UNIVERSIDADE ESTADUAL DE SANTA CRUZ

Mestrado em Modelagem Computacional

Projeto 1

MÉTODOS NUMÉRICOS I

Professor: Dany S. Dominguez

Alunos: Everaldina Guimarães Barbosa
João Vitor Nascimento Ramos

Ilhéus – BA

2025.2

Sumário

Introdução

Como forma de entender melhor o comportamento de métodos numéricos aplicados a sistemas lineares, este trabalho propõe a análise de situações onde questões como mau condicionamento e erros de arredondamento tornam-se evidentes. A aplicação prática de diferentes técnicas permite não apenas encontrar soluções, mas também observar os limites de cada abordagem diante de problemas clássicos da álgebra linear numérica.

O propósito central não é apenas chegar ao resultado final, mas refletir sobre como a escolha do método influencia a precisão obtida, o número de iterações necessárias e o custo computacional envolvido. Dessa maneira, os exercícios oferecem uma oportunidade de comparar métodos, discutir suas vantagens e limitações, e desenvolver uma visão crítica sobre o uso adequado de cada procedimento.

Configuração do Ambiente de Execução

Todos os experimentos apresentados foram realizados em uma mesma máquina, garantindo consistência nos resultados. A configuração de hardware utilizada foi a seguinte:

- **Processador (CPU):** AMD Ryzen 5 5600, com 6 núcleos físicos e 12 threads lógicas.
- **Memória RAM:** 32 GB de memória física instalada.
- **Placa de Vídeo (GPU):** AMD Radeon RX 6750 XT, com 12 GB de memória dedicada.
- **Sistema Operacional:** Windows 64 bits.

Essa configuração oferece recursos suficientes para a manipulação de matrizes de ordem elevada, de modo que as diferenças de desempenho observadas refletem principalmente o comportamento dos algoritmos e não limitações do ambiente de execução.

Exercício 1: Matriz de Hilbert

Considere a matriz de Hilbert H, de ordem n, composta pelos elementos

$$h_{ij} = \frac{1}{i+j-1}, \quad i = 1, \ldots, n,\ j = 1, \ldots, n.$$

O sistema a ser resolvido é dado por

$$Hx = b,$$

em que o vetor independente b possui elementos definidos como a soma de todos os elementos da i-ésima linha da matriz de Hilbert, isto é,

$$b_i = \sum_{j=1}^{n} h_{ij}.$$

Nessas condições, o sistema linear assume a forma

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 + \frac{1}{2} + \cdots + \frac{1}{n} \\ \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n+1} \\ \vdots \\ \frac{1}{n} + \frac{1}{n+1} + \cdots + \frac{1}{2n-1} \end{bmatrix}.$$

Métodos aplicados. O sistema será resolvido para $n = 5$, $n = 9$ e $n = 15$ por meio de:

- **Métodos diretos:** Eliminação de Gauss sem pivotamento, com pivotamento parcial, com pivotamento parcial escalonado e com pivotamento total.
- **Métodos iterativos:** Jacobi, Gauss–Seidel e Sobre-relaxamento (SOR).

Em cada caso, serão reportados os valores de x obtidos, o erro relativo por componente, o tempo de execução e, para os métodos iterativos, o número de iterações até a convergência.

Chute inicial. Para todos os métodos iterativos, adotou-se como aproximação inicial

$$x_i^{(0)} = \frac{b_i}{a_{ii}}, \qquad i = 1, 2, \ldots, n,$$

isto é, o quociente entre o termo independente e o elemento da diagonal principal correspondente. Essa escolha é simples, garante consistência dimensional e fornece um ponto de partida razoável sem custo adicional de cálculo.

Critério de erro. A solução exata do sistema é o vetor $x^\star = (1, 1, \ldots, 1)^T$. Assim, o erro relativo em cada componente será calculado como

$$\varepsilon_i = \frac{|x_i - 1|}{|1|} \times 100\%,$$

além da norma do erro global, quando conveniente. Para os métodos iterativos, a convergência será verificada pelo critério da variação relativa entre normas máximas consecutivas de x:

$$\delta^{(k)} = \frac{\left| \|x^{(k)}\|_\infty - \|x^{(k-1)}\|_\infty \right|}{\max\{\|x^{(k)}\|_\infty, \|x^{(k-1)}\|_\infty, \varepsilon_{\min}\}} < \tau,$$

sendo adotada a tolerância $\tau = 10^{-12}$ em todos os experimentos com métodos iterativos. Esse valor foi escolhido por estar alguns dígitos acima do limite de arredondamento da aritmética em dupla precisão, equilibrando custo computacional e confiabilidade numérica.

Precisão numérica. Todos os cálculos foram realizados em aritmética de ponto flutuante em dupla precisão (`double`), a qual fornece aproximadamente 15 a 16 dígitos significativos. Esse nível de precisão é suficiente para evidenciar os efeitos do mau condicionamento da matriz de Hilbert nas ordens analisadas, permitindo identificar discrepâncias sutis da ordem de 10^{-13} a 10^{-15} entre a solução obtida e o vetor exato $x^\star$.

Resolução pelo Método de Gauss (sem pivotamento)

Resultados para $n = 5$

A solução numérica obtida mostrou-se praticamente idêntica ao vetor exato $x^\star = (1, 1, 1, 1, 1)^T$, com discrepâncias apenas da ordem de 10^{-15} a 10^{-13}, compatíveis com os limites de arredondamento em aritmética de dupla precisão.

A Tabela 1 apresenta os valores de cada componente da solução e os respectivos erros relativos (em porcentagem). O erro relativo médio foi de aproximadamente $2.19 \times 10^{-11}\%$, enquanto o máximo observado atingiu $4.29 \times 10^{-11}\%$. O tempo de execução foi de 5.0×10^{-7} s (500 ns).

i	x_i obtido	Erro relativo ε_i [%]
0	$9.9999999999998690 \times 10^{-1}$	1.31×10^{-12}
1	$1.0000000000001554 \times 10^{0}$	1.55×10^{-11}
2	$9.9999999999957079 \times 10^{-1}$	4.29×10^{-11}
3	$1.0000000000003944 \times 10^{0}$	3.94×10^{-11}
4	$9.9999999999989886 \times 10^{-1}$	1.01×10^{-11}

Tabela 1: Solução numérica e erros relativos em porcentagem para $n = 5$.

Resultados para $n = 9$

A solução numérica obtida permaneceu próxima ao vetor exato, mas com erros já bem mais significativos do que no caso $n = 5$, reflexo do mau condicionamento da matriz de Hilbert à medida que a ordem cresce.

A Tabela 2 apresenta os valores de cada componente da solução e os respectivos erros relativos (em porcentagem). O erro relativo médio foi de aproximadamente $8.22 \times 10^{-4}\%$, enquanto o máximo atingiu $2.32 \times 10^{-3}\%$. O tempo de execução foi de 1.4×10^{-6} s (1400 ns).

i	x_i obtido	Erro relativo ε_i [%]
0	$9.9999999967399722 \times 10^{-1}$	3.26×10^{-8}
1	$1.0000000224419550 \times 10^{0}$	2.24×10^{-6}
2	$9.9999962073490112 \times 10^{-1}$	3.79×10^{-5}
3	$1.0000027046632298 \times 10^{0}$	2.70×10^{-4}
4	$9.9999008688084168 \times 10^{-1}$	9.91×10^{-4}
5	$1.0000202295720209 \times 10^{0}$	2.02×10^{-3}
6	$9.9997677724112555 \times 10^{-1}$	2.32×10^{-3}
7	$1.0000140221804308 \times 10^{0}$	1.40×10^{-3}
8	$9.9999653638456354 \times 10^{-1}$	3.46×10^{-4}

Tabela 2: Solução numérica e erros relativos em porcentagem para $n = 9$.

Resultados para $n = 15$

Para ordem 15, a eliminação de Gauss sem pivotamento produziu uma solução altamente instável, refletindo o severo mau condicionamento da matriz de Hilbert. Alguns valores permaneceram próximos de 1, enquanto outros apresentaram desvios enormes, resultando em erros relativos que variaram de $10^{-6}\%$ até mais de $10^{3}\%$.

A Tabela 3 apresenta os valores de cada componente da solução e os respectivos erros relativos (em porcentagem). O erro relativo médio foi de aproximadamente $3.31 \times 10^{2}\%$, enquanto o máximo atingiu $1.38 \times 10^{3}\%$. O tempo de execução foi de 4.4×10^{-6} s (4400 ns).

i	x_i obtido	Erro relativo ε_i [%]
0	$1.0000000133972358 \times 10^{0}$	1.34×10^{-6}
1	$9.9999527664194532 \times 10^{-1}$	4.72×10^{-4}
2	$1.0002800795425326 \times 10^{0}$	2.80×10^{-2}
3	$9.9365286763951810 \times 10^{-1}$	6.35×10^{-1}
4	$1.0733069825070010 \times 10^{0}$	7.33×10^{0}
5	$5.0374915376517593 \times 10^{-1}$	4.96×10^{1}
6	$3.1242826465544855 \times 10^{0}$	2.12×10^{2}
7	$-4.9837719250889023 \times 10^{0}$	5.98×10^{2}
8	$1.2236498759119220 \times 10^{1}$	1.12×10^{3}
9	$-1.2841984428433998 \times 10^{1}$	1.38×10^{3}
10	$1.1449447288014076 \times 10^{1}$	1.04×10^{3}
11	$-2.7540836869593579 \times 10^{0}$	3.75×10^{2}
12	$5.1392927792281362 \times 10^{-1}$	4.86×10^{1}
13	$1.9193952892354285 \times 10^{0}$	9.19×10^{1}
14	$7.6530232734679571 \times 10^{-1}$	2.35×10^{1}

Tabela 3: Solução numérica e erros relativos em porcentagem para $n = 15$.

Conclusão do método

- Para $n = 5$, os erros permaneceram próximos ao limite de arredondamento em dupla precisão, indicando estabilidade confiável.
- Em $n = 9$, houve aumento expressivo: os erros cresceram cerca de 10^{7} vezes em relação ao caso anterior, embora o custo computacional ainda se mantivesse baixo.
- No caso $n = 15$, a instabilidade numérica dominou, com erros da ordem de centenas a milhares de por cento, evidenciando falha completa do método sem pivotamento.

Esses resultados evidenciam que, conforme cresce a ordem da matriz de Hilbert, a eliminação de Gauss sem pivotamento torna-se inviável, exigindo técnicas de pivotamento ou métodos alternativos para garantir precisão.

Resolução pelo Método de Gauss (pivotamento parcial)

Resultados para $n = 5$

Os valores obtidos permaneceram extremamente próximos de $x^\star$, com erros relativos na faixa de $10^{-13}\%$ a $10^{-10}\%$, compatíveis com arredondamento em dupla precisão. O tempo de execução foi de 8.0×10^{-7} s (800 ns).

A média dos erros relativos ficou em $3.46 \times 10^{-11}\%$ e o erro máximo em $8.24 \times 10^{-11}\%$.

i	x_i obtido	Erro relativo ε_i [%]
0	$9.9999999999999667 \times 10^{-1}$	3.33×10^{-13}
1	$9.9999999999996902 \times 10^{-1}$	3.10×10^{-12}
2	$1.0000000000003768 \times 10^{0}$	3.77×10^{-11}
3	$9.9999999999917566 \times 10^{-1}$	8.24×10^{-11}
4	$1.0000000000004956 \times 10^{0}$	4.96×10^{-11}

Tabela 4: Solução numérica e erros relativos (em porcentagem) para $n = 5$ com pivotamento parcial.

Resultados para $n = 9$

Os valores obtidos permaneceram próximos de $x^\star$, mas com erros crescentes em relação ao caso $n = 5$, reflexo do mau condicionamento da matriz de Hilbert. O tempo de execução foi de 2.8×10^{-6} s (2800 ns). A média dos erros relativos ficou em $7.09 \times 10^{-4}\%$ e o erro máximo em $2.00 \times 10^{-3}\%$.

i	x_i obtido	Erro relativo ε_i [%]
0	$9.9999999971818143 \times 10^{-1}$	2.82×10^{-8}
1	$1.0000000193907881 \times 10^{0}$	1.94×10^{-6}
2	$9.9999967244145671 \times 10^{-1}$	3.28×10^{-5}
3	$1.0000023350590712 \times 10^{0}$	2.34×10^{-4}
4	$9.9999144425854591 \times 10^{-1}$	8.56×10^{-4}
5	$1.0000174548698531 \times 10^{0}$	1.75×10^{-3}
6	$9.9997996715421056 \times 10^{-1}$	2.00×10^{-3}
7	$1.0000120936294825 \times 10^{0}$	1.21×10^{-3}
8	$9.9999701328322177 \times 10^{-1}$	2.99×10^{-4}

Tabela 5: Solução numérica e erros relativos (em porcentagem) para $n = 9$ com pivotamento parcial.

Resultados para $n = 15$

Para $n = 15$, mesmo com o uso de pivotamento parcial, a solução apresentou grande instabilidade numérica. A média dos erros relativos foi de $4.11 \times 10^2\%$ e o erro máximo atingiu $1.65 \times 10^3\%$. O tempo de execução foi de 7.5×10^{-6} s (7500 ns).

i	x_i obtido	Erro relativo ε_i [%]
0	$9.9999997298431254 \times 10^{-1}$	2.70×10^{-6}
1	$1.0000010150941292 \times 10^{0}$	1.02×10^{-4}
2	$1.0000797046016587 \times 10^{0}$	7.97×10^{-3}
3	$9.9663856951793361 \times 10^{-1}$	3.36×10^{-1}
4	$1.0500470522612544 \times 10^{0}$	5.00×10^{0}
5	$6.0586445492586971 \times 10^{-1}$	3.94×10^{1}
6	$2.8828788166461714 \times 10^{0}$	1.88×10^{2}
7	$-4.8212306321291996 \times 10^{0}$	5.82×10^{2}
8	$1.2991298092682106 \times 10^{1}$	1.20×10^{3}
9	$-1.5505491514445122 \times 10^{1}$	1.65×10^{3}
10	$1.5781504975594494 \times 10^{1}$	1.48×10^{3}
11	$-6.9497365429970346 \times 10^{0}$	7.95×10^{2}
12	$2.9866145680685849 \times 10^{0}$	1.99×10^{2}
13	$1.0979851209169875 \times 10^{0}$	9.80×10^{0}
14	$8.8354625170659218 \times 10^{-1}$	1.16×10^{1}

Tabela 6: Solução numérica e erros relativos (em porcentagem) para $n = 15$ com pivotamento parcial.

Conclusão do método

- Para $n = 5$, o pivotamento parcial manteve erros próximos ao limite de precisão de máquina, mas com custo quase 2 vezes maior, sem ganhos relevantes de estabilidade.
- Ao aumentar para $n = 9$, houve leve melhora de precisão (redução em torno de 15%), porém acompanhada de tempo de execução cerca de duas vezes maior.
- Com $n = 15$, a instabilidade não foi controlada: os erros cresceram ainda mais em relação ao caso sem pivotamento, enquanto o tempo quase dobrou.

Em síntese, o pivotamento parcial traz pequenas vantagens apenas em ordens intermediárias, mas em geral aumenta o custo computacional sem resolver o problema do mau condicionamento em matrizes maiores.

Resolução pelo Método de Gauss (pivotamento parcial com pesos)

Resultados para $n = 5$

Os valores obtidos permaneceram extremamente próximos de $x^\star$, com erros relativos na faixa de $10^{-13}\%$ a $10^{-10}\%$, compatíveis com arredondamento em dupla precisão. O tempo de execução foi de 1.2×10^{-6} s (1200 ns). A média dos erros relativos ficou em $4.27\times10^{-11}\%$ e o erro máximo em $1.00 \times 10^{-10}\%$.

i	x_i obtido	Erro relativo ε_i [%]
0	$9.9999999999999800 \times 10^{-1}$	1.998401×10^{-13}
1	$9.9999999999994427 \times 10^{-1}$	5.573320×10^{-12}
2	$1.0000000000004905 \times 10^{0}$	4.904965×10^{-11}
3	$9.9999999999899647 \times 10^{-1}$	1.003531×10^{-10}
4	$1.0000000000005858 \times 10^{0}$	5.857537×10^{-11}

Tabela 7: Solução numérica e erros relativos (em porcentagem) para $n = 5$ com pivotamento parcial por pesos.

Resultados para $n = 9$

Os valores obtidos permaneceram próximos de $x^\star$, mas com erros numéricos mais significativos em relação a $n = 5$, refletindo o aumento do mau condicionamento da matriz de Hilbert. O tempo de execução foi de 2.7×10^{-6} s (2700 ns). A média dos erros relativos ficou em $7.01 \times 10^{-4}\%$ e o erro máximo em $1.98 \times 10^{-3}\%$.

i	x_i obtido	Erro relativo ε_i [%]
0	$9.9999999972251241 \times 10^{-1}$	2.774876×10^{-8}
1	$1.0000000191191725 \times 10^{0}$	1.911917×10^{-6}
2	$9.9999967673456192 \times 10^{-1}$	3.232654×10^{-5}
3	$1.0000023059524632 \times 10^{0}$	2.305952×10^{-4}
4	$9.9999154680452618 \times 10^{-1}$	8.453195×10^{-4}
5	$1.0000172521477817 \times 10^{0}$	1.725215×10^{-3}
6	$9.9998019388446557 \times 10^{-1}$	1.980612×10^{-3}
7	$1.0000119596736752 \times 10^{0}$	1.195967×10^{-3}
8	$9.9999704576736548 \times 10^{-1}$	2.954233×10^{-4}

Tabela 8: Solução numérica e erros relativos (em porcentagem) para $n = 9$ com pivotamento parcial por pesos.

Resultados para $n = 15$

Para $n = 15$, mesmo com o uso de pivotamento parcial por pesos, a solução apresentou forte instabilidade numérica. Os erros relativos variaram desde a ordem de $10^{-6}\%$ até valores acima de $10^{3}\%$. A média dos erros ficou em $4.05 \times 10^{2}\%$ e o erro máximo atingiu $1.67 \times 10^{3}\%$. O tempo de execução foi de 6.6×10^{-6} s (6600 ns).

i	x_i obtido	Erro relativo ε_i [%]
0	$1.0000000319025486 \times 10^{0}$	3.190255×10^{-6}
1	$9.9999178657632382 \times 10^{-1}$	8.213424×10^{-4}
2	$1.0004382114561177 \times 10^{0}$	4.382115×10^{-2}
3	$9.9060361652258366 \times 10^{-1}$	9.396383×10^{-1}
4	$1.1046620400417131 \times 10^{0}$	1.046620×10^{1}
5	$3.1109588331439841 \times 10^{-1}$	6.889041×10^{1}
6	$3.8751647525238533 \times 10^{0}$	2.875165×10^{2}
7	$-6.8844554607896482 \times 10^{0}$	7.884455×10^{2}
8	$1.5313582838135011 \times 10^{1}$	1.431358×10^{3}
9	$-1.5732614719679260 \times 10^{1}$	1.673261×10^{3}
10	$1.2313502035114496 \times 10^{1}$	1.131350×10^{3}
11	$-1.4790718072535556 \times 10^{0}$	2.479072×10^{2}
12	$-1.1738392490124845 \times 10^{0}$	2.173839×10^{2}
13	$2.7540757666885392 \times 10^{0}$	1.754076×10^{2}
14	$6.0686416684201472 \times 10^{-1}$	3.931358×10^{1}

Tabela 9: Solução numérica e erros relativos (em porcentagem) para $n = 15$ com pivotamento parcial por pesos.

Conclusão do método

- Para $n = 5$, manteve erros no limite de precisão de máquina, assim como os demais métodos, mas o tempo foi 2,4 vezes maior que o sem pivotamento e 1,5 vez maior que o parcial, sem ganhos práticos.
- Ao aumentar para $n = 9$, trouxe apenas melhora marginal de precisão (cerca de 1% em relação ao parcial e ao sem pivotamento), ao custo de 1,9 vez o tempo do sem pivotamento.
- Em $n = 15$, não conteve a instabilidade: apresentou erros semelhantes ao parcial, mas até superiores ao sem pivotamento, com tempo 1,5 vez maior que este e próximo ao parcial.

Em síntese, o pivotamento por escala pouco acrescenta em estabilidade: para ordens pequenas apenas aumenta o custo (até 2,4 vezes), em ordens intermediárias traz ganhos mínimos, e em ordens maiores pode até gerar resultados piores que o sem pivotamento.

Resolução pelo Método de Gauss (pivotamento total)

Resultados para $n = 5$

Os valores obtidos permaneceram extremamente próximos de $x^\star$, com erros relativos na faixa de $10^{-14}\%$ a $10^{-10}\%$, compatíveis com arredondamento em dupla precisão. O tempo de execução foi de 1.7×10^{-6} s (1700 ns). A média dos erros relativos ficou em $5.13\times10^{-11}\%$ e o erro máximo em $1.19 \times 10^{-10}\%$.

i	x_i obtido	Erro relativo ε_i [%]
0	$9.9999999999999956 \times 10^{-1}$	4.44×10^{-14}
1	$9.9999999999991640 \times 10^{-1}$	8.36×10^{-12}
2	$1.0000000000006128 \times 10^{0}$	6.13×10^{-11}
3	$9.9999999999881017 \times 10^{-1}$	1.19×10^{-10}
4	$1.0000000000006775 \times 10^{0}$	6.77×10^{-11}

Tabela 10: Solução numérica e erros relativos (em porcentagem) para $n = 5$ com pivotamento total.

Resultados para $n = 9$

Os valores obtidos apresentaram maior afastamento de $x^\star$, com erros relativos crescendo da ordem de $10^{-8}\%$ até $10^{-3}\%$, refletindo o mau condicionamento da matriz de Hilbert. O tempo de execução foi de 5.4×10^{-6} s (5400 ns). A média dos erros relativos ficou em $6.52 \times 10^{-4}\%$ e o erro máximo em $1.84 \times 10^{-3}\%$.

i	x_i obtido	Erro relativo ε_i [%]
0	$9.9999999973917819 \times 10^{-1}$	2.61×10^{-8}
1	$1.0000000179136685 \times 10^{0}$	1.79×10^{-6}
2	$9.9999969782445230 \times 10^{-1}$	3.02×10^{-5}
3	$1.0000021516375917 \times 10^{0}$	2.15×10^{-4}
4	$9.9999212377515512 \times 10^{-1}$	7.88×10^{-4}
5	$1.0000160558481568 \times 10^{0}$	1.61×10^{-3}
6	$9.9998158518321312 \times 10^{-1}$	1.84×10^{-3}
7	$1.0000111104225871 \times 10^{0}$	1.11×10^{-3}
8	$9.9999725747803547 \times 10^{-1}$	2.74×10^{-4}

Tabela 11: Solução numérica e erros relativos (em porcentagem) para $n = 9$ com pivotamento total.

Resultados para $n = 15$

Para a matriz de Hilbert de ordem 15, o método de Gauss com pivotamento total não conseguiu conter a instabilidade numérica. Os erros relativos variaram de $10^{-6}\%$ até a ordem de $10^{3}\%$, refletindo a severa perda de precisão causada pelo mau condicionamento.

O tempo de execução foi de 1.06×10^{-5} s (10600 ns). A média dos erros relativos foi de $4.80 \times 10^{2}\%$, com erro máximo de $2.01 \times 10^{3}\%$.

i	x_i obtido	Erro relativo ε_i [%]
0	$1.0000000342426496 \times 10^{0}$	3.42×10^{-6}
1	$9.9999131475720304 \times 10^{-1}$	8.69×10^{-4}
2	$1.0004617245322356 \times 10^{0}$	4.62×10^{-2}
3	$9.9008734255203312 \times 10^{-1}$	9.91×10^{-1}
4	$1.1109250851683967 \times 10^{0}$	1.11×10^{1}
5	$2.6396904638905827 \times 10^{-1}$	7.36×10^{1}
6	$4.1101874156505858 \times 10^{0}$	3.11×10^{2}
7	$-7.6922886207609542 \times 10^{0}$	8.69×10^{2}
8	$1.7268962273965968 \times 10^{1}$	1.63×10^{3}
9	$-1.9090844902959663 \times 10^{1}$	2.01×10^{3}
10	$1.6380783275406987 \times 10^{1}$	1.54×10^{3}
11	$-4.8736900721304179 \times 10^{0}$	5.87×10^{2}
12	$6.8343038611111329 \times 10^{-1}$	3.17×10^{1}
13	$2.1547049747974425 \times 10^{0}$	1.15×10^{2}
14	$6.9332061726012284 \times 10^{-1}$	3.07×10^{1}

Tabela 12: Solução numérica e erros relativos (em porcentagem) para $n = 15$ com pivotamento total.

Conclusão do método

- Para $n = 5$, manteve erros no limite de precisão de máquina, como nos demais métodos, mas o tempo foi 3,4 vezes maior que o sem pivotamento, 2,1 vezes maior que o parcial e 1,4 vez maior que o pivotamento por pesos.
- Em $n = 9$, trouxe leve melhora de precisão (menos de 10% frente ao parcial e ao pesos), mas o tempo foi 3,9 vezes maior que o sem pivotamento, 1,9 vez maior que o parcial e 2,0 vezes maior que o pesos.
- Para $n = 15$, não conteve a instabilidade: apresentou erros 45% piores que o sem pivotamento e cerca de 17–18% acima do parcial e do pesos. O tempo foi 2,4 vezes maior que o sem pivotamento, 1,4 vez maior que o parcial e 1,6 vez maior que o pesos.

Em síntese, o pivotamento total não trouxe vantagens: em matrizes pequenas apenas encareceu, em ordens intermediárias gerou ganhos marginais inferiores a 10% com custo quase duas vezes maior, e em ordens maiores foi o mais instável, com erros superiores e tempo mais elevado.

Análise das Soluções Diretas

Para facilitar a comparação entre os métodos, os valores de erro e tempo foram normalizados tomando o caso sem pivotamento como referência (1.0). Assim, cada entrada da Tabela 13 representa o fator relativo em relação a esse método base.

$n = 5$	Sem pivotamento	Parcial	Escala	Total
Erro médio	1.0	1.57	1.94	2.33
Erro máximo	1.0	1.92	2.33	2.77
Tempo	1.0	1.6	2.4	3.4
$n = 9$	Sem pivotamento	Parcial	Escala	Total
Erro médio	1.0	0.86	0.85	0.79
Erro máximo	1.0	0.86	0.85	0.79
Tempo	1.0	2.0	1.9	3.9
$n = 15$	Sem pivotamento	Parcial	Escala	Total
Erro médio	1.0	1.24	1.22	1.45
Erro máximo	1.0	1.20	1.21	1.46
Tempo	1.0	1.7	1.5	2.4

Tabela 13: Comparação relativa dos métodos de Gauss. Valores normalizados pelo caso sem pivotamento (referência = 1.0).

A Tabela 13 evidencia tendências claras:

- Para ordens pequenas ($n = 5$), o pivotamento apenas aumentou os erros e o tempo de execução, sem qualquer ganho de estabilidade.
- Em ordens intermediárias ($n = 9$), houve leve melhoria de precisão (até 20% no erro médio e máximo), mas ao custo de um tempo até quatro vezes maior, principalmente no pivotamento total.
- Em ordens maiores ($n = 15$), nenhum método conseguiu conter a instabilidade das matrizes de Hilbert: todos apresentaram erros superiores ao caso sem pivotamento, sendo o total o mais desfavorável, tanto em precisão quanto em custo.

Em síntese, o pivotamento mostrou-se pouco vantajoso neste conjunto de testes: para matrizes pequenas apenas encarece, em dimensões intermediárias gera ganhos marginais com custo elevado, e em ordens maiores intensifica a instabilidade numérica.

Resolução pelo Método de Jacobi

O método de Jacobi para o sistema linear $Ax = b$ se escreve na forma iterativa

$$x^{(k+1)} = B\,x^{(k)} + c,$$

onde

$$B = D^{-1}(L + U), \qquad c = D^{-1}b,$$

com $A = D - L - U$, sendo D a parte diagonal, $-L$ a parte triangular inferior e $-U$ a parte triangular superior de A.

A condição de convergência é

$$\rho(B) < 1,$$

onde $\rho(B)$ é o raio espectral de B, isto é, o maior valor absoluto dos autovalores de B.

No caso das matrizes de Hilbert, que não são diagonalmente dominantes, tem-se

$$b_{ij} = \frac{h_{ij}}{h_{ii}}, \quad i \neq j,$$

de modo que cada linha de B contém apenas elementos positivos. O somatório de cada linha é dado por

$$r_i = \sum_{j \neq i} b_{ij} = \frac{1}{h_{ii}} \left(\sum_{j=1}^{n} h_{ij} - h_{ii} \right).$$

Pelo teorema de Perron–Frobenius, para matrizes não negativas vale a desigualdade

$$\min_i r_i \leq \rho(B) \leq \max_i r_i.$$

$$n = 5: \quad \rho(B) \approx 3.44, \quad r_{\min} \approx 1.28, \quad r_{\max} \approx 5.71,$$

$$n = 9: \quad \rho(B) \approx 6.91, \quad r_{\min} \approx 1.83, \quad r_{\max} \approx 11.27,$$

$$n = 15: \quad \rho(B) \approx 12.13, \quad r_{\min} \approx 2.32, \quad r_{\max} \approx 19.59.$$

Em todos esses casos, $\rho(B) > 1$, o que explica o crescimento exponencial das iterações e a divergência observada na prática, com fatores médios de afastamento entre 1.25 e 1.33 por passo.

Nota-se, portanto, que o raio espectral cresce com a ordem da matriz de Hilbert, aumentando a instabilidade. De forma inversa, ao reduzir a ordem da matriz, $\rho(B)$ diminui. Para $n = 3$ já se observa $\rho(B) > 1$, mantendo a divergência, mas em $n = 2$ temos $\rho(B) < 1$, o que garante a convergência do método.

No caso $n = 2$, jacobi convergiu em 199 iterações, produzindo solução praticamente idêntica a $x^\star = (1, 1)^T$, com erros relativos da ordem de $10^{-11}\%$. O tempo de execução foi de 9.849×10^{-4} s (984,9 μs). A média dos erros relativos foi $3.21 \times 10^{-11}\%$ e o máximo $3.21 \times 10^{-11}\%$.

i	x_i obtido	Erro relativo ε_i [%]
0	$9.9999999999967970 \times 10^{-1}$	3.2030×10^{-11}
1	$9.9999999999967848 \times 10^{-1}$	3.2152×10^{-11}

Tabela 14: Solução numérica e erros relativos (em porcentagem) para $n = 2$ com Jacobi.

Resolução pelo Método de Gauss–Seidel

Resultados para $n = 5$

O método de Gauss–Seidel aplicado ao sistema de Hilbert de ordem 5 convergiu após aproximadamente 3.82×10^5 iterações. A solução obtida manteve-se próxima do vetor exato, com erro relativo médio de $1.10 \times 10^{-6}\%$ e erro máximo de $2.36 \times 10^{-6}\%$. Embora a precisão final seja aceitável, o custo computacional foi elevado: o tempo de execução alcançou 5.90×10^{-2} segundos, muito superior ao observado nos métodos diretos.

i	x_i obtido	Erro relativo ε_i [%]
0	1.0000000002018732	2.02×10^{-8}
1	0.9999999962941493	3.71×10^{-7}
2	1.0000000157885036	1.58×10^{-6}
3	0.9999999763618385	2.36×10^{-6}
4	1.0000000114888317	1.15×10^{-6}

Tabela 15: Solução numérica e erros relativos (em porcentagem) para $n = 5$ com Gauss–Seidel.

Resultados para $n = 9$

Para o sistema de Hilbert de ordem 9, o método de Gauss–Seidel convergiu apenas após cerca de 3.85×10^8 iterações, um custo computacional extremamente elevado. A solução aproximada apresentou erro relativo médio de $3.79 \times 10^{-2}\%$ e erro máximo de $1.12 \times 10^{-1}\%$, valores significativamente piores que os observados para $n = 5$. O tempo total de execução atingiu 1.72×10^2 segundos.

i	x_i obtido	Erro relativo ε_i [%]
0	1.0000000006639236	6.64×10^{-8}
1	0.9999996477987723	3.52×10^{-5}
2	1.0000097270311905	9.73×10^{-4}
3	0.9999100657688069	8.99×10^{-3}
4	1.0003892179788130	3.89×10^{-2}
5	0.9991070671869994	8.93×10^{-2}
6	1.0011188494002046	1.12×10^{-1}
7	0.9992767287133796	7.23×10^{-2}
8	1.0001887162545013	1.89×10^{-2}

Tabela 16: Solução numérica e erros relativos (em porcentagem) para $n = 9$ com Gauss–Seidel.

Resultados para $n = 15$

Para o sistema de Hilbert de ordem 15, o método de Gauss–Seidel também alcançou convergência, mas apenas após cerca de 1.99×10^9 iterações, um esforço computacional impraticável na prática. O erro relativo médio foi de $4.60 \times 10^{-2}\%$ e o máximo atingiu $1.26 \times 10^{-1}\%$, valores que evidenciam a perda progressiva de precisão à medida que cresce a ordem da matriz. O tempo de execução foi de aproximadamente 2.23×10^3 segundos (cerca de 37 minutos).

i	x_i obtido	Erro relativo ε_i [%]
0	0.9999999532492485	4.68×10^{-6}
1	1.0000027140827228	2.71×10^{-4}
2	0.9999635295082447	3.65×10^{-3}
3	1.0001803467134800	1.80×10^{-2}
4	0.9996903281339047	3.10×10^{-2}
5	0.9998331378170020	1.67×10^{-2}
6	1.0009075650354267	9.08×10^{-2}
7	0.9998598420092316	1.40×10^{-2}
8	0.9990548928728834	9.45×10^{-2}
9	0.9997033194527555	2.97×10^{-2}
10	1.0007805640517724	7.81×10^{-2}
11	1.0008707590563712	8.71×10^{-2}
12	0.9997077967941070	2.92×10^{-2}
13	0.9987370234041397	1.26×10^{-1}
14	1.0007082747693503	7.08×10^{-2}

Tabela 17: Solução numérica e erros relativos (em porcentagem) para $n = 15$ com Gauss–Seidel.

Conclusão do método

O método de Gauss–Seidel mostrou-se mais robusto que Jacobi, convergindo em todos os casos testados ($n = 5$, $n = 9$, $n = 15$).

- **Robustez:** sempre convergiu, ao contrário do Jacobi.
- **Velocidade:** exigiu de 10^5 a 10^9 iterações, com tempos que chegaram a mais de meia hora.
- **Precisão:** boa em $n = 5$ ($10^{-6}\%$), mas degradando para cerca de $10^{-2}\%$ em ordens maiores.

Em síntese, foi mais estável que Jacobi, mas lento e impreciso quando comparado aos métodos diretos, sendo mais indicado apenas em sistemas grandes e esparsos.

Controle de Estabilidade via Relaxamento em Métodos Iterativos

Uma das principais vantagens dos métodos iterativos em relação aos diretos é a possibilidade de introduzir um *parâmetro de relaxamento* $\omega > 0$ no processo de atualização do vetor solução. A fórmula básica do Gauss–Seidel é modificada para

$$x_i^{(k+1)} = (1-\omega)\, x_i^{(k)} + \omega\, \tilde{x}_i^{(k+1)},$$

em que $\tilde{x}_i^{(k+1)}$ corresponde ao valor obtido pela iteração padrão.

Quando $0 < \omega < 1$, aplica-se o *sub–relaxamento*: as atualizações são mais conservadoras, o que tende a aumentar a estabilidade, embora à custa de uma convergência mais lenta. Já para $1 < \omega < 2$, ocorre o *sobre–relaxamento*, que em muitos casos acelera o processo, mas pode gerar oscilações e até divergência quando ω é escolhido de forma excessiva.

Assim, o parâmetro ω atua como um recurso de ajuste, permitindo equilibrar estabilidade e velocidade de acordo com o problema — uma flexibilidade inexistente nos métodos diretos.

Resolução pelo Método de Gauss–Seidel com Sobre–Relaxamento (SOR)

Como já sabemos que o método de Gauss–Seidel converge para os sistemas analisados, podemos explorar maneiras de acelerar essa convergência. Uma estratégia clássica é aplicar o *sobre–relaxamento* (Successive Over–Relaxation, SOR), introduzindo um parâmetro $\omega > 1$ de forma a reduzir o número de iterações e tornar o processo mais eficiente em termos de tempo.

Assim, ao aplicar o sobre–relaxamento, realizamos um estudo de compromisso entre velocidade e precisão, na tentativa de obter uma convergência não apenas garantida, mas também alcançada em tempo viável. Nas subseções a seguir, analisamos como a variação do parâmetro ω influencia o desempenho do método em sistemas de Hilbert de ordens $n = 5$, $n = 9$ e $n = 15$.

SOR: variação de ω (caso $n = 5$)

Nos testes com a matriz de Hilbert de ordem 5, o parâmetro de sobre–relaxamento influenciou diretamente a velocidade de convergência. Com $\omega = 1.1$, o método exigiu mais de 3×10^5 iterações, enquanto valores maiores de ω reduziram o esforço de forma consistente: em $\omega = 1.8$, bastaram pouco mais de 4×10^4 iterações, com tempo seis vezes menor e precisão ainda superior à obtida inicialmente.

O limite, porém, aparece em $\omega = 1.9$: embora o número de iterações tenha diminuído ainda mais, a precisão foi severamente comprometida, indicando que valores excessivos de ω tornam o método instável.

ω	Iterações	Tempo [s]	Erro médio [%]	Erro máx. [%]
1.1	3.12×10^{5}	4.48×10^{-2}	1.34×10^{-6}	2.89×10^{-6}
1.2	2.61×10^{5}	3.73×10^{-2}	1.09×10^{-6}	2.34×10^{-6}
1.3	2.16×10^{5}	3.09×10^{-2}	8.74×10^{-7}	1.88×10^{-6}
1.4	1.76×10^{5}	2.55×10^{-2}	6.91×10^{-7}	1.48×10^{-6}
1.5	1.40×10^{5}	2.00×10^{-2}	5.31×10^{-7}	1.14×10^{-6}
1.6	1.07×10^{5}	1.55×10^{-2}	3.88×10^{-7}	8.30×10^{-7}
1.7	7.56×10^{4}	1.09×10^{-2}	2.59×10^{-7}	5.51×10^{-7}
1.8	4.21×10^{4}	6.11×10^{-3}	1.27×10^{-7}	2.66×10^{-7}
1.9	3.23×10^{4}	4.71×10^{-3}	8.73×10^{-5}	1.90×10^{-4}

Tabela 18: Gauss–Seidel com SOR para matriz de Hilbert ($n = 5$): iterações, tempo e erros relativos médios e máximos em porcentagem.

Em resumo, o SOR alcançou melhor desempenho em torno de $\omega = 1.8$, que reduziu significativamente o número de iterações sem perda de precisão. Valores mais altos, como $\omega = 1.9$, indicaram início de instabilidade numérica.

SOR: variação de ω (caso $n = 9$)

No caso da matriz de Hilbert de ordem 9, o SOR mostrou um trade-off claro entre custo computacional e precisão. Com $\omega = 1.1$, obteve-se boa acurácia, mas ao preço de mais de 2.5×10^{8} iterações e tempo acima de 100 s. O aumento de ω reduziu drasticamente o esforço: em $\omega = 1.3$, o número de iterações caiu para cerca de um terço, e em torno de $\omega = 1.4$ o tempo baixou para pouco mais de 1 s.

Essa redução, contudo, veio acompanhada de perda progressiva de precisão: os erros cresceram de forma consistente, chegando a ultrapassar 1% já para $\omega \geq 1.5$ e alcançando até 4.5% nos limites testados.

ω	Iterações	Tempo [s]	Erro médio [%]	Erro máx. [%]
1.1	2.57×10^{8}	1.03×10^{2}	4.43×10^{-2}	1.41×10^{-1}
1.2	2.10×10^{8}	8.39×10^{1}	5.29×10^{-2}	1.72×10^{-1}
1.3	8.21×10^{7}	3.35×10^{1}	1.61×10^{-1}	3.53×10^{-1}
1.4	2.90×10^{6}	1.19×10^{0}	8.18×10^{-1}	2.23×10^{0}
1.5	3.16×10^{6}	1.30×10^{0}	9.90×10^{-1}	2.78×10^{0}
1.6	3.09×10^{6}	1.26×10^{0}	1.14×10^{0}	3.39×10^{0}
1.7	2.80×10^{6}	1.15×10^{0}	1.24×10^{0}	3.97×10^{0}
1.8	2.25×10^{6}	9.39×10^{-1}	1.42×10^{0}	4.39×10^{0}
1.9	9.19×10^{5}	3.77×10^{-1}	1.62×10^{0}	4.53×10^{0}

Tabela 19: Gauss–Seidel com SOR para a matriz de Hilbert ($n = 9$): iterações, tempo e erros relativos médios e máximos em porcentagem.

Em resumo, valores de ω entre 1.3 e 1.4 representaram o melhor equilíbrio entre tempo e precisão, enquanto ω mais elevados aceleraram ainda mais a convergência, mas à custa de erros acima de 1%.

SOR: variação de ω (caso $n = 15$)

No sistema de Hilbert de ordem 15, o SOR reduziu drasticamente o esforço em comparação ao Gauss–Seidel puro, que demandava quase 2×10^9 iterações e mais de meia hora de processamento. Com o SOR, a convergência foi atingida com cerca de 10^6 iterações e tempos de apenas alguns segundos, um ganho de ordem de grandeza em eficiência.

O preço dessa aceleração, entretanto, foi a perda de acurácia: mesmo para os menores valores de ω, os erros médios permaneceram na faixa de 1% a 10%, muito acima dos $\approx 0.05\%$ obtidos pelo Gauss–Seidel puro. À medida que ω aumentou, o tempo continuou baixo, mas os erros se agravaram, ultrapassando 10% nos limites superiores.

ω	Iterações	Tempo [s]	Erro médio [%]	Erro máx. [%]
1.1	2.39×10^5	2.60×10^{-1}	6.63×10^0	1.81×10^1
1.2	4.42×10^6	4.82×10^0	1.67×10^0	5.13×10^0
1.3	4.19×10^6	4.47×10^0	1.86×10^0	6.17×10^0
1.4	4.01×10^6	4.37×10^0	2.07×10^0	7.34×10^0
1.5	3.84×10^6	4.24×10^0	2.32×10^0	8.67×10^0
1.6	3.71×10^6	3.97×10^0	2.70×10^0	1.02×10^1
1.7	3.63×10^6	3.92×10^0	3.52×10^0	1.20×10^1
1.8	3.72×10^6	3.98×10^0	5.56×10^0	1.46×10^1
1.9	4.41×10^6	4.92×10^0	1.18×10^1	2.09×10^1

Tabela 20: Gauss–Seidel com SOR para a matriz de Hilbert ($n = 15$): iterações, tempo e erros relativos médios e máximos em porcentagem.

Conclusão sobre o Sobre–Relaxamento

O sobre–relaxamento reduziu de forma significativa o número de iterações e o tempo em comparação ao Gauss–Seidel puro, sobretudo em matrizes menores: no caso $n = 5$, valores de ω entre 1.6 e 1.8 proporcionaram o melhor equilíbrio entre custo e precisão.

Para ordens maiores ($n = 9$ e $n = 15$), porém, o ganho de velocidade veio acompanhado de forte degradação da acurácia, com erros já da ordem de 1% a 10%, muito acima dos observados no Gauss–Seidel.

Em síntese, o SOR mostrou-se promissor apenas em sistemas moderados, enquanto em matrizes muito mal condicionadas o benefício em tempo não compensa a perda de precisão.

Resolução pelo Método de Jacobi com Sub–Relaxamento

Diferentemente do Gauss–Seidel, cujo processo iterativo converge para os sistemas analisados, o método de Jacobi apresenta maior propensão à divergência em matrizes mal condicionadas, como as de Hilbert. Por esse motivo, em vez de buscarmos acelerar um processo já convergente, como no caso do sobre–relaxamento, aqui o objetivo é outro: estabilizar o método.

Para isso, introduzimos o *sub–relaxamento*, adotando valores $0 < \omega < 1$ de forma a suavizar as atualizações e reduzir oscilações. Essa estratégia tende a retardar a convergência, mas pode impedir que a sequência de aproximações diverja, viabilizando a obtenção de soluções numéricas aceitáveis mesmo em situações em que o Jacobi puro falharia.

Nas subseções seguintes, analisamos a influência do parâmetro ω na convergência do método de Jacobi, começando pelo caso $n = 5$ e, posteriormente, ampliando para sistemas de ordem maior.

Jacobi: variação de ω (caso $n = 5$)

Para a matriz de Hilbert de ordem 5, o método de Jacobi padrão divergiu. A introdução de um fator de sub–relaxamento ($0 < \omega < 1$) alterou esse comportamento: valores adequados de ω estabilizaram o processo e permitiram convergência.

ω	Iterações	Tempo [s]	Erro médio [%]	Erro máx. [%]
0.05	7.43×10^{4}	7.57×10^{-3}	1.55×10^{-1}	3.30×10^{-1}
0.10	3.71×10^{4}	3.96×10^{-3}	1.55×10^{-1}	3.30×10^{-1}
0.20	2.28×10^{6}	2.18×10^{-1}	1.10×10^{-5}	2.37×10^{-5}
0.30	1.58×10^{6}	1.52×10^{-1}	7.31×10^{-6}	1.58×10^{-5}
0.40	1.22×10^{6}	1.18×10^{-1}	5.48×10^{-6}	1.18×10^{-5}

Tabela 21: Jacobi com sub–relaxamento aplicado à matriz de Hilbert ($n = 5$). Valores indicados em notação científica.

Os resultados indicam três regimes distintos: para ω muito baixos (0.05 ou 0.1), a convergência ocorreu rapidamente, mas com erros relativamente altos; para valores intermediários (0.2 a 0.4), alcançou-se excelente precisão, embora à custa de milhões de iterações e maior tempo; já para $\omega \geq 0.5$, o método deixou de convergir.

Assim, o sub–relaxamento mostrou-se eficaz para estabilizar o Jacobi em $n = 5$, mas apenas dentro de uma faixa restrita de parâmetros.

Jacobi: variação de ω (caso $n = 9$)

Para a matriz de Hilbert de ordem 9, o Jacobi padrão divergiu. Com sub–relaxamento ($0 < \omega < 1$), houve convergência apenas para valores baixos de ω (até 0.2), mas sempre com erro médio próximo de $1.26 \times 10^{-1}\%$, bem acima do Gauss–Seidel, e com alto custo em iterações.

ω	Iterações	Tempo [s]	Erro médio [%]	Erro máx. [%]
0.05	1.63×10^{5}	4.55×10^{-2}	1.26×10^{-1}	1.98×10^{-1}
0.10	8.15×10^{4}	2.30×10^{-2}	1.26×10^{-1}	1.98×10^{-1}
0.20	4.08×10^{4}	1.13×10^{-2}	1.26×10^{-1}	1.98×10^{-1}

Tabela 22: Jacobi com sub–relaxamento aplicado à matriz de Hilbert ($n = 9$). Valores indicados em notação científica.

Em síntese, o Jacobi relaxado só convergiu para ω muito baixos, mas com erros ainda altos e elevado custo. Para $\omega \geq 0.3$, não houve convergência, confirmando a baixa eficácia do método diante do mau condicionamento.

Jacobi: variação de ω (caso $n = 15$)

Para a matriz de Hilbert de ordem 15, o método de Jacobi padrão divergiu. Com a introdução do sub–relaxamento, a convergência ocorreu apenas para $\omega = 0.05$ e $\omega = 0.10$, ambos resultando em erros médios da ordem de $10^{-2}\%$ a $10^{-1}\%$. Esses casos, contudo, exigiram centenas de milhares de iterações e tempos ainda consideráveis. Para $\omega \geq 0.20$, não houve convergência dentro do limite máximo de iterações.

ω	Iterações	Tempo [s]	Erro médio [%]	Erro máx. [%]
0.05	2.62×10^{5}	1.83×10^{-1}	7.63×10^{-2}	1.52×10^{-1}
0.10	1.31×10^{5}	9.33×10^{-2}	7.63×10^{-2}	1.52×10^{-1}

Tabela 23: Jacobi com sub–relaxamento aplicado à matriz de Hilbert ($n = 15$). Valores indicados em notação científica.

Conclui-se que, para $n = 15$, o sub–relaxamento não garantiu estabilidade de forma geral: apenas valores muito baixos de ω levaram à convergência, e ainda assim com alto custo em iterações. Na prática, o Jacobi mostrou-se pouco competitivo diante do Gauss–Seidel em sistemas de maior ordem.

Conclusão sobre o Sub–Relaxamento

A análise mostrou que o parâmetro ω atua como mecanismo de estabilização para o Jacobi. Para $n = 5$, valores intermediários ($0.2 \leq \omega \leq 0.4$) deram boa precisão, mas com custo elevado, enquanto valores muito baixos (0.05 ou 0.1) levaram a convergência mais rápida, porém com erros maiores.

Nos casos $n = 9$ e $n = 15$, apenas ω muito pequenos (até 0.2 e 0.1, respectivamente) geraram convergência, ainda assim com erros acima do Gauss–Seidel e custo elevado.

Em síntese, o sub–relaxamento tornou possível a convergência do Jacobi onde ele diverge, mas às custas de muitas iterações e precisão limitada. Assim, trata-se mais de um recurso paliativo de estabilização do que de uma alternativa competitiva, sobretudo em sistemas de maior ordem.

Análise das Soluções Iterativas

Para destacar as diferenças entre os métodos, os valores de iterações, tempo e erro foram normalizados tomando o Gauss–Seidel como referência (1.0). Assim, cada entrada da Tabela 24 representa o fator relativo em relação a esse método base.

$n = 5$	Gauss–Seidel (ref.)	Jacobi ($\omega = 0.4$)	SOR ($\omega \approx 1.8$)
Iterações	1.0	3.20	1.10×10^{-1}
Tempo	1.0	2.01	1.03×10^{-1}
Erro médio	1.0	4.98	1.15×10^{0}
Erro máximo	1.0	5.02	1.13×10^{0}
$n = 9$	Gauss–Seidel (ref.)	Jacobi ($\omega = 0.2$)	SOR ($\omega \approx 1.9$)
Iterações	1.0	1.06×10^{-4}	2.4×10^{-3}
Tempo	1.0	6.55×10^{-5}	2.2×10^{-3}
Erro médio	1.0	3.33	4.27×10^{1}
Erro máximo	1.0	1.76	4.04×10^{1}
$n = 15$	Gauss–Seidel (ref.)	Jacobi ($\omega = 0.1$)	SOR ($\omega \approx 1.4$)
Iterações	1.0	6.59×10^{-5}	2.0×10^{-3}
Tempo	1.0	4.18×10^{-5}	2.0×10^{-3}
Erro médio	1.0	1.66	4.51×10^{1}
Erro máximo	1.0	1.21	5.83×10^{1}

Tabela 24: Comparação relativa dos métodos iterativos. Valores normalizados pelo Gauss–Seidel (referência = 1.0). Para o Jacobi, foram usados valores de sub–relaxamento que efetivamente convergiram em cada caso.

A comparação evidencia que, para $n = 5$, o SOR reduziu em cerca de 90% o número de iterações e o tempo, mantendo erros próximos ao Gauss–Seidel. O Jacobi sub–relaxado ($\omega = 0.4$) também convergiu, mas foi mais lento e com erros até cinco vezes maiores. Nos casos $n = 9$ e $n = 15$, o Jacobi com ω muito baixos (0.2 e 0.1) mostrou-se extremamente rápido (fatores 10^4–10^5 em relação ao Gauss–Seidel), porém menos preciso, com erros até três vezes superiores. O SOR, por sua vez, manteve ganhos expressivos de tempo ($\sim 500\times$), mas com perda de precisão acentuada, chegando a erros 40–60$\times$ maiores que os do Gauss–Seidel.

Em síntese:

- O **SOR** é altamente eficaz em matrizes pequenas ($n = 5$), mas perde qualidade em sistemas mais mal condicionados.
- O **Jacobi com sub–relaxamento** pode estabilizar e oferecer tempos ínfimos em ordens maiores, embora com erro superior ao Gauss–Seidel.
- O **Gauss–Seidel** permanece como referência de estabilidade e precisão, mas é o mais custoso em tempo para sistemas grandes.

Comparação Global: Métodos Diretos e Iterativos

Nesta seção são comparados conjuntamente os métodos diretos (eliminação de Gauss, com e sem pivotamento) e os iterativos (Jacobi com sub–relaxamento, Gauss–Seidel e SOR). Para uniformizar as análises, os valores de erro médio, erro máximo e tempo foram normalizados tomando como referência o método de Gauss sem pivotamento (= 1.0). A Tabela 25 resume os resultados para $n = 5$, $n = 9$ e $n = 15$.

$n = 5$	Sem pivot.	Parcial	Escala	Total	Jacobi ($\omega = 0.4$)	Gauss–Seidel	SOR ($\omega \approx 1.8$)
Erro médio	1.0	1.57	1.94	2.33	5.0	5.0×10^4	4.4×10^4
Erro máximo	1.0	1.92	2.33	2.77	5.0	5.5×10^4	4.8×10^4
Tempo	1.0	1.6	2.4	3.4	2.0	1.2×10^5	1.0×10^4
$n = 9$	Sem pivot.	Parcial	Escala	Total	Jacobi ($\omega = 0.2$)	Gauss–Seidel	SOR ($\omega \approx 1.9$)
Erro médio	1.0	0.86	0.85	0.79	3.3	4.6×10^4	2.0×10^6
Erro máximo	1.0	0.86	0.85	0.79	1.8	4.8×10^4	2.0×10^6
Tempo	1.0	2.0	1.9	3.9	6.6×10^{-5}	1.2×10^8	2.7×10^5
$n = 15$	Sem pivot.	Parcial	Escala	Total	Jacobi ($\omega = 0.1$)	Gauss–Seidel	SOR ($\omega \approx 1.4$)
Erro médio	1.0	1.24	1.22	1.45	1.7	1.4×10^3	8.1×10^3
Erro máximo	1.0	1.20	1.21	1.46	1.2	1.5×10^3	1.2×10^4
Tempo	1.0	1.7	1.5	2.4	4.2×10^{-5}	5.1×10^8	3.8×10^5

Tabela 25: Comparação relativa entre métodos diretos e iterativos. Valores normalizados pelo Gauss sem pivotamento (= 1.0). Para Jacobi, foram usados os valores de ω que efetivamente convergiram em cada caso.

A Tabela 25 mostra que, para $n = 5$, os métodos diretos mantêm erros próximos ao limite de máquina. O Jacobi sub–relaxado convergiu com erros cerca de cinco vezes maiores e tempo duas vezes superior ao Gauss sem pivotamento, enquanto o SOR reduziu o tempo em quase 90% com precisão semelhante ao Gauss–Seidel. Em $n = 9$, o Jacobi convergiu com erros duas a três vezes maiores, mas tempo praticamente nulo, ao passo que Gauss–Seidel e SOR tiveram custos astronômicos e erros dezenas a centenas de vezes piores. Já em $n = 15$, o Jacobi ($\omega = 0.1$) manteve erros levemente superiores aos diretos, mas tempo ínfimo; em contrapartida, Gauss–Seidel e SOR tornaram-se impraticáveis, com enorme custo e forte perda de precisão.

Em síntese:

- Os **métodos diretos** seguem mais precisos e estáveis, embora apresentem instabilidade crescente em $n = 15$.
- O **Jacobi com sub–relaxamento** consegue estabilizar e manter tempos mínimos, mas com erros maiores que os diretos.
- O **Gauss–Seidel e o SOR** são úteis apenas em dimensões pequenas, pois se tornam proibitivos em custo e precisão em ordens maiores.

Exercício 2: Inversão por decomposição LU (sem pivotamento)

Utilize o método de decomposição LU, **sem pivotamento**, para calcular as inversas das matrizes abaixo. Em cada caso, verifique o resultado computando $A\,A^{-1}$ e confira se $\approx I$. Comente a qualidade numérica dos resultados.

$$A = \begin{bmatrix} 10 & 2 & -1 \\ -3 & -6 & 2 \\ 1 & 1 & 5 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 4 & 9 & 16 \\ 4 & 9 & 16 & 25 \\ 9 & 16 & 25 & 36 \\ 16 & 25 & 36 & 49 \end{bmatrix}$$

Caso 3×3

Decomposição LU

Para a matriz

$$A_{3\times3} = \begin{bmatrix} 10 & 2 & -1 \\ -3 & -6 & 2 \\ 1 & 1 & 5 \end{bmatrix},$$

a fatoração LU produziu as seguintes matrizes:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.3000000 & 1 & 0 \\ 0.1000000 & -1.481481 \times 10^{-1} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 10 & 2 & -1 \\ 0 & -5.400000 & 1.700000 \\ 0 & 0 & 5.351852 \end{bmatrix}.$$

Resultados Numéricos da Inversão

A inversa obtida foi:

$$A^{-1} = \begin{bmatrix} 1.107266 \times 10^{-1} & 3.806228 \times 10^{-2} & 6.920415 \times 10^{-3} \\ -5.882353 \times 10^{-2} & -1.764706 \times 10^{-1} & 5.882353 \times 10^{-2} \\ -1.038062 \times 10^{-2} & 2.768166 \times 10^{-2} & 1.868512 \times 10^{-1} \end{bmatrix}.$$

Multiplicando $A \cdot A^{-1}$, obteve-se

$$A \cdot A^{-1} = \begin{bmatrix} 1.0000000 & -2.081668 \times 10^{-17} & 0 \\ -2.081668 \times 10^{-17} & 0.9999999 & 5.551115 \times 10^{-17} \\ -6.938894 \times 10^{-18} & 2.775558 \times 10^{-17} & 1.0000000 \end{bmatrix}.$$

O resultado é uma matriz muito próxima da identidade, I. Os elementos fora da diagonal principal são da ordem de 10^{-17}, consistentes com erros de precisão numérica (aritmética de ponto flutuante).

O erro absoluto elemento a elemento foi:

$$E = \begin{bmatrix} 0 & 2.081668 \times 10^{-17} & 0 \\ 2.081668 \times 10^{-17} & 2.220446 \times 10^{-16} & 5.551115 \times 10^{-17} \\ 6.938894 \times 10^{-18} & 2.775558 \times 10^{-17} & 0 \end{bmatrix}.$$

O erro da inversão foi quantificado através do máximo elemento em magnitude da matriz resíduo $E_1 = R_1 - I = A_1 A_1^{-1} - I$. Esta métrica é definida por

$$\text{Erro} = \max_{1 \le i,j \le n} |(E_1)_{ij}|.$$

$$\textbf{Máximo:} \quad \max |E| = 2.220446 \times 10^{-16}$$

Resultados Experimentais

Para aumentar a confiabilidade das medições, o procedimento de inversão foi executado 1000 vezes consecutivas, e o tempo médio obtido foi de aproximadamente $4{,}507 \times 10^{-7}$ s. Os resultados indicam que o método de inversão via decomposição LU apresentou **excelente estabilidade numérica**, com erros residuais na ordem de 10^{-16}, o que está em conformidade com o **erro de arredondamento esperado em operações de ponto flutuante de dupla precisão**.

Caso 4×4

Decomposição LU

Para a matriz

$$A_{4\times4} = \begin{bmatrix} 1 & 4 & 9 & 16 \\ 4 & 9 & 16 & 25 \\ 9 & 16 & 25 & 36 \\ 16 & 25 & 36 & 49 \end{bmatrix},$$

a fatoração LU resultou em:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ 9 & 2.857143 & 1 & 0 \\ 16 & 5.571429 & 3 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 4 & 9 & 16 \\ 0 & -7 & -20 & -39 \\ 0 & 0 & 1.142857 & 3.428571 \\ 0 & 0 & 0 & 4.973799 \times 10^{-14} \end{bmatrix}.$$

Note que o último pivô $U_{44} \approx 4.97 \times 10^{-14}$, praticamente nulo. O determinante calculado como produto dos pivôs foi:

$$\det(A_{4\times4}) \approx -3.98 \times 10^{-13} \approx 0.$$

Isso confirma que a matriz $A_{4\times4}$ é **singular** (ou mal-condicionada a ponto de não ser invertível numericamente). Assim, o processo de inversão por LU falhou, pois não é possível calcular uma inversa única para $A_{4\times4}$.

Análise da Dependência Linear

Os elementos da matriz $A_{4\times4}$ seguem o padrão $a_{ij} = (i + j - 1)^2$, o que permite demonstrar **analiticamente** a sua singularidade. As linhas de A_2 (denotadas por L_i) não são linearmente independentes. De fato, observa-se que:

$$L_3 - 2L_2 + L_1 = 0, \qquad L_4 - 2L_3 + L_2 = 0.$$

Por exemplo, para a terceira linha:

$$[9,\ 16,\ 25,\ 36] - 2 \times [4,\ 9,\ 16,\ 25] + [1,\ 4,\ 9,\ 16] = [0,\ 0,\ 0,\ 0].$$

A existência dessas combinações lineares não triviais entre as linhas mostra que elas são **linearmente dependentes**. Consequentemente, o determinante da matriz é nulo,

$$\det(A_2) = 0,$$

o que confirma que A é **singular** e, portanto, não admite matriz inversa.

Resultados Numéricos da Inversão

A tentativa de calcular a inversa numericamente, apesar da singularidade, produz a seguinte matriz A_2^{-1} (resultante da instabilidade numérica):

$$A^{-1} = \begin{bmatrix} 2.010536 \times 10^{13} & -6.031607 \times 10^{13} & 6.031607 \times 10^{13} & -2.010536 \times 10^{13} \\ -6.031607 \times 10^{13} & 1.809482 \times 10^{14} & -1.809482 \times 10^{14} & 6.031607 \times 10^{13} \\ 6.031607 \times 10^{13} & -1.809482 \times 10^{14} & 1.809482 \times 10^{14} & -6.031607 \times 10^{13} \\ -2.010536 \times 10^{13} & 6.031607 \times 10^{13} & -6.031607 \times 10^{13} & 2.010536 \times 10^{13} \end{bmatrix}.$$

A verificação do produto $A \cdot A^{-1}$ demonstra a falha do cálculo:

$$A \cdot A^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.125 & 1 & 0 & -0.125 \\ 0.25 & 0.5 & 0.5 & -0.25 \\ 1.125 & -0.5 & 2.5 & 0 \end{bmatrix}.$$

O resultado não se aproxima da matriz identidade. O erro, medido pela mesma métrica do máximo elemento em magnitude do resíduo ($E_2 = |R_2 - I|$), é:

$$E = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.125 & 0 & 0 & 0.125 \\ 0.25 & 0.5 & 0.5 & 0.25 \\ 1.125 & 0.5 & 2.5 & 1 \end{bmatrix}.$$

$$\textbf{Máximo:} \quad \max |E| = 2.5$$

Esses valores elevados (com entradas de ordem unitária no resíduo) confirmam que $A_{4\times4}$ é **singular** e que os resultados numéricos para a "inversa" são **inválidos** (produto $A \cdot A^{-1}$ distante de I).

Resultados Experimentais

O experimento com a matriz singular foi repetido 1000 vezes, resultando em um tempo médio de execução de aproximadamente $5{,}364 \times 10^{-7}$ s. Apesar da execução ter sido concluída, observou-se a presença de **pivôs próximos de zero**, indicando **instabilidade numérica significativa**. Os resultados da inversa obtida não são confiáveis, uma vez que o produto AA^{-1} se afastou substancialmente da matriz identidade, evidenciando que o sistema é **singular** e que a inversão direta por decomposição LU não é aplicável neste caso.