem a aplicação de coeficiente de relaxação:

Iteração	x^k	**Erro**
1	[7,25, −8,125, −0,5417]	1,0
2	[1,2917, −5,4167, 4,6944]	1,1
3	[2,0139, −3,1597, 1,4745]	1,0190
4	[4,5116, −6,0185, 1,0031]	0,4750
5	[2,4853, −5,2411, 2,9273]	0,3866
6	[2,5873, −4,3300, 1,8802]	0,2418
7	[3,5325, −5,3261, 1,6162]	0,1870
8	[2,8514, −5,1176, 2,3158]	0,1367
9	[2,8329, −4,7585, 1,9814]	0,0755

Saída com a aplicação de coeficiente de relaxação $\lambda = 0.3$:

Iteração	x^k	**Erro**
1	[2,175, −1,67625, −0,06713]	1,0
2	[3,3254, −3,0323, −0,16746]	0,4078

3	[3,8331, −4,0727, −0,12866]	0,2555
4	[3,9514, −4,8129, 0,07322]	0,1538
5	[3,8526, −5,2859, 0,39598]	0,0895

Saída com a aplicação de coeficiente de relaxação $\lambda = 0.5$:

Iteração	x^k	**Erro**
1	[3,625, −3,15625, −0,33854]	1,0
2	[4,2962, −4,9868, −0,04001]	0,3671
3	[3,9081, −5,7304, 0,73984]	0,1361
4	[3,3376, −5,7647, 1,5272]	0,1366
5	[2,9412, −5,4858, 2,0457]	0,0945

Saída com a aplicação de coeficiente de relaxação $\lambda = 1.5$:

Iteração	x^k	**Erro**
1	[10,875, −14,90625, 2,10938]	1,0
2	[−12,12305, 11,37744, 7,62524]	2,16808
3	[26,87668, −26,87730, −15,75003]	1,45103
4	[−26,89403, 15,04665, 40,59311]	1,38800
5	[26,02708, −3,34880, −65,82752]	1,61666
6	[18,77938, −68,53078, 99,65116]	1,66058
7	[−112,98100, 186,98951, −102,84784]	1,36650
8	[316,29663, −414,60311, 42,23532]	1,45101
9	[−629,54006, 704,38309, 182,88783]	1,58860
10	[1049,49307, −1008,89548, −675,58684]	1,63248
11	[−1395,53388, 1037,65802, 1573,87415]	1,55351
12	[1285,80441, −309,52670, −2892,75584]	1,54407
13	[104,53870, −2099,95755, 4388,73686]	1,65913
14	[−4049,62291, 7371,99860, −5151,62051]	1,29413
15	[12261,04258, −16752,24661, 3183,59502]	1,44006
16	[−26159,77186, 30376,89847, 5163,89851]	1,55148
17	[45328,30976, −45318,50767, −25259,30727]	1,66994

18	[−64164,36079, 51831,29397, 63212,93425]	1,70644
19	[66698,41077, −28536,50438, −122197,03210]	1,51730
20	[−19618,01077, −62672,76381, 194345,18331]	1,62876

3 Análise dos resultados

Após a execução dos algoritmos, é possível afirmar que a utilização de coeficientes de relaxação adequados em ambos os casos foi fundamental para a convergência dos métodos, melhorando tanto o desempenho quanto a estabilidade dos algoritmos.

Sem o uso de coeficiente de relaxação ($\lambda = 1$), o Método de Jacobi atingiu o limite máximo de iterações e oscilou significativamente, apresentando uma convergência lenta e instável. Por outro lado, o Método de Gauss-Seidel apresentou uma convergência rápida, exigindo apenas 9 iterações, se mostrando muito mais eficaz e estável.

Com o uso de coeficiente de relaxação $\lambda = 0.3$, no Método de Jacobi, a convergência aumentou consideravelmente, necessitando-se de apenas 6 iterações para executar, houve um crescimento na estabilidade do método também. Já no Método de Gauss-Seidel, a convergência foi mais rápida do que sem o uso de coeficiente de relaxação porém, não houve tanta diferença, ou seja, a convergência aumentou um pouco e o método continuou estável.

Utilizando um coeficinte de relaxação $\lambda = 0.5$, o Método de Jacobi obteve uma convergência moderada, não tão rápida quanto um $\lambda = 0.3$, foram 9 iterações e o método continuou estável. No Método de Gauss-Seidel, a convergência foi mais rápida do que com um $\lambda = 0.3$ e, desse modo, o erro diminuiu mais rapidamente.

Por fim, com o uso de coeficiente de relaxação $\lambda = 1.5$, o Método de Jacobi atingiu o limite máximo de iterações e divergiu notadamente, apresentando uma baixa estabilidade. Por outro lado, o Método de Gauss-Seidel explodiu rapidamente, com os valores alternando de sinais e crescendo profundamente, mostrando uma clara divergência do método.

Portanto, é possível afirmar que sem a utiliziação de coeficiente de relaxação, o Método de Gauss-Seidel se mostrou mais estável do que o Método de Jacobi. O uso de coeficientes de **sub-relaxação** ($0 < \lambda < 1$) acelerou a convergência dos métodos iterativos e o uso de coeficientes de sobrerrelaxação ($1 < \lambda < 2$) pode acelerar a convergência dos métodos **iterativos lentos** quando a diferença do erro é pequena a cada iteração mas, neste caso, provocou uma divergência e uma consequente elevada instabilidade nos métodos.

4 Referências

JUSTO, Dagoberto Adriano Rizzotto; SAUTER, Esequia; AZEVEDO, Fabio Souto de; GUIDI, Leonardo Fernandes; KONZEN, Pedro Henrique de Almeida (org.). Cálculo numérico: Um livro colaborativo versão Python. Porto Alegre: 2020. Disponível em: https://www.ufrgs.br/reamat/CalculoNumerico/livro-py/livro-py.pdf. Acesso em: 24 jan. 2023.

PETERS, Sérgio; SZEREMETA, Júlio Felipe. Cálculo numérico computacional. Florianópolis: Editora UFSC, 2018. 526 p. (Série Didática). Disponível em: https://sergiopeters.prof.ufsc.br/livro-calculo-numerico-computacional/. Acesso em: 27 maio 2025.