Revista de Estudos Interdisciplinares

ANÁLISE DO DESEMPENHO DAS FATORAÇÕES QR E CHOLESKY USANDO ESTRUTURA DE DADOS INDEXADA E DE LISTAS ENCADEADAS

ANALYSIS OF THE PERFORMANCE OF QR AND CHOLESKY FACTORIZATIONS USING INDEXED AND LINKED LIST DATA STRUCTURES

Recebido em: 21/01/2026
Aceito em: 10/06/2026
Publicado em: 21/07/2026

Débora Rezende Jalles[1]
Universidade Federal Fluminense

Ricardo Silveira Sousa[2]
Universidade Federal Fluminense

Thiago Jordem Pereira[3]
Universidade Federal Fluminense

Wagner Rambaldi Telles[4]
Universidade Federal Fluminense

Resumo: Inúmeros problemas reais das áreas de física, química e engenharias envolvem sistemas lineares de grande porte e esparsos, compostos por milhares de equações Em alguns casos, tais sistemas admitem uma única solução, passível de ser obtida através de métodos diretos. Dentre eles, destacam-se aqueles que utilizam a decomposição da matriz de coeficientes, como ocorre com a Fatoração QR e de Cholesky. Esses métodos visam obter uma solução muito próxima da exata em um número limitado de passos. Eeste trabalho constitui no estudo e na implemetação dos métodos de Fatoração QR e de Cholesky usando estrutura de dados indexada e de listas encadeadas. Os resultados computacionais indicam que a estrutura de listas encadeadas teve um desempenho inferior com a Fatoração QR, mas o resultado inverteu-se com o método de Cholesky. Contudo, em ambos os métodos, o emprego de listas encadeadas resultou em um consumo de memória inferior, mostrando-se uma alternativa eficiente e eficaz para o armazenamento de dados.

Palavras-chave: Fatoração QR; Fatoração de Cholesky; Listas Encadeadas. Matrizes Esparsas.

Abstract: Numerous real-world problems in physics, chemistry, and engineering involve large-scale, sparse linear systems composed of thousands of equations. In some cases, such systems admit a single solution, obtainable through direct methods. Among these, those that utilize the decomposition of the coefficient matrix stand out, as occurs with QR and Cholesky factorization. These methods aim to obtain a solution very close to the exact one in

[1] Mestre em Modelagem Computacional Ciência e Tecnologia na Universidade Federal Fluminense. Brasil, Rio de Janeiro. Santo Antônio de Pádua, 2025. Professora de Matemática/Ciências Físicas da Secretaria Municipal de Educação de Itaperuna, RJ. E-mail: deborajalles@id.uff.br

[2] Doutorado em Ciência da Computação e Matemática Computacional na USP. Professor e pesquisador permanente do Programa de Pós-graduação em Modelagem Computacional Ciência e Tecnologia da Universidade Federal Fluminense. E-mail: rsousa@id.uff.br

[3] Doutorado em Modelagem Computacional na UERJ. Professor e pesquisador permanente do Programa de Pós-graduação em Modelagem Computacional Ciência e Tecnologia da Universidade Federal Fluminense. E-mail: tjordem@id.uff.br

[4] Doutorado em Modelagem Computacional na UERJ. Professor colaborador do Doutorado Profissional em Modelagem e Tecnologia para Meio Ambiente Aplicadas em Recursos Hídricos (AmbHidro), ofertado pelo Instituto Federal Fluminense (IFFluminense). Professor e pesquisador permanente do Programa de Pós-graduação em Modelagem Computacional Ciência e Tecnologia da Universidade Federal Fluminense. E-mail: wtelles@id.uff.br

a limited number of steps. This work consists of the study and implementation of the QR and Cholesky factorization methods using indexed data structures and linked lists. The computational results indicate that the linked list structure performed worse with QR factorization, but the result was reversed with the Cholesky method. However, in both methods, the use of linked lists resulted in lower memory consumption, proving to be an efficient and effective alternative for data storage.

Keywords: QR Factorization; Cholesky Factorization; Linked Lists; Sparse Matrices.

INTRODUÇÃO

Dado um conjunto de equações com as mesmas variáveis, sua solução consiste no conjunto de valores que satisfazem todas essas equações simultaneamente. Se sua matriz de coeficientes for quadrada e formada por vetores linearmente independentes, o sistema possui solução única e pode ser representado como:

$$\boldsymbol{Ax} = \boldsymbol{b}, \tag{1}$$

sendo $\boldsymbol{A}$ a matriz dos coeficientes, $\boldsymbol{x}$ o vetor (ou matriz coluna) das incógnitas e $\boldsymbol{b}$ o vetor (ou matriz coluna) das constantes.

Muitos trabalhos ressaltam a importância de solucionar sistemas lineares desta natureza. Colla (2007) utilizou a Fatoração de Cholesky para resolver sistemas de Redes Bayesianas. Costa (2008) apresentou um estudo sobre fluxo de potências usando métodos numéricos diretos. Pescador, Possamai e Possamai (2011) discutiram a aplicação de sistemas lineares nos cursos de Engenharia, abrangendo circuitos elétricos, no balanceamento de equações e cálculos de estruturas metálicas.

Dentre os métodos numéricos para resolver problemas com solução única, destacam-se os métodos diretos que envolvem a fatoração da matriz de coeficientes, como os métodos de Decomposição QR e de Cholesky (Arenales; Darezzo, 2015; Cunha, 2000). Tais métodos fornecem uma solução muito próxima da analítica (contando apenas com erros de arredondamento) em um número limitado de passos. Neste contexto, o tempo de resolução dos problemas é influenciado pelo método numérico escolhido.

Além da escolha do método numérico, a estrutura de dados empregada é de suma importância. Na estrutura indexada, todos os elementos são armazenados, inclusive os nulos, o que resulta em um número elevado de operações e uso ineficiente de memória. Como consequência, observa-se o aumento no tempo computacional e, frequentemente, a ocorrência de estouro de memória (*overflow*) para sistemas lineares maiores.

Uma alternativa para minimizar este problema é armazenar apenas os elementos não nulos das matrizes em uma estrutura de listas encadeadas. Essa abordagem permite otimizar o uso de memória e evitar operações desnecessárias com elementos nulos. Ressalta-se, contudo, que

realizar operações para os métodos em questão com esta estrutura de dados não é uma tarefa fácil.

Desta forma, este artigo apresenta os resultados obtidos a partir do estudo dos métodos numéricos de Fatoração QR e de Cholesky. Compara-se o desempenho computacional utilizando estrutura de dados indexada e de listas encadeadas quando se aplica a sistemas lineares esparsos com milhares de variáveis.

PROCEDIMENTOS METODOLÓGICOS

FATORAÇÃO QR

A Fatoração QR consiste em decompor a matriz de coeficientes $\boldsymbol{A}$ em um produto $\boldsymbol{QR}$, sendo $\boldsymbol{Q}$ uma matriz ortonormal e $\boldsymbol{R}$ triangular superior da seguinte forma:

$$\boldsymbol{A} = \boldsymbol{QR}, \tag{2}$$

Logo, o sistema linear descrito pela Equação (1) passa a ser reescrito como,

$$\boldsymbol{QRx} = \boldsymbol{b}. \tag{3}$$

Multiplica-se ambos os lados da equação (3) por $\boldsymbol{Q}^T$ (a transposta de $\boldsymbol{Q}$), visto que $\boldsymbol{Q}$ é ortogonal e $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$ (onde $\boldsymbol{I}$ é a matriz identidade). Assim, tem-se:

$$\boldsymbol{Rx} = \boldsymbol{Q}^T\boldsymbol{b}. \tag{4}$$

Considerando $\boldsymbol{Rx} = \boldsymbol{y}$, obtém-se o primeiro sistema linear,

$$\boldsymbol{y} = \boldsymbol{Q}^T\boldsymbol{b}. \tag{5}$$

Com o valor de $\boldsymbol{y}$, encontra-se posteriormente o valor de $\boldsymbol{x}$, visto que

$$\boldsymbol{y} = \boldsymbol{Rx}. \tag{6}$$

A ortogonalização de Gram-Schmidt transforma o conjunto de um espaço vetorial em um conjunto ortogonal (Trefethen; Bau, 1997; Boldrini *et al*., 1980). Ou seja, transforma os vetores coluna da matriz em vetores ortogonais $\hat{q}_k$ quando subtrai as componentes nas direções dos vetores anteriores. Com isso, o produto interno entre quaisquer dois vetores coluna distintos é nulo. Assim,

$$\hat{q}_k = a_k - \sum_{j=1}^{k-1} r_{jk} q_j, k = 1,\ldots,n. \tag{7}$$

Se, além de ortogonal, o vetor $\hat{q}_k$ for normalizado (ou seja, tiver comprimento igual a 1), obtém-se para $k = 1,\ldots,n$ os vetores ortonormais q_k:

$$q_k = \frac{\hat{q}_k}{r_{kk}} \tag{8}$$

Ao isolar a_k na equação (7) e substituir $\hat{q}_k$ encontrado em (8), tem-se que para $k = 1,...,n$:

$$a_k = \sum_{j=1}^{k-1} r_{jk} q_j + r_{kk} q_k \tag{9}$$

Sabendo os valores de k, tem-se:

$$\begin{aligned} a_1 &= q_1 r_{11} \\ a_2 &= q_1 r_{12} + q_2 r_{22} \\ &\vdots \\ a_n &= q_1 r_{1n} + q_2 r_{2n} + \ldots + q_n r_{nn} \end{aligned} \tag{10}$$

ou ainda, na forma matricial,

$$\begin{pmatrix} a_1 & a_2 \ldots a_n \end{pmatrix} = \begin{pmatrix} q_1 & q_2 \ldots q_n \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{pmatrix} \tag{11}$$

De acordo com Jalles (2025) destaca-se que o algoritmo da Fatoração QR requer aproximadamente n^3 operações para resolver o problema.

FATORAÇÃO DE CHOLESKY

Para resolver um sistema linear utilizando a Fatoração de Cholesky, a matriz de coeficientes $\boldsymbol{A}$ precisa ser simétrica ($\boldsymbol{A} = \boldsymbol{A}^T$) e positiva definida ($\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} > 0$ para todo $\boldsymbol{x} \neq \boldsymbol{0}$). Nesse processo, a matriz $\boldsymbol{A}$ é decomposta no produto de duas matrizes triangulares:

$$\boldsymbol{A} = \boldsymbol{L}^T \boldsymbol{L} \tag{12}$$

e assim, o sistema é reescrito como:

$$\boldsymbol{L}^T \boldsymbol{L}\boldsymbol{x} = \boldsymbol{b}. \tag{13}$$

Define-se $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{y}$, para obter o primeiro sistema linear triangular:

$$\boldsymbol{L}^T \boldsymbol{y} = \boldsymbol{b}. \tag{14}$$

Após determinar o valor de $\boldsymbol{y}$, pode-se encontra o valor de $\boldsymbol{x}$ resolvendo o segundo sistema linear triangular:

$$\boldsymbol{L}\boldsymbol{x} = \boldsymbol{y}. \tag{15}$$

Vale a pena mencionar que a representação matricial do produto $\boldsymbol{L}^T\boldsymbol{L} = \boldsymbol{A}$ é dada por:

$$\begin{pmatrix} l_{11} & & & & \\ l_{12} & l_{22} & & & \\ l_{13} & l_{23} & l_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ l_{1n} & l_{2n} & l_{3n} & \dots & l_{nn} \end{pmatrix} \begin{pmatrix} l_{11} & l_{12} & l_{13} & \dots & l_{1n} \\ & l_{22} & l_{23} & \dots & l_{2n} \\ & & l_{33} & \dots & l_{3n} \\ & & & \ddots & \vdots \\ & & & & l_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} \tag{16}$$

O produto entre as linhas da primeira matriz e as colunas da segunda (transposta da primeira) resulta nos elementos de $\boldsymbol{L}$, como mostrado a seguir. Para os elementos da diagonal, tem-se:

$$l_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} r_{ki}^2 \right)^{1/2}, \; i = 1,...,n. \tag{17}$$

Agora, para os elementos fora da diagonal, vem que:

$$l_{ij} = \frac{\left(a_{ij} - \sum_{k=1}^{i-1} l_{ki} l_{kj} \right)}{l_{ii}}, \; i = 1,...,n \text{ e } j = i+1,...,n. \tag{18}$$

Deste modo, tem-se que o algoritmo da Fatoração de Cholesky requer $n^3/3$, conforme apresentado no trabalho de Jalles (2025).

ESTRUTURA DE DADOS E ESPARSIDADE

A porcentagem de elementos não nulos de uma matriz recebe o nome de densidade, enquanto a porcentagem de elementos nulos, esparsidade. A partir desses parâmetros classificam-se as matrizes em densas ou esparsas.

Neste trabalho, o foco recai sobre sistemas lineares cujas matrizes de coeficientes são quadradas e esparsas, isto é, apresentam uma quantidade de elementos nulos significativamente maior do que a de elementos não nulos. As matrizes de maior densidade analisadas apresentam estrutura triangular, resultando em densidade aproximada de 50%.

Para resolução dos sistemas lineares foram utilizadas estruturas de dados indexada e de listas encadeadas. Na estrutura indexada, os elementos da matriz de coeficientes são acessados a partir de seus índices e são armazenados como matrizes, conforme a forma algébrica a_{ij}. Na linguagem C, uma matriz armazenada com estrutura de dados indexada, pode ser representada

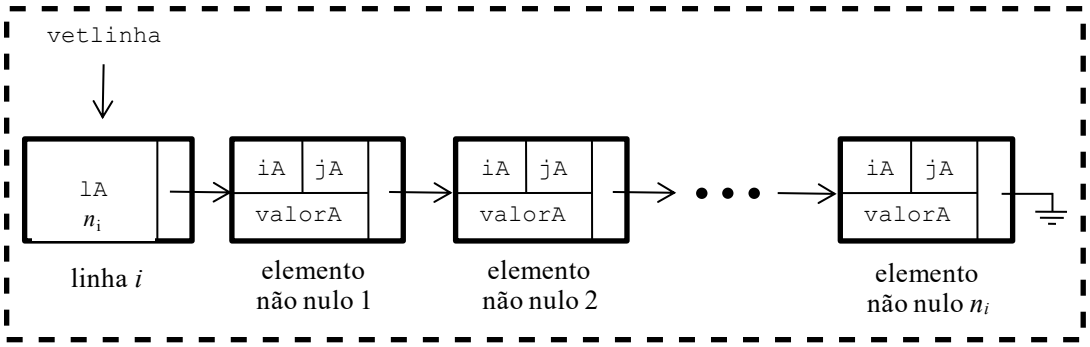
como "`double A[i][j]`", sendo *double* o tipo da variável de precisão dupla utilizada para armazenar números reais (Ziviani, 1999). Uma vantagem dessa estrutura é o acesso direto aos elementos a partir de seus respectivos índices.

No entanto, ao armazenar uma matriz esparsa de forma indexada reserva-se espaço na memória para todos os seus elementos. Dessa forma, além do armazenamento desnecessário, esses elementos participam de operações aritméticas também desnecessárias, como somas ou subtrações com zero, por exemplo.

Na estrutura de listas encadeadas para o armazenamento de matrizes, para cada linha deve haver uma sequência de nós, onde cada elemento não nulo é armazenado em um nó. Cada nó deve possuir informações sobre sua localização na matriz e um ponteiro para indicar o armazenamento do próximo elemento não nulo.

Neste trabalho, a estrutura de listas empregada utiliza uma combinação de duas listas encadeadas interligadas entre si. Desta forma, tem-se uma lista de linhas (com n nós), sendo que cada nó desta lista funciona como o nó cabeça da lista encadeada para armazenar os elementos não nulos para cada linha da matriz. Logo, para a lista encadeada de cada linha, o número de nós é determinado pela quantidade de elementos não nulos da linha correspondente da matriz. Esta estrutura foi utilizada para armazenar as matrizes ***A***, ***Q*** e ***R***. Na Figura 1, encontra-se uma representação da estrutura de lista encadeada para cada linha da matriz a ser armazenada.

Figura 1 – Representação de uma linha de uma matriz ***A*** com a estrutura de lista encadeada.



Fonte: elaborado pelos autores.

Na Figura 1 tem-se que: a) n_i: indica a quantidade de elementos não nulos na linha i da matriz, com $1 \leq n_i \leq n$; b) `vetlinha` é um ponteiro para lista de linhas da matriz; c) `lA`: armazena a referida linha da matriz, `lA`=1,...,n; d) `iA` e `jA`: armazenam o índice da linha e coluna respectivamente de cada elemento não nulo. Observa-se que para cada linha, o valor de `iA` é igual em todos os nós; e e) `valorA`: armazena o referido elemento não nulo da linha em questão. Neste caso, considera-se que a matriz tem pelo menos um elemento não nulo que se

encontra na diagonal. Observe que com o ponteiro `vetlinhaA` acessa-se cada linha da matriz fazendo-se o incremento para o endereço de memória a partir da primeira linha.

Diferentemente da estrutura indexada onde os elementos da matriz de coeficientes são acessados diretamente, na estrutura de listas encadeadas isso não ocorre. Para acessar um elemento a_{ij} é preciso percorrer a lista que representa a linha da matriz até encontrá-lo. Esta operação pode levar um tempo considerável dependendo do problema.

Para as operações, busca-se o elemento desejado e caso exista, efetua-se a operação. Se o elemento não for encontrado, então seu valor é nulo. Se for uma operação de soma ou subtração com um elemento não encontrado, trata-se de uma soma ou subtração por zero que não é realizada. Se for uma multiplicação (e um dos valores não foi encontrado), ou uma divisão de determinado elemento não encontrado (portanto, igual a zero) por outro qualquer (diferente de zero); o resultado também é sempre nulo. Logo, evitam-se estes tipos de operações. Com isso, nota-se uma economia no tempo computacional para problemas com grande número de elementos não nulos.

Além disso, o ganho em termos de economia de memória é muito grande com estrutura de lista encadeada. Por exemplo, tem-se que a média do tamanho dos problemas do repositório tratados neste trabalho é de 2.216 variáveis com 14.282 não nulos, o que gera uma média aproximada de 0,29% de densidade (porcentagem de elementos não nulos).

Assim, usando-se a estrutura indexada é necessário armazenar o total de 4.910.656 elementos, com a grande maioria sendo zero, já com estrutura de lista encadeada armazena-se somente os 14.282 não nulos. Considerando que o armazenamento de cada elemento (nulo ou não) requer 8 bytes com tipo de variável *double*, então seria necessário 39,28 Megabytes de memória para a estrutura indexada ao passo que a lista encadeada necessita apenas de 0,11 Megabytes, ou seja, uma diferença grande em termos de memória.

RESULTADOS COMPUTACIONAIS

AMBIENTE COMPUTACIONAL

Para obtenção dos resultados foi utilizado o compilador Borland, versão 5.02, instalado em um computador com as seguintes especificações: processador Intel(R) Core(TM) i5-5200U CPU 2.20 GHz e 4 Gb de memória RAM.

Os arquivos para entrada de dados são arquivos de texto, contendo os elementos não nulos da matriz. Na primeira linha do arquivo, encontra-se a ordem de grandeza do sistema e seu número de elementos não nulos. Nas próximas linhas é exibido o índice da linha, índice da coluna e o

valor de cada elemento, respectivamente. Essa estratégia economiza memória, melhora a eficiência nas operações e o acesso aos elementos a partir da linha, visto que os métodos empregados realizam operações que percorrem linha a linha da matriz. Alguns dos arquivos do repositório *online* que não apresentavam essa formatação, foram convertidos antes dos testes.

Na saída dos dados, foi impresso o nome do problema, ordem de grandeza, número de elementos não nulos, densidade, esparsidade, tempos computacionais gastos em cada etapa e o resíduo obtido com as soluções. Quando empregada a estrutura de listas encadeadas, foi contabilizado também o número de elementos não nulos presentes nas matrizes decompostas para cada método numérico. Ou seja, o número de elementos não nulos das matrizes $\boldsymbol{Q}$, $\boldsymbol{R}$ e $\boldsymbol{L}$.

MEMÓRIA E NÚMERO DE ELEMENTOS NÃO NULOS NAS MATRIZES DE DECOMPOSIÇÃO

Como relatado anteriormente, tem-se que na estrutura indexada todas as matrizes envolvidas são armazenadas em *arrays* e consomem, portanto, n^2 de memória para cada matriz. Já nas listas, apenas os elementos não nulos da matriz são armazenados, e dessa forma, o número de nós de cada lista encadeada utilizada para representar cada linha da matriz equivale ao número de elementos não nulos por linha (n_i, i=1,..., n), sendo que o somatório de n_i´s fornece a quantidade total de não nulos da matriz (nz).

Essa estratégia tende a beneficiar especialmente sistemas maiores e mais esparsos, em que o número de elementos nulos é bem maior que os não nulos. O método de Cholesky por ter suas restrições próprias, não pôde solucionar todos os problemas apresentados. Nas tabelas a seguir são apresentados os resultados em termos de uso da memória para as matrizes de decomposição envolvendo as formas de armazenagem, sendo adotadas as seguintes siglas: n: ondem de grandeza do problema; nz: número de elementos não nulos da matriz dos coeficientes; nq: número de elementos não nulos da matriz $\boldsymbol{Q}$ da decomposição QR; nr: número de elementos não nulos da matriz $\boldsymbol{R}$ da decomposição QR; nc: número de elementos não nulos da matriz $\boldsymbol{L}$ da decomposição Cholesky.

Para os primeiros testes computacionais, foi utilizado um gerador aleatório desenvolvido por Silva (2021). Logo, a Tabela 1 mostra as características destes problemas e fornece o número de elementos não nulos para cada matriz de decomposição. Nos problemas, *"tridiag"* e *"triang"* são abreviações para problemas com a matriz de coeficientes tridiagonal e triangular, respectivamente. A numeração no nome do problema indica sua dimensão.

Tabela 1 – Número de elementos não nulos dos problemas e nas matrizes de decomposição: gerador aleatório.

Problema	***n***	***nz***	***n²***	***nq***	***nr***	***nc***
tridiag100	100	298	10.000	4.246	298	-
triang100	100	5.050	10.000	100	5.050	-
tridiag500	500	1.498	250.000	33.897	1.498	-
triang500	500	125.250	250.000	500	125.250	-
tridiag1000	1.000	2.998	1.000.000	60.151	2.998	-
triang1000	1.000	500.500	1.000.000	1.000	500.500	-
tridiag1500	1.500	4.498	2.250.000	89.988	4.498	-
triang1500	1.500	1.125.750	2.250.000	1.500	1.125.750	-
tridiag2000	2.000	5.998	4.000.000	103.386	5.998	-
triang2000	2.000	2.001.000	4.000.000	2.000	2.001.000	-
tridiag3500	3.500	10.498	12.250.000	185.234	10.498	-
tridiag5000	5.000	14.998	25.000.000	257.263	14.998	-
tridiag6500	6.500	19.498	42.250.000	323.818	19.498	-
média	**1.938,46**	**293.679,54**	**7.270.769,23**	**81.775,62**	**293.678,92**	**-**

Fonte: elaborado pelos autores.

Nota-se, pelos dados obtidos, que durante a Fatoração QR de matrizes triangulares a matriz fatorada $\boldsymbol{Q}$ é uma matriz diagonal, enquanto a matriz $\boldsymbol{R}$ possui o mesmo número de elementos não nulos da matriz $\boldsymbol{A}$. Em todos os casos solucionados, as matrizes de coeficientes decompostas possuem um número de elementos não nulos menor que o número total de elementos da matriz $\boldsymbol{A}$ (n^2).

Na Tabela 2 apresenta-se a relação de elementos não nulos dos sistemas lineares e das matrizes de decomposição para os problemas do repositório tratados neste trabalho, obtidos na SuiteSparse Matrix Collection (Davis; Hus, 2011)[5].

Tabela 2 – Número de elementos não nulos dos problemas e nas matrizes de decomposição: repositório.

Problema	***n***	***nz***	***n²***	***nq***	***nr***	***nc***
*trefethen_20b**	19	83	361	19	83	19
*trefethen_20**	20	89	400	20	89	20
*bcsstk01**	48	224	2.304	2.027	952	877
bfw62a	62	450	3.844	3.470	1.715	-
*bcsstm02**	66	66	4.356	66	66	66
*trefethen_150**	150	1.095	22.500	150	1.095	150
*trefethen_200**	200	1.545	40.000	200	1.545	200
*trefethen_300**	300	2.489	90.000	300	2.489	300

[5] Repositório disponível em: https://sparse.tamu.edu. Acesso em: 14 jul. 2026.

*bcsstm06**	420	420	176.400	420	420	420
*bcsstm07**	420	3.836	176.400	420	3.836	420
*trefethen_500**	500	8.478	250.000	500	4.489	500
*trefethen_700**	700	6.677	490.000	700	6.677	700
sherman1	1.000	3.750	1.000.000	252.612	51.069	-
sherman2	1.080	23.094	1.166.400	701.066	328.551	-
*bcsstm09**	1.083	1.083	1.172.889	1.083	1.083	1.083
*bcsstm10**	1.086	22.092	1.179.396	1.086	11.589	1.086
sherman4	1.104	3.786	1.218.816	216.027	102.847	-
*1138_bus**	1.138	2.596	1.295.044	565.224	99.137	38.312
rdb1250	1.250	7.300	1.562.500	841.972	119.037	-
c-19	2.327	12.072	5.414.929	2.327	12.072	-
cavity10	2.597	76.367	6.744.409	-	-	-
sherman5	3.312	20.793	10.969.344	-	-	-
*bcsstm21**	3.600	3.600	12.960.000	3.600	3.600	3.600
c-26	4.307	19.422	18.550.249	-	-	-
c-28	4.598	17.594	21.141.604	4.508	17.594	-
sherman3	5.005	20.033	25.050.025	-	-	-
c-29	5.033	24.382	25.331.089	5.033	24.382	-
c-30	5.321	35.507	28.313.041	5.321	35.507	-
c-31	5.339	41.955	28.504.921	5.339	41.955	-
c-32	5.975	30.223	35.700.625	5.975	30.223	-
c-33	6.317	31.220	39.904.489	6.317	31.220	-
c-35	6.537	34.714	42.732.369	6.537	34.714	-
média	**2.216,06**	**14.282,34**	**9.724.022,00**	**94.011,39**	**34.572,71**	**3.183,53**

Fonte: elaborado pelos autores.

Nos problemas identificados com asterisco ao final do nome, a matriz de coeficientes é simétrica e definida positiva, de modo que o método de Cholesky pode ser utilizado somente nesses casos.

Em alguns dos problemas observados, nota-se que a decomposição gera pelo menos uma matriz diagonal ou uma matriz com mesmo número de elementos não nulos da matriz ***A*** (mantendo sua esparsidade), isso ocorre em vários problemas. Tal semelhançaa não é observada para os problemas *bcsstk01*, *bfw62a*, *sherman1*, *sherman2*, *sherman4*, *1138 bus* e *rdb1250* usando a decomposição QR, por exemplo. Para esses problemas nota-se um aumento no número de elementos não nulos, aumentando também o número de operações envolvidas na solução desses sistemas. Os problemas *cavity10*, *sherman5*, *c-26* e *sherman3* não chegaram a ser resolvidos dentro do tempo limite de teste (5 horas) usando a Decomposição QR.

PROBLEMAS DO GERADOR ALEATÓRIO

Os problemas obtidos com o gerador aleatório foram solucionados utilizando o método de Fatoração QR, tanto com estrutura de dados indexada quanto com listas encadeadas. Nesses casos, as matrizes de coeficientes geradas não são, em geral, simétricas e definidas positivas, o que inviabiliza a aplicação do método de Cholesky.

Nas próximas tabelas que serão apresentadas neste trabalho, adotam-se as seguintes abreviações para apresentação dos resultados: *TL*: tempo para leitura dos dados em segundos; *TD*: tempo para decomposição da matriz ***A*** em segundos (decomposição em ***Q*** e ***R*** na Fatoração QR e em ***C*** na Fatoração de Cholesky); *TRS*1: tempo de resolução do sistema linear 1 em segundos (sistema ortogonal na Fatoração QR ou triangular superior para Fatoraçãode Cholesky); *TRS*2: tempo de resolução do sistema linear 2 em segundos (sistema triangular superior na Fatoração QR ou triangular inferior na Fatoraçãode Cholesky); *TT*: tempo total de execução do programa em segundos; e *norma*: norma euclidiana do resíduo.

Para fins de simplificação, tempos inferiores a 0,0001 segundos foram registrados nas tabelas a seguir apenas como 0 (zero). A Tabela 3 apresenta os resultados para os problemas do gerador aleatório solucionados com o método de Fatoração QR com a estrutura de dados indexada.

Tabela 3 – Decomposição QR utilizando estrutura indexada - gerador aleatório.

Problema	*n*	*nz*	d(%)	TL	TD	TRS1	TRS2	TT	*norma*
tridiag100	100	298	2,98	0	0	0	0	0	7,54E-16
triang100	100	5050	50,5	0	0,015	0	0	0,015	1,70E-15
tridiag500	500	1498	0,6	0	0,547	0	0	0,547	1,60E-15
triang500	500	125250	50,1	0,141	0,703	0	0	0,703	1,36E-14
tridiag1000	1000	2998	0,3	0,031	9,820	0	0	9,820	2,43E-15
triang1000	1000	500500	50,05	0,453	12,800	0	0	12,800	1,80E-14
tridiag1500	1500	4498	0,2	0,063	45,500	0	0	45,500	2,91E-15
triang1500	1500	1125750	50,03	1,060	64,600	0	0,016	64,616	2,51E-14
tridiag2000	2000	5998	0,15	0,094	109,000	0	0,016	109,016	3,25E-15
triang2000	2000	2001000	50,02	1,590	161,000	0	0,015	161,015	3,35E-14
tridiag3500	3500	10498	0,08	0,234	806,000	0	0,032	806,032	4,42E-15
tridiag5000	5000	14998	0,06	0,469	3230,000	0	0,079	3230,079	5,33E-15
tridiag6500	6500	19498	0,05	0,734	8550,000	0	0,141	8550,141	6,08E-15
média	**1938,46**	**293679,53**	**19,62**	**0,376**	**999,230**	**0**	**0,023**	**999,253**	**9,12E-15**

Fonte: elaborado pelos autores.

Nota-se que para os sistemas de mesma ordem de grandeza e diferentes densidades, os problemas com matrizes triangulares tiveram um tempo total maior em relação a sistemas

lineares com matrizes tridiagonais, pois têm maior número de elementos não nulos. Assim, para os sistemas de mesma ordem de grandeza, os problemas solucionados em menos tempo foram os tridiagonais. Observe que o tempo de decomposição dessas matrizes é cerca de 99,99% do tempo total. Destaca-se que a norma do resíduo foi muito baixa para todos os problemas. A Tabela 4 apresenta agora os resultados para o mesmo método usando a estrutura de listas encadeadas.

Tabela 4 – Decomposição QR utilizando estrutura de listas encadeadas - gerador aleatório.

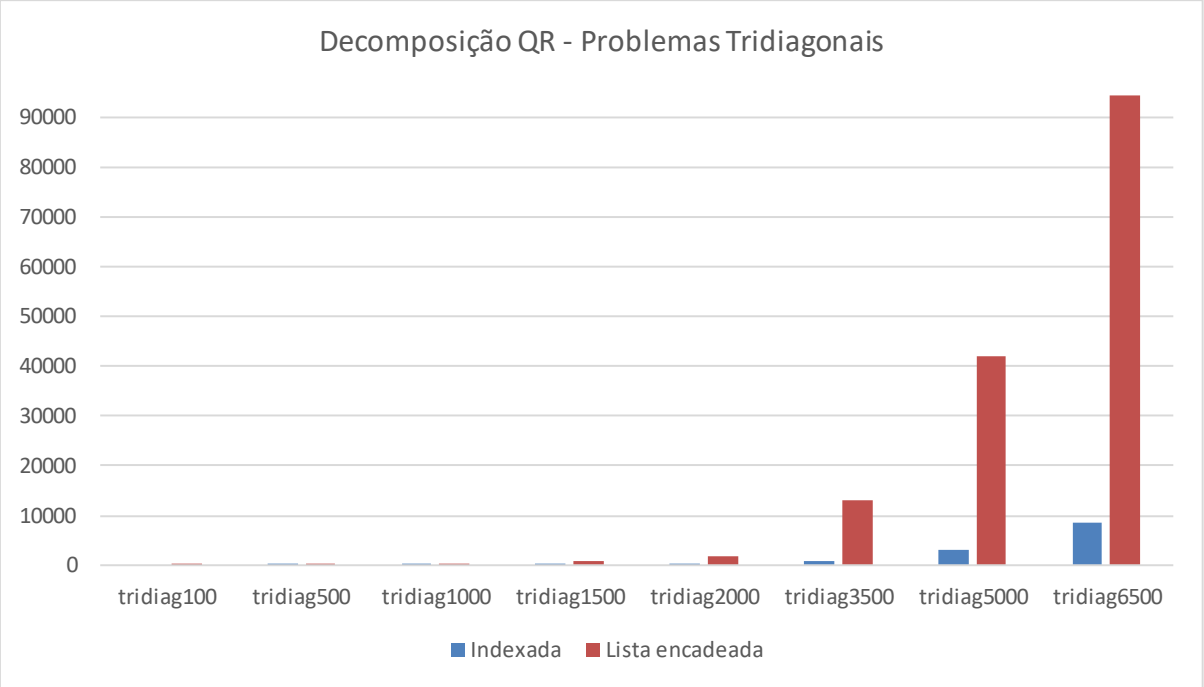
Problema	***n***	***nz***	**d(%)**	**TL**	**TD**	**TRS1**	**TRS2**	**TT**	***norma***
tridiag100	100	298	2,98	0	0,062	0	0	0,062	7,54E-16
triang100	100	5050	50,5	0	0,094	0	0	0,094	1,70E-15
tridiag500	500	1498	0,6	0	31,700	0,110	0	31,700	1,60E-15
triang500	500	125250	50,1	0,172	47,300	0	0	47,410	9,12E-15
tridiag1000	1000	2998	0,3	0,016	231,000	0,344	0	231,344	2,43E-15
triang1000	1000	500500	50,05	1,270	1000,000	0,016	0	1000,016	1,81E-14
tridiag1500	1500	4498	0,2	0,016	921,000	0,922	0	921,922	2,91E-15
triang1500	1500	1125750	50,03	4,220	5060,000	0,015	0	5060,015	2,51E-14
tridiag2000	2000	5998	0,15	0,000	1700,000	1,440	0	1701,440	3,25E-15
triang2000	2000	2001000	50,02	8,350	14700,000	0,016	0	14700,016	3,35E-14
tridiag3500	3500	10498	0,08	0,016	13100,000	6,790	0	13106,790	4,42E-15
tridiag5000	5000	14998	0,06	0,015	41900,000	14,500	0	41914,500	5,33E-15
tridiag6500	6500	19498	0,05	0,032	94300,000	23,700	0	94323,700	6,08E-15
média	**1938,46**	**293679,53**	**19,62**	**1,086**	**13307,012**	**3,681**	**0,000**	**13310,693**	**8,79E-15**

Fonte: elaborado pelos autores.

A Tabela 4 apresenta tempos computacionais significativamente maiores que os anteriores utilizando a estrutura indexada. Na média, o método gastou aproximadamente 33% mais tempo para resolver os mesmos problemas utilizando-se a estrutura de listas encadeadas. Observa-se, como no resultado anterior, que o tempo de decomposição é cerca de 99,99% do tempo total de resolução dos problemas. Com relação à norma dos resíduos, nota-se um resultado semelhante aquele obtido com a estrutura indexada.

A Figura 2 apresenta uma comparação dos tempos para o método com as duas estruturas considerando os problemas tridiagonais.

Figura 2 – Comparação do tempo da decomposição QR entre estruturas indexadas e listas encadeadas.



Fonte: elaborado pelos autores.

Estes resultados indicam que o uso de listas encadeadas apresentou desempenho inferior para esta classe de problemas. O tempo de resolução foi superior em todos os casos do gerador aleatório, dado que os problemas possuem dimensões moderadas e densidade média de aproximadamente 19,62%.

PROBLEMAS DO REPOSITÓRIO

Diversos problemas do repositório online *SuiteSparse Matrix Collection* foram resolvidos por ambos os métodos numéricos com as estruturas de dados discutidas anteriormente. Os resultados a seguir detalham o desempenho do método de Fatoração QR, comparando a utilização da estrutura indexada com a de listas encadeadas.

A Tabela 5 apresenta os resultados para os problemas oriundos do repositório que foram resolvidos com Fatoração QR usando a estrutura indexada.

Tabela 5 – Decomposição QR utilizando estrutura indexada - repositório.

Problema	***n***	***nz***	**d(%)**	**TL**	**TD**	**TRS1**	**TRS2**	**TT**	***norma***
trefethen_20b	19	83	22,99	0	0	0	0	0	1,33E-15
trefethen_20	20	89	22,25	0	0	0	0	0	1,54E-15
bcsstk01	48	224	9,72	0	0	0	0	0	0
bfw62a	62	450	11,71	0	0	0	0	0	0

bcsstm02	66	66	1,51	0	0,015	0	0	0,015	1,78E-15
trefethen_150	150	1095	4,87	0	0,063	0,015	0	0,078	5,77E-14
trefethen_200	200	1545	3,86	0	0,188	0	0	0,188	1,59E-13
trefethen_300	300	2489	2,76	0	0,625	0	0	0,625	6,29E-13
bcsstm06	420	420	0,24	0,015	0,313	0	0	0,313	9,14E-14
bcsstm07	420	3836	2,17	0	1,050	0,016	0	1,066	2,22E-12
trefethen_500	500	8478	1,79	0,015	2,750	0,016	0	2,766	1,32E-12
trefethen_700	700	6677	1,36	0,016	10,000	0,016	0	10,016	3,01E-12
sherman1	1000	3750	0,37	0,032	31,200	0,015	0	31,215	0
sherman2	1080	23094	1,98	0,047	57,700	0,015	0	57,715	1,34E-01
bcsstm09	1083	1083	0,09	0,015	10,700	0,016	0,015	10,731	3,73E-13
bcsstm10	1086	22092	0,98	0,032	47,100	0,140	0,016	47,256	7,77E-11
sherman4	1104	3786	0,31	0,031	31,800	0,031	0,031	31,862	1,84E-11
1138_bus	1138	2596	0,2	0,032	31,700	0,015	0,000	31,715	0
rdb1250	1250	7300	0,47	0,032	148,000	0,015	0,016	148,031	0
c-19	2327	12072	0,22	0,125	189,000	0,219	0,125	189,344	8,67E-17
cavity10	2597	76367	1,13	0,188	2640,000	0	0,125	2640,125	5,98E-23
sherman5	3312	20793	0,19	0,234	2070,000	0	0,297	2070,297	1,61E-08
bcsstm21	3600	3600	0,03	0,234	879,000	0,265	0,188	879,453	3,26E-12
c-26	4307	19422	0,1	EM	EM	EM	EM	EM	EM
c-28	4598	17594	0,08	EM	EM	EM	EM	EM	EM
sherman3	5005	20033	0,08	EM	EM	EM	EM	EM	EM
c-29	5033	24382	0,1	EM	EM	EM	EM	EM	EM
c-30	5321	35507	0,12	EM	EM	EM	EM	EM	EM
c-31	5339	41955	0,15	EM	EM	EM	EM	EM	EM
c-32	5975	30223	0,08	EM	EM	EM	EM	EM	EM
c-33	6317	31220	0,08	EM	EM	EM	EM	EM	EM
c-35	6537	34714	0,08	EM	EM	EM	EM	EM	EM
média	**2216,06**	**14282,34**	**2,87**	**0,045**	**265,416**	**0,034**	**0,035**	**265,486**	**5,00E-03**

Fonte: elaborado pelos autores.

Como era esperado, observa-se que o tempo de decomposição representa, na média geral, cerca de 99,99% do tempo total de execução. Nota-se que, para alguns problemas, embora a ordem de grandeza seja maior, o número de elementos não nulos é inferior ao de seus antecessores. Isso justifica um tempo total de resolução reduzido em comparação a problemas como *sherman5* e *bcsstm21*, bem como para matrizes diagonais, a exemplo de *bcsstm06*, *bcsstm09* e *bcsstm21*.

A norma dos resíduos das soluções apresentou-se satisfatória para a maioria dos problemas, com exceção do *sherman2,* que registrou 1,34E-01. Em geral, nota-se que o tempo total

de resolução dos sistemas cresce a medida que aumenta a ordem de grandeza ou no número de elementos não nulos.

É importante destacar que problemas com ordem de grandeza superior a 3600 não foram solucionados com o método de Fatoração QR utilizando-se a estrutura de dados indexada, devido a ocorrências de estouro de memória (EM) (*overflow*). A Tabela 6 apresenta os resultados obtidos utilizando decomposição QR com a estrutura de listas encadeadas.

Tabela 6 – Decomposição QR utilizando estrutura de listas encadeadas - repositório.

Problema	***n***	***nz***	**d(%)**	**TL**	**TD**	**TRS1**	**TRS2**	**TT**	***norma***
trefethen_20b	19	83	22,99	0	0	0	0	0	1,33E-15
trefethen_20	20	89	22,25	0,016	0,016	0	0,016	0,032	1,54E-15
bcsstk01	48	224	9,72	0	0	0	0	0	0
bfw62a	62	450	11,71	0	0,032	0	0,032	0,064	0
bcsstm02	66	66	1,51	0	0	0	0	0	1,78E-15
trefethen_150	150	1095	4,87	0	0,063	0	0,063	0,126	5,77E-14
trefethen_200	200	1545	3,86	0	0,140	0	0,140	0,28	1,59E-13
trefethen_300	300	2489	2,76	0	0,484	0	0,484	0,968	6,29E-13
bcsstm06	420	420	0,24	0,016	1,080	0	1,080	2,16	9,14E-13
bcsstm07	420	3836	2,17	0	1,890	0	1,890	3,78	2,22E-12
trefethen_500	500	8478	1,79	0	2,450	0	2,470	4,92	1,32E-12
trefethen_700	700	6677	1,36	0	8,470	0	8,470	16,94	3,01E-12
sherman1	1000	3750	0,37	0	2010,100	0	2020,300	4030,4	0
sherman2	1080	23094	1,98	0,016	7200,200	0	7220,100	14420,3	1,34E-01
bcsstm09	1083	1083	0,09	0	18,600	0	18,300	36,9	3,73E-13
bcsstm10	1086	22092	0,98	0,016	39,700	0	39,400	79,1	7,77E-11
sherman4	1104	3786	0,31	0	3120,100	0	3120,300	6240,4	1,84E-11
1138_bus	1138	2596	0,2	0,031	5680,800	0	5690,200	11371	0
rdb1250	1250	7300	0,47	0,016	10500,70	0	10600,50	21101,2	0
c-19	2327	12072	0,22	0,016	223,200	0	223,100	446,3	8,67E-17
cavity10	2597	76367	1,13	-	-	-	-	-	-
sherman5	3312	20793	0,19	-	-	-	-	-	-
bcsstm21	3600	3600	0,03	0	692,400	0	692,900	1385,300	3,26E-12
c-26	4307	19422	0,1	-	-	-	-	-	-
c-28	4598	17594	0,08	0,015	1660,200	0	1660,500	3320,700	0
sherman3	5005	20033	0,08	-	-	-	-	-	-
c-29	5033	24382	0,1	0,031	2430,500	0	2440,900	4871,400	0
c-30	5321	35507	0,12	0,109	2760,300	0	2760,500	5520,800	0
c-31	5339	41955	0,15	0,047	3130,500	0	3130,100	6260,600	0
c-32	5975	30223	0,08	0,094	3550,100	0	3550,400	7100,500	0
c-33	6317	31220	0,08	0,047	4070,700	0	4070,100	8140,800	0
c-35	6537	34714	0,08	0,032	5210,100	0	5210,200	10420,300	0
média	**2216,06**	**14282,34**	**2,87**	**0,017**	**1868,315**	**0**	**1873,658**	**3741,973**	**4,00E-03**

Fonte: elaborado pelos autores.

A ausência de tempos reportados para determinados casos indica que o método utilizado não resolveu o problema dentro do limite pré-estabelecido de 5 horas.

Observa-se que os tempos mais elevados para resolver o sistema triangular superior com a matriz ***R*** (TRS2), ocorrem em problemas altamente esparsos (densidade inferior a 2%), porém com um elevado número de elementos não nulos em *nr*, conforme apresentado na Tabela 2.

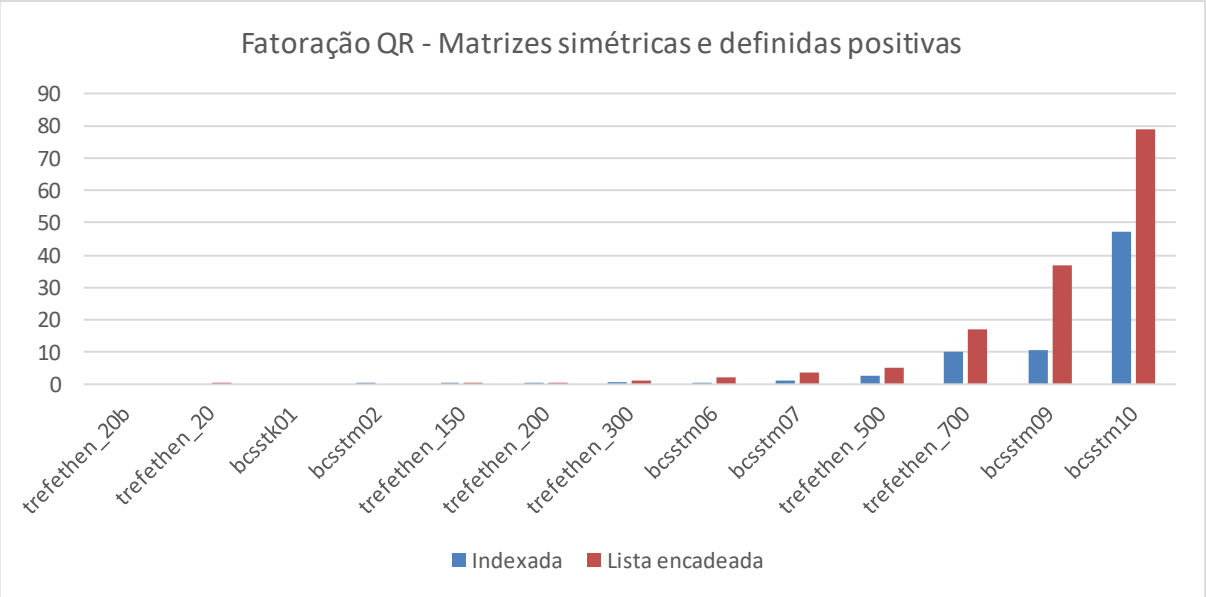
Nota-se, novamente, um aumento nos tempos de execução proporcional ao aumento da ordem dos sistemas lineares. As exceções ocorrem em matrizes diagonais ou em casos onde a ordem de grandeza aumenta, mas o número de elementos não nulos diminui.

Comparando os resultados apresentados nas Tabelas 5 e 6, nota-se um aumento significativo no tempo médio total de resolução dos problemas, sendo este aproximadamente 14 vezes superior.

Mesmo com o aumento no tempo de resolução de muitos problemas usando a estrutura de listas encadeadas, problemas com ordem de grandeza superior a 3600 não foram solucionados pela Fatoração QR com a estrutura indexada. No entanto, a estrutura de listas encadeadas permitiu a resolução desses sistemas maiores, ainda que com um custo computacional elevado, mas com resíduos nulos.

A Figura 3 apresenta o comparativo entre os tempos de resolução para sistemas lineares cujas matrizes são simétricas e definidas positivas, com exceção dos dois maiores casos (*1138bus* e *bcsstm21*) para melhor visualização da escala.

Figura 3 – Comparação do tempo de resolução da decomposição QR entre estruturas indexadas e listas encadeadas.



Fonte: elaborado pelos autores.

Na Figura 3, nota-se que a Fatoração QR com a estrutura de dados de listas encadeadas gastou um tempo maior quando comparado a estrutura indexada, mas no geral esta diferença foi menor com relação a vários outros problemas como pode ser observado na Tabela 6. Por exemplo, para os dois últimos problemas apresentados na Figura 3 o método com listas encadeadas precisou de 36,9 e 79,1 segundos para resolver os respectivos sistemas lineares, ao passo que o método numérico com estrutura indexada gastou 10,7 e 47,2 segundos para resolvê-los. Já no caso do problema *sherman4* em que a matriz dos coeficientes não tem nenhuma propriedade, a Fatoração QR com listas encadeadas precisou de 6.204,4 segundos para encontrar a solução e o método com a estrutura indexada exigiu apenas 31,8 segundos.

A partir de agora, apresentam-se os resultados numéricos para os problemas do repositório com as matrizes simétricas e definidas positivas sendo resolvidos com a Fatoração de Cholesky com as duas estruturas de dados. A Tabela 7 mostra os resultados obtidos com o método de Cholesky com a estrutura de dados indexada para os problemas do repositório.

Tabela 7 – Decomposição Cholesky utilizando estrutura indexada - problemas do repositório.

Problema	***n***	***nz***	**d(%)**	**TL**	**TD**	**TRS1**	**TRS2**	**TT**	***norma***
trefethen_20b	19	83	22,99	0	0	0	0	0	1,65E+00
trefethen_20	20	89	22,25	0	0	0	0	0	1,93E+00
bcsstk01	48	224	9,72	0	0	0	0	0	0,00E+00
bcsstm02	66	66	1,51	0	0	0	0	0	1,32E-14
trefethen_150	150	1095	4,87	0	0	0	0	0	1,26E+01
trefethen_200	200	1545	3,86	0	0	0	0	0	3,20E+01
trefethen_300	300	2489	2,76	0	0,016	0	0	0,016	9,54E+01
bcsstm06	420	420	0,24	0	0,047	0	0	0,047	2,79E-12
bcsstm07	420	3836	2,17	0	0,047	0	0	0,047	1,59E+04
trefethen_500	500	8478	1,79	0,016	0,078	0	0,015	0,093	1,81E+02
trefethen_700	700	6677	1,36	0,016	0,234	0	0	0,234	2,82E+02
bcsstm09	1083	1083	0,09	0,031	0,86	0	0,015	0,875	1,36E-12
bcsstm10	1086	22092	0,98	0,015	0,891	0	0,031	0,922	7,73E+04
1138_bus	1138	2596	0,2	0,016	1,03	0	0	1,03	0,00E+00
bcsstm21	3600	3600	0,03	0,140	31,4	0,125	0,187	31,712	7,84E-12
média	**650,0**	**3624,86**	**4,988**	**0,015**	**2,306**	**0,008**	**0,016**	**2,331**	**6,25E+03**

Fonte: elaborado pelos autores.

O maior tempo total registrado entre os problemas foi de 31,8 segundos, tempo significativamente inferior aos 879,4 segundos observados na Fatoração QR para o problema *bcsstm21*, com ótima precisão numérica na solução.

Em diversos casos nota-se que a norma dos resíduos apresentou-se excessivamente elevada. Nestas situações, as soluções não podem ser consideradas válidas, como observado nos problemas das classes *trefethen*, o *bcsstm07* e *bcsstm10*. Por outro lado, obteve-se bons resultados para os problemas: *bcsstk01*, *bcsstm02*, *bcsstm06*, *bcsstm09*, *1138_bus* e *bcsstm21*.

Como observado anteriormente, o tempo total cresce à medida que a ordem de grandeza ou o número de elementos não nulos do problema aumenta. Ressalta-se, ainda, que o tempo de decomposição corresponde a aproximadamente 98,9% do tempo total.

A Tabela 8 apresenta os resultados obtidos utilizando o mesmo método, porém com a estrutura de dados em listas encadeadas.

Tabela 8 – Decomposição Cholesky com estrutura de listas encadeadas - problemas do repositório.

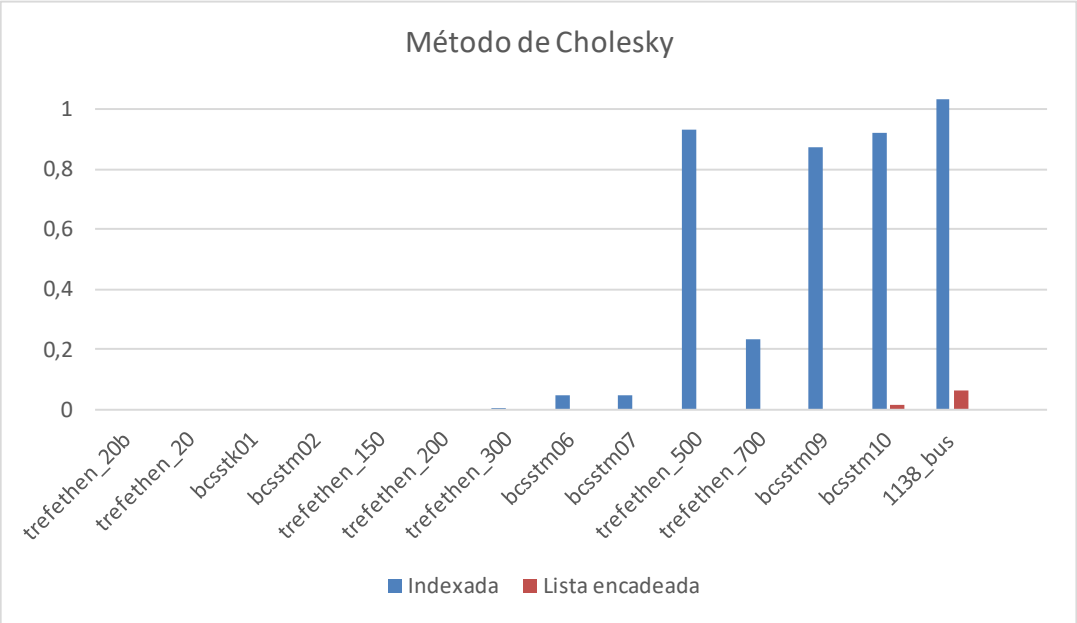
Problema	***n***	***nz***	**d(%)**	**TL**	**TD**	**TRS1**	**TRS2**	**TT**	***norma***
trefethen_20b	19	83	22,99	0	0	0	0	0	1,65E+00
trefethen_20	20	89	22,25	0	0	0	0	0	1,93E+00
bcsstk01	48	224	9,72	0	0	0	0	0	0,00E+00
bcsstm02	66	66	1,51	0	0	0	0	0	1,33E-14
trefethen_150	150	1095	4,87	0	0	0	0	0	1,26E+01
trefethen_200	200	1545	3,86	0	0	0	0	0	3,20E+01
trefethen_300	300	2489	2,76	0	0	0	0	0	9,54E+01
bcsstm06	420	420	0,24	0	0	0	0	0	2,79E-12
bcsstm07	420	3836	2,17	0	0	0	0	0	1,59E+04
trefethen_500	500	8478	1,79	0	0	0	0	0	1,81E+02
trefethen_700	700	6677	1,36	0	0	0	0	0	2,82E+02
bcsstm09	1083	1083	0,09	0	0	0	0	0	1,36E-12
bcsstm10	1086	22092	0,98	0,016	0,016	0	0	0,016	7,72E+04
1138_bus	1138	2596	0,2	0,016	0,062	0	0,063	0,062	0,00E+00
bcsstm21	3600	3600	0,03	0,016	0,125	0	0	0,125	7,84E-12
média	**650,0**	**3624,86**	**4,988**	**0,0032**	**0,013**	**0**	**0,004**	**0,013**	**6,24E+03**

Fonte: Elaboração própria

Novamente, o tempo total de resolução para a maioria dos problemas foi muito próximo de zero. Em alguns casos, a norma do resíduo das soluções foi bastante elevada (assemelhando-se aos resultados da estrutura indexada) e, nestas situações, a solução encontrada pelo método não é considerada válida, visto que está muito distante da solução analítica. Ressalta-se que, no geral, o desempenho do método Cholesky com a estrutura de listas encadeadas superou a estrutura indexada. Para o problema de maior dimensão (*bcsstm21*), enquanto a estrutura indexada demandou 31,7 segundos para encontrar a solução, já a estrutura de listas encadeadas

necessitou de apenas 0,125 segundos para obter o resultado com a mesma precisão (norma do resíduo). A Figura 4 apresenta o comprativo entre alguns tempos de resolução para problemas em que o método de Cholesky forneceu boas soluções utilizando ambas estruturas.

Figura 4 – Tempo da decomposição de Cholesky com estrutura indexada e listas encadeadas.



Fonte: elaborado pelos autores.

Para os problemas em que o método de Cholesky forneceu solução válida, em termos de estrutura de dados, utilizando as listas encadeadas o tempo de resolução foi bem menor que a estrutura indexada. Ao comparar estes resultados com os obtidos via Fatoração QR, observa-se uma redução ainda mais expressiva no tempo total de resolução dos problemas.

ANÁLISE DE RESULTADOS

Dentre os problemas estudados, observa-se que o estouro de memória (*overflow*) ocorreu em sistemas lineares com ordem de grandeza superior a 3.600, ao utilizar a Fatoração QR com estrutura indexada. No entanto, o mesmo não ocorreu com o emprego da estrutura de listas encadeadas. De modo geral, para os problemas em que a matriz dos coeficientes é definida positiva e simétrica, a Fatoração de Cholesky com listas encadeadas apresentou um excelente desempenho à versão com estrutura indexada. Assim, a Fatoração de Cholesky apresentou soluções de alta precisão (norma de resíduo reduzida) em tempos computacionais menores que os da Fatoração QR.

A estrutura de listas encadeadas mostrou-se mais eficiente que a indexada quando aplicada à Fatoração de Cholesky. No caso da Fatoração QR, embora o uso de listas tenha

permitido solucionar um maior número de problemas, o tempo de resolução foi elevado se comparado ao obtido pelo método de Cholesky.

CONSIDERAÇÕES FINAIS

Inúmeras aplicações científicas envolvem a resolução de sistemas lineares esparsos, muitas vezes com ordens de grandezas que atinge de milhões de elementos. A resolução eficiente destes sistemas lineares dependem da escolha criteriosa de um método numérico e de uma estrutura de dados adequada, que permitam alcançar soluções de alta precisão (baixa norma do resíduo) e no menor tempo possível.

Neste trabalho, foram solucionados sistemas lineares esparsos utilizando os métodos de Fatoração QR e de Cholesky, comparando-se o desempenho entre as estruturas de dados indexada e de lista encadeadas.

O uso da estrutura de dados de listas encadeadas associada ao método de Fatoração QR permitiu fornecer soluções a problemas que antes não haviam sido solucionados com a estrutura indexada devido ao estouro de memória (como problemas com ordem superior a 3600). No entanto, observou-se que, para os problemas nos quais ambas as estruturas foram bem sucedidas, a estrutura indexada apresentou desempenho temporal superior.

Por outro lado, o método de Cholesky resolveu um número menor de problemas (apenas com matrizes simétricas e positivas definidas). Para esse método, as listas encadeadas forneceram soluções em menor tempo computacional do que a estrutura indexada. Ressalta-se também que algumas soluções obtidas com a Fatoração de Cholesky foram invalidadas devido à norma do resíduo ser muito alta. Além do tempo de resolução, observou-se que a precisão numérica não sofreu alterações significativas apenas pela troca da estrutura de dados.

De forma geral, a estrutura de listas encadeadas se mostrou uma alternativa eficiente para armazenamento de dados de sistemas lineares esparsos, porém com um tempo computacional elevado quando se utiliza a Decomposição QR. Assim, identificou-se a necessidade de melhoramento do algoritmo que faz a resolução do TRS2, pois visto que o tempo gasto nesta rotina é praticamente o mesmo tempo utilizado para fazer a decomposição da matriz original $\boldsymbol{A}$ em $\boldsymbol{Q}$ e $\boldsymbol{R}$.

Portanto, para trabalhos futuros, propõe-se principalmente o aprimoramento de rotinas e o refinamento das soluções obtidas para problemas maiores do repositório. Sugere-se, ainda, a investigação de outros métodos numéricos, como a Decomposição LU com a estrutura de dados de listas encadeadas.

REFERÊNCIAS

ARENALES, S.; DAREZZO, A. **Cálculo numérico: aprendizagem com apoio de software.** 2. ed. São Paulo: Cengage Learning, 2015.

BOLDRINI, J. L.; COSTA, S. I. R.; FIGUEIREDO, V. L.; WETZLER, H. G. **Álgebra linear.** 3. ed. São Paulo: Harper & Row do Brasil, 1980.

COLLA, E. C. **Aplicação de técnicas de fatoração de matrizes esparsas para inferência em redes bayesianas.** 2007. Dissertação (Mestrado em Ciências da Computação) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2007.

COSTA, E. C. **Estudo de fluxo de potência com aplicação de métodos diretos na resolução de sistemas de equações lineares.** 2008. Dissertação (Mestrado Profissional) – Instituto de Matemática, Estatística e Computação Científica, Universidade Estadual de Campinas, Campinas, 2008.

CUNHA, M. C. C. **Métodos numéricos.** 2. ed. Campinas: Editora da Unicamp, 2000.

DAVIS, T. A.; HU, Y. The University of Florida Sparse Matrix Collection. **ACM Transactions on Mathematical Software,** New York, v. 38, n. 1, art. 1, p. 1-25, 2011. DOI: https://doi.org/10.1145/2049662.2049663.

JALLES, D. R. **Métodos de decomposição para sistemas lineares com estrutura de dados de listas encadeadas.** 2025. Dissertação (Mestrado em Modelagem Computacional em Ciência e Tecnologia) – Universidade Federal Fluminense, Volta Redonda, 2025.

PESCADOR, A.; POSSAMAI, J. P.; POSSAMAI, C. R. Aplicação de álgebra linear na engenharia. In: CONGRESSO BRASILEIRO DE EDUCAÇÃO EM ENGENHARIA, 39., 2011, Blumenau. **Anais [...]**. Blumenau: FURB, 2011. p. 1-9. Disponível em: https://admin.abenge.org.br/cobenge/legado/arquivos/8/sessoestec/art2127.pdf. Acesso em: 14 jul. 2026.

RUGGIERO, M. G.; LOPES, V. L. R. **Cálculo numérico: aspectos teóricos e computacionais.** 2. ed. São Paulo: Pearson, 2000.

SILVA, L. H. **Métodos iterativos para sistemas lineares: pré-condicionadores, estruturas de dados e outras técnicas.** 2021. Dissertação (Mestrado em Modelagem Computacional em Ciência e Tecnologia) – Universidade Federal Fluminense, Volta Redonda, 2021.

SUITESPARSE MATRIX COLLECTION. **SuiteSparse Matrix Collection:** formerly the University of Florida Sparse Matrix Collection. [S. l.], [s. d.]. Disponível em: https://sparse.tamu.edu/. Acesso em: 10 set. 2025.

TREFETHEN, L. N.; BAU, D. **Numerical linear algebra.** Philadelphia: Society for Industrial and Applied Mathematics, 1997.

ZIVIANI, N. **Projeto de algoritmos com implementações em Pascal e C.** 4. ed. São Paulo: Pioneira, 1999.