Universidade Federal Fluminense

DÉBORA REZENDE JALLES

Métodos de Decomposição para Sistemas Lineares com Estrutura de Dados de Listas Encadeadas

VOLTA REDONDA

2025

DÉBORA REZENDE JALLES

Métodos de Decomposição para Sistemas Lineares com Estrutura de Dados de Listas Encadeadas

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Modelagem Computacional.

Orientador:

Ricardo Silveira Sousa

Coorientador:

Wagner Rambaldi Telles

Universidade Federal Fluminense

VOLTA REDONDA

2025

Ficha catalográfica automática - SDC/BEM
Gerada com informações fornecidas pelo autor

J26m Jalles, Débora Rezende
Métodos de Decomposição para Sistemas Lineares com Estrutura de Dados de Listas Encadeadas / Débora Rezende Jalles. - 2025.
89 f.: il.

Orientador: Ricardo Silveira Sousa.
Coorientador: Wagner Rambaldi Telles.
Dissertação (mestrado)-Universidade Federal Fluminense, Escola de Engenharia Industrial e Metalúrgica de Volta Redonda, Volta Redonda, 2025.

1. Listas encadeadas. 2. Fatoração LU. 3. Fatoração QR. 4. Fatoração de Cholesky. 5. Produção intelectual. I. Sousa, Ricardo Silveira, orientador. II. Telles, Wagner Rambaldi, coorientador. III. Universidade Federal Fluminense. Escola de Engenharia Industrial e Metalúrgica de Volta Redonda. IV. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

Métodos de Decomposição para Sistemas Lineares com Estrutura de Dados de Listas Encadeadas

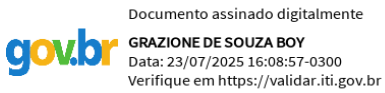
Débora Rezende Jalles

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Modelagem Computacional.

Aprovada por:

Prof. Ricardo Silveira Sousa, D.Sc. / MCCT-UFF(Presidente)

Prof. Gustavo Benitez, D.Sc. / MCCT-UFF



Prof. Grazione de Souza, D.Sc. / IPRJ-UERJ

Volta Redonda, 24 de Junho de 2025.

Agradecimentos

Aos meus pais, Denilson e Vanderléa, e ao meu companheiro, Vinícius, pelo suporte de sempre.

Ao meu orientador, professor Ricardo Silveira Sousa, por toda a paciência e, principalmente, pelo tempo dedicado a este trabalho.

Ao meu coorientador, professor Wagner Rambaldi, pelo apoio e contribuição.

Agradeço também aos demais professores, aos membros das bancas de qualificação e defesa, à secretária Camila e a todos que, direta ou indiretamente, contribuíram para tornar este sonho possível.

Resumo

Frequentemente problemas de Matemática, Física, Engenharia, Economia, Química e áreas afins envolvem a resolução de sistemas lineares, os quais podem ter dimensões superiores a 1 milhão de equações. Nesses casos, as matrizes de coeficientes destes sistemas em consideração são esparsas. Para problemas de solução única, uma alternativa são os métodos numéricos diretos, pois apresentam uma solução muito próxima da exata (salvos erros de arredondamento) em um número finito de passos. Nesse ponto, destacam-se os métodos que envolvem a fatoração da matriz de coeficientes. No entanto, ao implementá-los é essencial o uso de uma estrutura de dados apropriada, a fim de evitar falhas de alocação de memória e reduzir o tempo de processamento. Tendo em vista estes fatores, neste trabalho foram pesquisados e implementados os métodos de Fatoração LU, QR e de Cholesky, com uso da estrutura de dados indexada e de listas encadeadas. Na estrutura de listas encadeadas, os elementos nulos não foram armazenados, economizando memória e evitando operações desnecessárias. Essa estratégia tornou possível a resolução de problemas da ordem de milhares de elementos. Dentre os métodos e estruturas utilizados, os melhores resultados em termos de norma do resíduo gerado e tempo para a resolução dos problemas foram obtidos com o método de decomposição LU e com a estrutura de dados de listas encadeadas.

Abstract

Problems in Mathematics, Physics, Engineering, Economics, Chemistry, and related fields frequently involve solving linear systems, which can have dimensions exceeding 1 million equations. In such cases, the coefficient matrices of these systems are sparse. For problems with a unique solution, one alternative is to use direct numerical methods, as they provide solutions very close to the exact ones (except for rounding errors) in a finite number of steps. At this point, methods involving the factorization of the coefficient matrix stand out. However, when implementing these methods, it is essential to use an appropriate data structure in order to avoid memory allocation failures and to reduce processing time. Considering these factors, this work researched and implemented the LU, QR, and Cholesky factorization methods using both indexed data structures and linked lists. In the linked list structure, zero elements were not stored, saving memory and avoiding unnecessary operations. This strategy made it possible to solve problems involving thousands of elements. Among the methods and data structures used, the best results in terms of residual norm and solution time were obtained with the LU decomposition method and the linked list data structure.

Palavras-chave

1. Listas encadeadas
2. Fatoração LU
3. Fatoração QR
4. Fatoração de Cholesky

Glossário

CSR	:	*Compressed Sparse Row*
CSC	:	*Compressed Sparse Colunm*
CDS	:	*Compressed Diagonal Storage*
FGS	:	*Forward Gauss-Seidel*
BGS	:	*Backward Gauss-Seidel*
SGS	:	*Symmetric Gauss-Seidel*
FSOR	:	*Forward Successive Over Relaxation*
BSOR	:	*Backward Successive Over Relaxation*
SSOR	:	*Symmetric Successive Over Relaxation*
GPS	:	*Global Positioning System*
UFF	:	Universidade Federal Fluminense
URL	:	*Uniform Resource Locator*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Por definição, um sistema linear é um conjunto de equações que apresentam as mesmas variáveis. Resolvê-lo implica em determinar um conjunto de valores que satisfaçam a todas as equações simultaneamente. De uma forma geral, esses sitemas podem ter uma, infinitas ou nenhuma solução. Para sistemas lineares quadrados (ou sistemas com o mesmo número de linhas e colunas) em que a matriz de coeficiente das equações é composta por vetores linearmente independentes (ou seja, que um vetor de coeficientes não pode ser escrito como uma combinação linear de outros), admite-se que o sistema possui apenas uma solução.

Devido à diversidade de aplicações das técnicas de solução de sistemas de equações em diferentes áreas das Ciências Exatas e das Engenharias, é possível encontrar na literatura uma extensa bibliografia sobre o assunto. Por exemplo, em Colla (2007), fez-se uso do método de fatoração Cholesky aplicado a matrizes esparsas para construir um algoritmo para inferência em Redes Bayesianas.

Já Costa (2008) apresentou um estudo sobre o fluxo de potências usando métodos diretos (fatoração **LU**, triangulação de Gauss, escalonamento de Gauss-Jordan e fatoração Cholesky) para solução dos sistemas de equações. Destaca-se que Pescador, Possamai e Possamai (2011) resaltaram a importância do estudo dos sistemas lineares nos cursos de engenharia por meio das aplicações em circuitos elétricos, balanceamento de equações químicas e na construção de estruturas metálicas. Por sua vez, Levorato (2017) discutiu sobre a importância e aplicação dos sistemas lineares em circuitos elétricos, no balanceamento de equações químicas, nos modelos econômicos de Leontief e no funcionamento do *Global Positioning System* (GPS).

1.1 Justificativa

Para problemas que possuem apenas uma solução possível, uma alternativa de resolução são os métodos diretos, que fornecem o valor exato da solução, a menos de erros de arredondamento, como a eliminação de Gauss, fatoração **LU**, a fatoração **QR**, fatoração Cholesky e outros (ARENALES e DAREZZO, 2015; CUNHA, 2000). Cada método numérico possui uma complexidade em seu algoritmo, ou seja, um número de operações que devem ser feitas para alcançar a solução. Consequentemente, o tempo de execução para resolvê-los é diferente. Nesse ponto, destacam-se os métodos que envolvem a fatoração da matriz de coeficientes utilizada, sendo eles alvo desse estudo.

Em contrapartida à escolha do método, o uso de uma estrutura de dados apropriada mostra-se essencial para um bom desempenho computacional, já que os dados armazenados têm impacto na memória e no tempo de processamento. Com a estrutura indexada, por exemplo, tratando de matrizes há a necessidade de armazenar todos os elementos (mesmo que muitos deles sejam nulos) (WEISS, 2013). Isso gera um uso de memória que muitas vezes pode causar estouro de *buffer*.

Diferentemente da estrutura indexada, uma lista encadeada é uma estrutura formada de nós ordenados. Nessa estrutura, o espaço reservado na memória refere-se a determinado nó adcionado (WEISS, 2013). Dessa forma, é possível não salvar elementos nulos. Essa estratégia foi utilizada aqui para reduzir a utilização de memória e evitar operações desnecessárias com elementos nulos.

1.2 Objetivos gerais

Nesse trabalho, buscou-se estudar, pesquisar e implementar as fatorações **LU**, **QR** e de Cholesky utilizando duas formas de armazenar e manipular os dados: estrutura indexada e listas encadeadas. Nestas, os elementos (das matrizes de coeficiente) com valores nulos não foram armazenados a fim de poupar memória, possibilitando resolver problemas maiores (evitando o estouro de *buffer*) e reduzindo o tempo de execução.

1.3 Objetivos específicos

Dentre os objetivos específicos, tem-se:

- Estudar e pesquisar métodos de resolução e estrutura de dados no armazenamento de matrizes;
- Implementar os programas envolvendo os métodos numéricos de fatoração **LU**, fatoração **QR** e fatoração de Cholesky com estrutura de dados indexada e como listas encadeadas;
- Validar os programas comparando os resultados encontrados aos obtidos por meio do *software SCILAB* (2023)(amplamente utilizado com a mesma finalidade);
- Resolver os problemas propostos, obtidos a partir de um gerador aleatório ou retirados de repositórios *online*, e compará-los quanto ao método numérico e estrutura de dados empregada em termos de tempo computacional e norma do resíduo obtido;
- Apresentar vantagens e desvantagens na escolha de determinados métodos e estruturas para um conjunto de problemas.

Espera-se ainda que esse trabalho contribua para o estudo de métodos numéricos e estrutura de dados apropriadas para solução de sistemas lineares esparsos.

1.4 Organização do texto

O capítulo 2 é dedicado a uma revisão bibliográfica sobre os métodos númericos utilizados na solução de sistemas de equações, destacando a fatoração **LU**, fatoração **QR** e fatoração de Cholesky. Em seguida, no capítulo 3 são discutidos temas como: esparsidade, estrutura de dados indexada e de listas encadeadas e suas respectivas implementações. O capítulo 4 apresenta os problemas estudados neste trabalho oriundos de um gerador aleatório e um repositório *online*. No capítulo 5 estão os resultados em termos de tempo de resolução e norma de resíduo da solução para cada método numérico estudado usando as estruturas de dados indexada e de listas encadeadas. Por fim, no capítulo 6 temos a conclusão e propostas para trabalhos futuros.

Capítulo 2

Sistemas lineares e métodos numéricos

Este capítulo apresenta, brevemente, trabalhos que ressaltam a importância de sistemas lineares em estudos nas Ciências Exatas e nas Engenharias; as formas de representá-los, diferenças entre os métodos númericos diretos e iterativos e os métodos estudados nessa dissertação (fatoração **LU**, fatoração **QR** e fatoração Cholesky) com seus respectivos algoritmos.

2.1 Definições

Dado um sistema do tipo $\mathbf{Ax} = \mathbf{b}$, em que $\mathbf{A} = (a_{ij})$, com $i = 1, \ldots, n$; $j = 1, \ldots, n$; $\mathbf{x} \in \mathbb{R}$ e $\mathbf{b} = (b_i)$ com $i = 1, \ldots, n$; é possível representá-lo na forma matricial como:

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \\ \vdots \\ b_n \end{pmatrix} \tag{2.1}$$

ou ainda, como somatório, (2.2),

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad (i = 1, 2, \ldots, n). \tag{2.2}$$

Resolver esse sistema significa encontrar o vetor $\mathbf{x}^T = (x_1, x_2, ..., x_n)$ que satisfaça todas as equações do sistema simultaneamente. Uma garantia para que o sistema possua solução única, é a condição de que o determinante da matriz $\mathbf{A}$ seja diferente de zero

(WATKINS, 2002). Isso também indica que a matriz **A** é inversível.

Em termos gráficos, a solução de um sistema no $\mathbb{R}^2$, por exemplo, corresponde a intersecção de duas retas. A Figura 2.1 é uma representação gráfica do sistema:

$$\begin{cases} 2x + 3y = 6 \\ x - y = 1 \end{cases}. \tag{2.3}$$

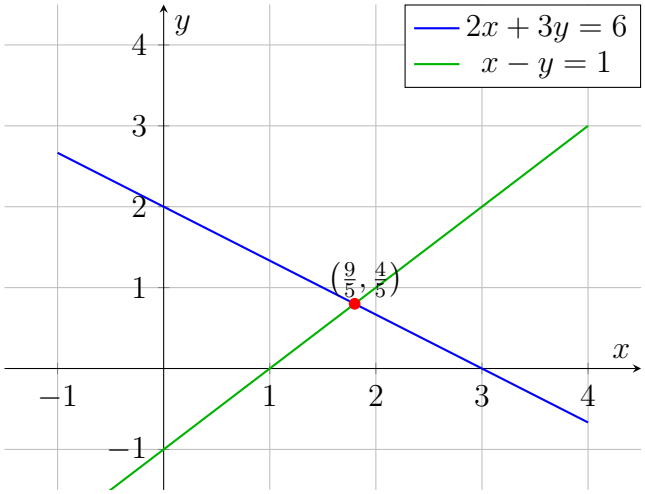


Figura 2.1: Solução de um sistema no $\mathbb{R}^2$.
Fonte: Elaboração própria.

Os métodos numéricos empregados para obter a solução de um sistema linear são classificados em dois tipos: diretos ou iterativos.

Geralmente, métodos numéricos diretos modificam a matriz de coeficientes durante o processo de resolução, o que pode ser um problema para matrizes de grande dimensão e/ou matrizes esparsas. No entanto, essas técnicas fornecem uma solução muito próxima da exata (salvos erros de arredondamento) em um número limitado de passos. São exemplos de métodos diretos: Eliminação de Gauss, Eliminação de Gauss Jordan, Fatoração **LU**, Fatoração **QR**, Fatoração de Cholesky e Método de Cramer (ARENALES e DAREZZO, 2015; CUNHA, 2000; SILVA, 2021).

Por outro lado, os métodos iterativos como Método de Newton, Gauss-Seidel, Lagrange e outros, não modificam a matriz original, entretanto não apresentam garantia de convergência e dependem de condições específicas para se ter soluções acuradas (SILVA, 2021).

No contexto dos métodos diretos, nas fatorações ou decomposições utilizam-se sistemas triangulares, que são aqueles que acima ou abaixo da diagonal principal todos os elementos são nulos. Um sistema é dito triangular inferior se sua diagonal é não nula e para todo $i < j$, seus coeficientes são nulos (2.4). Ou seja:

$$\begin{cases} l_{11}y_1 & = b_1 \\ l_{21}y_1 + l_{22}y_2 & = b_2 \\ l_{n1}y_1 + l_{n2}y_2 + \ldots + l_{nn}y_n & = b_n \end{cases} \qquad (i, j = 1, \ldots n). \tag{2.4}$$

De maneira análoga, um sistema é triangular superior se sua diagonal é não nula e para todo $j < i$, seus coeficientes são nulos (2.5). Ou seja:

$$\begin{cases} u_{11}x_1 + u_{12}x_2 + \ldots + u_{1n}x_n & = y_1 \\ u_{22}x_2 + \ldots + u_{2n}x_n & = y_2 \\ u_{nn}x_n & = y_n \end{cases} \qquad (i, j = 1, \ldots n). \tag{2.5}$$

Esses sistemas são comparativamente favoráveis de se resolver em relação a outros, por substituição regressiva ou progressiva, respectivamente. A seguir, estão os Algoritmos 1 e 2 utilizados para resolução de sistemas triangulares inferior e superior, respectivamente dados por,

$$\mathbf{L}y = b \tag{2.6}$$

e

$$\mathbf{U}x = y. \tag{2.7}$$

Algoritmo 1: Algoritmo para solução de um sistema triangular inferior

Entrada: l_{ij}, b_i

1: $y_1 = \dfrac{b_1}{l_{11}}$

2: **para** $i = 2, \ldots, n$ **faça**

3: $\quad y_i = \dfrac{(b_i - \sum_{(j=1)}^{(i-1)} l_{ij}y_j)}{l_{ii}}$

Fonte: Golub e Loan (2013); Arenales e Darezzo (2015).

Algoritmo 2: Algoritmo para solução de um sistema triangular superior

Entrada: u_{ij}, y_i

1: $x_n = \dfrac{y_n}{u_{nn}}$

2: **para** $i = (n-1), (n-2), \ldots, 1$ **faça**

3: $\quad x_i = \dfrac{(y_i - \sum_{(j=i+1)}^{(n)} u_{ij} x_j)}{u_{ii}}$

Fonte: Golub e Loan (2013); Arenales e Darezzo (2015).

Contando as operações envolvidas (seja nos Algoritmos 1 ou 2), tem-se n divisões, $\sum_{j=1}^{n-1} j = \frac{n(n-1)}{2}$ adições e $\sum_{j=1}^{n-1} j = \frac{n(n-1)}{2}$ multiplicações (usando a soma de uma progressão aritmética), resultando em n^2 operações (GOLUB e LOAN, 2013; ARENALES e DAREZZO, 2015; CUNHA, 2000). Logo, possuem baixa complexidade e consequentemente precisam de menos tempo de execução, comparativamente a outros métodos diretos, para que se obtenha a solução (ARENALES e DAREZZO, 2015).

2.2 Aplicações

Diversos trabalhos como Heath (2002), Colla (2007), Costa (2008), Pescador, Possamai e Possamai (2011), Coelho (2014), Levorato (2017), Silva (2021) e Frango (2022) ressaltam a importância dos sistemas lineares em aplicações nas Engenharias e nas Ciências Exatas, seja, por exemplo, para utilização de redes bayesianas, cálculo de potências, estruturas metálicas, balanceamento de equações químicas, modelos econômicos ou outros fins. Muitas vezes esses problemas são grandes (representados como matrizes de coeficientes esparsas) e resolvê-los manualmente se torna impraticável. Diante disso, surge a necessidade de estudar esses sistemas e as diferentes maneiras de solucioná-los computacionalmente.

Coelho (2014), por exemplo, buscou solucionar sistemas nos quais as matrizes de coeficientes são esparsas utilizando os métodos iterativos de Jacobi e Gauss-Seidel comparando diferentes estruturas de dados: *Compressed Sparse Row* (CSR), *Compressed Sparse Colunm* (CSC), *Compressed Diagonal Storage* (CDS) e *Skyline*. Já Silva (2021) implementou e comparou os métodos iterativos de Jacobi, Gauss-Seidel, Gradiente Conjugado e o GMRES utilizando a estrutura de dados *Compressed Sparse Row* (CSR). Foi feito também o uso de precondicionadores e técnicas de refinamento das soluções.

Frango (2022) analisou o método de Gradiente Conjugado usando a estrutura de dados *Compressed Sparse Row* (CSR) com uso de diferentes tipos de precondicionadores como

de Jacobi, *Damped Jacobi*, *Forward Gauss Seidel* (FGS), *Backward Gauss-Seidel* (BGS), *Symmetric Gauss-Seidel* (SGS), *Forward Successive Over Relaxation* (FSOR), *Backward Successive Over Relaxation* (BSOR) e *Symmetric Successive Over Relaxation* (SSOR).

Neste trabalho foram implementados métodos diretos, como fatoração **LU**, fatoração **QR** e fatoração de Cholesky utilizando a estrutura de listas encadeadas.

2.3 Fatoração LU

A fatoração ou decomposição **LU** consiste em decompor a matriz **A** em um produto de duas matrizes, **L** triangular inferior e **U** triangular superior.

$$\mathbf{A} = \mathbf{LU}. \tag{2.8}$$

Assim, o sistema original $\mathbf{Ax} = \mathbf{b}$, torna-se:

$$\mathbf{LUx} = \mathbf{b}. \tag{2.9}$$

Substituindo $\mathbf{Ux} = \mathbf{y}$, obtém-se o primeiro sistema triangular:

$$\mathbf{Ly} = \mathbf{b}. \tag{2.10}$$

Resolve-se o primeiro sistema, Equação (2.10), encontra-se $\mathbf{y}$ e então resolve-se o segundo sistema, Equação (2.11) onde é encontrado o valor de $\mathbf{x}$,

$$\mathbf{Ux} = \mathbf{y}. \tag{2.11}$$

Basicamente, há uma troca de um sistema linear geral (que pode ser denso ou esparso) por dois sistemas triangulares, como mostrado adiante.

Dessa forma, sejam $\mathbf{A} = (a_{ij})$, $\mathbf{L} = (l_{ij})$ e $\mathbf{U} = (u_{ij})$, com $i = 1, ..., n$ e $j = 1, ..., n$ matrizes triangulares e $\mathbf{LU} = \mathbf{A}$, então, tem-se que:

$$\begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & ... & 1 \end{pmatrix} \times \begin{pmatrix} u_{11} & u_{12} & ... & u_{1n} \\ & u_{22} & ... & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix} \tag{2.12}$$

Multiplicando a matriz **L** pela **U**, encontra-se **A**. Assim, ao multiplicar a primeira linha de **L** pelas colunas de **U**, obtém-se a primeira linha de **U** diretamente (igual a primeira linha de **A**),

$$1 \times u_{11} = a_{11}$$

$$u_{11} = a_{11},$$

$$1 \times u_{12} = a_{12}$$

$$u_{12} = a_{12},$$

$$\vdots$$

$$1 \times u_{1n} = a_{1n}$$

$$u_{1n} = a_{1n}.$$

Ao multiplicar as demais linhas de **L** pela primeira coluna de **U**, obtem-se a primeira coluna de **L**:

$$l_{21} \times u_{11} = a_{21}$$

$$l_{21} = a_{21}/u_{11}$$

$$l_{21} \times u_{12} + u_{22} = a_{22}$$

$$\vdots$$

$$l_{21} \times u_{1n} + u_{2n} = a_{2n}$$

$$l_{31} \times u_{11} = a_{31}$$

$$l_{31} = a_{31}/u_{11}$$

$$\vdots$$

$$l_{n1} \times u_{11} = a_{n1}$$

$$l_{n1} = a_{n1}/u_{11}$$

E, assim, são preenchidas as matrizes **U** e **L**: a matriz **U** por linhas e a **L** por colunas. De forma resumida, tem-se que:

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \qquad (i, j = 1, \ldots n) \tag{2.13}$$

para $i \leq j$ e

$$l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj})/u_{jj} \qquad (i, j = 1, \ldots n) \tag{2.14}$$

para $i > j$.

A seguir é apresentado o algoritmo da fatoração **LU**.

Algoritmo 3: Algoritmo da Fatoração **LU**

Entrada: a_{ij}
1: **para** $i = 1, \ldots, n-1$ **faça**
2: **para** $j = i, \ldots, n$ **faça**
3: $u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$
4: **para** $j = i+1, \ldots, n$ **faça**
5: $l_{ji} = (a_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki})/u_{ii}$
6: **para** $i = n$ **faça**
7: $u_{ii} = a_{ii} - \sum_{k=1}^{n-1} l_{ik} u_{ki}$
8: $l_{ii} = 1$

Fonte: Chapra e Canale (2015), Burden e Faires (2011), Arenales e Darezzo (2015), Datta (2010).

O Algoritmo 3 realiza $2n^3/3$ operações para fatorar a matriz **A** em **LU** (GOLUB e LOAN, 2013). Conhecidas **L** e **U**, tem-se mais $2n^2$ operações efetuadas, visto que restam dois sistemas triangulares para serem resolvidas (Algoritmos 1 e 2)(CUNHA, 2000).

2.4 Fatoração QR

A fatoraçao **QR** consiste em decompor uma matriz **A** em duas matrizes: uma matriz ortonormal (ortogonal com norma igual a 1) **Q** e uma matriz triangular superior **R**, sendo que:

$$\mathbf{A} = \mathbf{QR}. \tag{2.15}$$

Para solucionar um sistema linear do tipo $\mathbf{Ax} = \mathbf{b}$ onde $\mathbf{A} = \mathbf{QR}$, tem-se pela fatoração **QR** que:

$$\mathbf{QRx} = \mathbf{b}. \tag{2.16}$$

Como **Q** é ortogonal, a fim de simplificar o sistema, multiplica-se ambos os lados da equação por $\mathbf{Q}^T$ (a transposta de **Q**), visto que $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ (onde **I** é a matriz identidade), obtendo-se

$$\mathbf{Rx} = \mathbf{Q}^T\mathbf{b}$$

Substituindo $\mathbf{Rx} = \mathbf{y}$, calcula-se **y** a partir do sistema (2.17):

$$\mathbf{y} = \mathbf{Q}^T\mathbf{b} \tag{2.17}$$

E posteriormente, sabendo o valor de **y**, determina-se **x**, por meio da solução de:

$$\mathbf{Rx} = \mathbf{y} \tag{2.18}$$

Na fatoração **QR**, o processo de ortogonalização de Gram-Schmidt transforma o conjunto de um espaço vetorial em um conjunto ortogonal (TREFETHEN e BAU, 1997; DEMMEL, 1997; BOLDRINI *et al.*, 1980). Em outras palavras, ele transforma os vetores coluna da matriz em vetores ortogonais $\hat{q}_k$ ao subtrair as componentes nas direções dos vetores anteriores, garantindo que o produto interno entre quaisquer dois vetores coluna distintos seja nulo. Para isso, se $k = 1, ..., n$:

$$\hat{q}_k = a_k - \sum_{j=1}^{k-1} r_{jk} q_j \tag{2.19}$$

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

$$q_k = \frac{\hat{q}_k}{r_{kk}} \tag{2.20}$$

Isolando a_k da equação (2.19) e substituindo $\hat{q}_k$ encontrado em (2.20), tem-se que para $k = 1, ..., n$:

$$a_k = \sum_{j=1}^{k-1} r_{jk} q_j + r_{kk} q_k \tag{2.21}$$

Como os valores de k são conhecidos, tem-se:

$$a_1 = q_1 r_{11}$$

$$a_2 = q_1 r_{12} + q_2 r_{22}$$

$$\vdots$$

$$a_n = q_1 r_{1n} + q_2 r_{2n} + ... + q_n r_{nn} \tag{2.22}$$

ou, na forma matricial,

$$\begin{pmatrix} a_1 & a_2 & ... & a_n \end{pmatrix} = \begin{pmatrix} q_1 & q_2 & ... & q_n \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & ... & r_{1n} \\ 0 & r_{22} & ... & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & r_{nn} \end{pmatrix} \tag{2.23}$$

A seguir, é apresentado o algoritmo da fatoração **QR**.

Algoritmo 4: Algoritmo de Fatoração **QR** (Gram-Schmidt Clássico)

Entrada: a_{ij}

$R_{1,1} = \sqrt{\sum_{i=1}^{m} A_{i,1}^2}$

para $i = 1, \ldots, m$ **faça**

  $Q_{i,1} = \frac{A_{i,1}}{R_{1,1}}$

para $k = 2, \ldots, n$ **faça**

  para $j = 1, \ldots, k-1$ **faça**

    $R_{j,k} = \sum_{i=1}^{m} Q_{i,j} \cdot A_{i,k}$

  para $i = 1, \ldots, m$ **faça**

    $z_i = A_{i,k} - \sum_{j=1}^{k-1} Q_{i,j} \cdot R_{j,k}$

  $R_{k,k} = \sqrt{\sum_{i=1}^{m} z_i^2}$

  para $i = 1, \ldots, m$ **faça**

    $Q_{i,k} = \frac{z_i}{R_{k,k}}$

Fonte: Golub e Loan (2013).

A fatoração **QR** mostrada no Algoritmo 4 requer aproximadamente n^3 operações para ser efetuada.

2.5 Fatoração de Cholesky

Dado um sistema $\mathbf{Ax} = \mathbf{b}$, onde a matriz $\mathbf{A}$ é simétrica ($\mathbf{A} = \mathbf{A}^T$) e positiva definida ($\mathbf{x}^T\mathbf{Ax} > 0$ para todo $\mathbf{x} \neq 0$), diz-se que ela pode ser decomposta pelo produto de duas matrizes triagulares:

$$\mathbf{A} = \mathbf{L}^T\mathbf{L} \tag{2.24}$$

e, assim, o sistema inicial $\mathbf{Ax} = \mathbf{b}$ é reescrito como:

$$\mathbf{L}^T\mathbf{Lx} = \mathbf{b}. \tag{2.25}$$

Define-se $\mathbf{Lx} = \mathbf{y}$, e resolve-se primeiro sistema triangular:

$$\mathbf{L}^T\mathbf{y} = \mathbf{b} \tag{2.26}$$

Posteriormente, com o valor de $\mathbf{y}$ encontra-se $\mathbf{x}$ resolvendo um segundo sistema triangular:

$$\mathbf{Lx} = \mathbf{y}. \tag{2.27}$$

Análogo a fatoração $\mathbf{LU}$, a seguir é apresentado o produto $\mathbf{L}^T\mathbf{L} = \mathbf{A}$ na forma matricial de acordo com Arenales e Darezzo (2015):

$$\begin{pmatrix} l_{11} & & & & \\ l_{12} & l_{22} & & & \\ l_{13} & l_{23} & l_{33} & & \\ \vdots & \vdots & \ddots & & \\ l_{n1} & l_{2n} & l_{3n} \ldots & r_{nn} \end{pmatrix} \times \begin{pmatrix} l_{11} & l_{12} & l_{13} & \ldots & l_{1n} \\ & l_{22} & l_{23} & \ldots & l_{2n} \\ & & l_{33} & \ldots & l_{3n} \\ & & & \ddots & \vdots \\ & & & & l_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_{21} & a_{22} & a_{13} & ... & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} \\ \vdots & \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & ... & a_{nn} \end{pmatrix} \tag{2.28}$$

Ao multiplicar as linhas da primeira matriz pelas colunas da segunda (transposta da primeira) obtém-se os elementos de $\mathbf{L}$, como mostrado a seguir:

$$l_{11}^2 = a_{11}$$

$$l_{11}l_{12} = a_{12}$$

$$l_{11}l_{13} = a_{13}$$

Separando os elementos referentes a diagonal, tem-se

$$l_{11}^2 = a_{11}$$

$$l_{11} = \sqrt{a_{11}}, \tag{2.29}$$

$$r_{12}^2 + r_{22}^2 = a_{22}$$

$$l_{22} = (a_{22} - l_{12}^2)^{\frac{1}{2}}, \tag{2.30}$$

$$l_{13}^2 + l_{23}^2 + l_{33}^2 = a_{33}$$

$$l_{33} = (a_{33} - l_{13}^2 + l_{23}^2)^{\frac{1}{2}}. \tag{2.31}$$

Pode-se generalizar para as diagonais que para $i = 1, \ldots, n$:

$$l_{ii} = (a_{ii} - \sum_{k=1}^{i=1} r_{ki}^2)^{\frac{1}{2}} \tag{2.32}$$

Para os elementos fora da diagonal, observa-se a construção das linhas de $\mathbf{L}$:

$$l_{11}l_{12} = a_{12}$$

$$l_{12} = \frac{a_{12}}{l_{11}} \tag{2.33}$$

$$l_{11}l_{13} = a_{13}$$

$$l_{13} = \frac{a_{13}}{l_{11}} \tag{2.34}$$

$$\vdots$$

$$l_{12}l_{13} + l_{22}l_{23} = a_{23}$$

$$l_{23} = \frac{a_{23} - l_{12}l_{13}}{l_{22}} \tag{2.35}$$

Generalizando, fora das diagonais, para $i = 1, \ldots, n$ e $j = i + 1, \ldots, n$:

$$l_{ij} = \frac{(a_{ij} - \sum_{k=1}^{i-1} l_{ki}l_{kj})}{l_{ii}} \tag{2.36}$$

A seguir, tem-se o Algoritmo da fatoração de Cholesky com os passos mostrados aqui.

Algoritmo 5: Algoritmo da Fatoração de Cholesky.

Entrada: a_{ij} (matriz simétrica e positiva definida)

1: **para** $i = 1 : n$ **faça**
2: $\quad l_{ii} = (a_{ii} - \sum l_{ki}^2)^{\frac{1}{2}}$
3: $\quad$ **para** $j = i + 1 : n$ **faça**
4: $\quad\quad l_{ij} = (a_{ij} - \sum_{k=1}^{i-1} l_{ki} l_{kj})/l_{ii}$

Fonte: Arenales e Darezzo (2015), Davis (2006), Saad (2003), Ruggiero e Lopes (2000) .

A fatoração de Cholesky mostrada no Algoritmo 5 requer $n^3/3$ operações para ser efetuada (GOLUB e LOAN, 2013). Além do método numérico, outra escolha importante na solução de um sistema é a estrutura de dados utilizada, que será discutida no capítulo a seguir.

Capítulo 3

Esparsidade e estrutura de dados

Este capítulo apresenta os conceitos de densidade, esparsidade, estruturas de dados indexada e de listas encadeadas, que foram objeto de estudo e implementação neste trabalho, além de alguns exemplos.

3.1 Densidade e esparsidade de uma matriz

A densidade de uma matriz retrata a porcentagem de elementos não nulos presentes nela, enquanto a esparsidade refere-se à porcentagem de elementos nulos dela. Dessa forma, a esparsidade é o complemento da densidade. Dada uma matriz quadrada de ordem n e seja nz o número de elementos não nulos da mesma, tem-se que sua densidade (D) e esparsidade (E) são dadas, respectivamente, por:

$$D = \frac{nz}{n \times n} \times 100, \tag{3.1}$$

$$E = (100 - D). \tag{3.2}$$

Uma matriz quadrada de ordem n, possui n^2 elementos. Considera-se agora, uma matriz quadrada de ordem $n = 8$, contendo 32 elementos não nulos (CUNHA, 2000):

$$\mathbf{M} = \begin{pmatrix} -2 & 1 & & 1 & & & & \\ 1 & -4 & 1 & & 1 & & & \\ & 1 & -4 & 1 & & 1 & & \\ 1 & & 1 & -4 & 1 & & 1 & \\ & 1 & & 1 & -4 & 1 & & 1 \\ & & 1 & & 1 & -4 & 1 & \\ & & & 1 & & 1 & -4 & 1 \\ & & & & 1 & & 1 & -2 \end{pmatrix} \tag{3.3}$$

A matriz $\mathbf{M}$ apresenta densidade e esparsidade iguais (50%).

No exemplo a seguir tem-se a matriz *"cage3"* retirada de repositório *online* (`https://sparse.tamu.edu/vanHeukelum/cage3`) (este repositório será descrito no capítulo 4). Ela possui ordem 5 e 19 elementos não nulos:

$$\begin{pmatrix} .666666666666667 & .366555998208319 & .300110668458348 & .366555998208319 & .300110668458348 \\ .100036889486116 & .533407112305565 & & .200073778972232 & \\ .122185332736106 & & .577703998805546 & & .244370665472212 \\ .050018444743058 & .100036889486116 & & .283314888590275 & .183277999104159 \\ .0610926663680531 & & .122185332736106 & .150055334229174 & .27224066696528 \end{pmatrix} \tag{3.4}$$

Para uma matriz de quinta ordem o número total de elementos é 25, portanto, no caso da matriz *"cage3"*, a densidade é de 76% e esparsidade 24%.

Matrizes com grande número de elementos não nulos, são chamadas de matrizes densas. No entanto, em alguns casos o número de elementos nulos pode ser bem maior que o número de não nulos. Essas matrizes recebem o nome de matrizes esparsas e, geralmente, são oriundas de aplicações práticas.

A seguir são apresentados na Tabela 3.1 os valores não nulos do problema *"bcsstm02"*. Ele foi retirado de repositório *online* (`https://sparse.tamu.edu/HB/bcsstm02`). Nesses problemas, os valores estão dispostos em colunas. A primeira coluna representa as linhas, a segunda coluna representa as colunas da matriz e na terceira coluna está o valor do respectivo elemento.

Tabela 3.1: Problema *"bcsstm02"*.

1	1	.09213858051
2	2	.09213858051
3	3	.09213858051
4	4	.137995737983
5	5	.137995737983
6	6	.137995737983
7	7	.137995737983
8	8	.137995737983
9	9	.137995737983
10	10	.09213858051
11	11	.09213858051
12	12	.09213858051
13	13	.172828573455
14	14	.172828573455
15	15	.172828573455
16	16	.0852383576022
17	17	.0852383576022
18	18	.0852383576022
19	19	.0852383576022
20	20	.0852383576022
21	21	.0852383576022
22	22	.172828573455
23	23	.172828573455
24	24	.172828573455
25	25	.0617332189107
26	26	.0617332189107
27	27	.0617332189107
28	28	.141308341476
29	29	.141308341476
30	30	.141308341476
31	31	.141308341476
32	32	.141308341476
33	33	.141308341476
34	34	.0617332189107
35	35	.0617332189107
36	36	.0617332189107
37	37	.125426638452
38	38	.125426638452
39	39	.125426638452
40	40	.0533208927371
41	41	.0533208927371
42	42	.0533208927371
43	43	.0533208927371
44	44	.0533208927371
45	45	.0533208927371
46	46	.125426638452
47	47	.125426638452
48	48	.125426638452
49	49	.0231706100487
50	50	.0231706100487
51	51	.0231706100487
52	52	.0305931884075
53	53	.0305931884075
54	54	.0305931884075
55	55	.0648568699369
56	56	.0648568699369
57	57	.0648568699369
58	58	.0648568699369
59	59	.0648568699369
60	60	.0648568699369
61	61	.0305931884075
62	62	.0305931884075
63	63	.0305931884075
64	64	.019746938665
65	65	.019746938665
66	66	.019746938665

Fonte: Elaboração própria.

O problema *"bcsstm02"* é um problema de ordem 66. Sua matriz, portanto, possui um total de 4356 elementos. Contudo, trata-se de uma matriz diagonal. Ou seja, uma matriz que possui todos os elementos nulos a exceção dos elementos pertencentes a diagonal. Esse problema apresenta densidade aproximada de 1,51% e esparsidade 98,48%. A seguir, na Figura 3.1, encontra-se uma imagem do problema *"bcsstm02"*. Ela revela como estão dispostos os elementos não nulos na matriz.

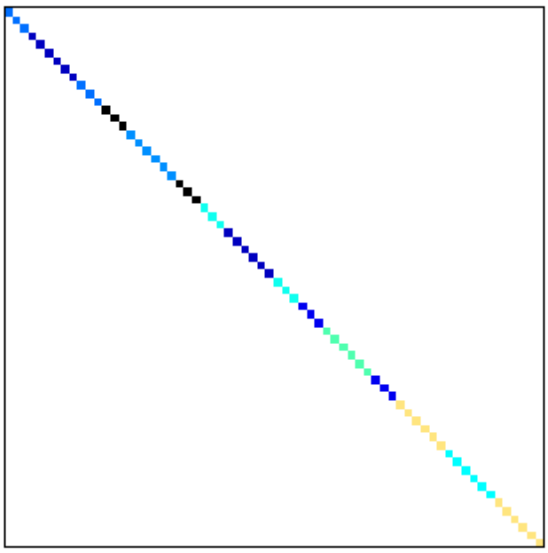


Figura 3.1: Problema *"bcsstm02"*.
Fonte: Davis e Hu (2011).

Neste trabalho buscou-se resolver sistemas nos quais as matrizes de coeficientes são quadradas e majoritariamente esparsas.

3.2 Estrutura indexada e listas encadeadas

Para resolver os sistemas lineares, foram utilizadas neste trabalho dois tipos de estruturas de dados: indexada e de listas encadeadas.

Na estrutura indexada os coeficientes das equações estão dispostos a partir de um índice e são armazenados no formato de matrizes. Todos os elementos são armazenados em suas respectivas posições da matriz de acordo com a forma algébrica.

A declaração, em linguagem C, de uma matriz genérica de tamanho fixo, usando estrutura de dados indexada, é simplesmente "*double* A[i][j]", onde *double* indica o tipo

da variável, ponto flutuante de precisão dupla (*double precision floating point*), usada para armazenar números reais. Esse tipo de estrutura apresenta a vantagem de possuir um acesso direto aos elementos armazenados, visto que cada elemento pode ser obtido a partir de seu índice $a[i][j]$. A Figura 3.2 ilustra essa estrutura e o modo como os elementos são organizados na memória.

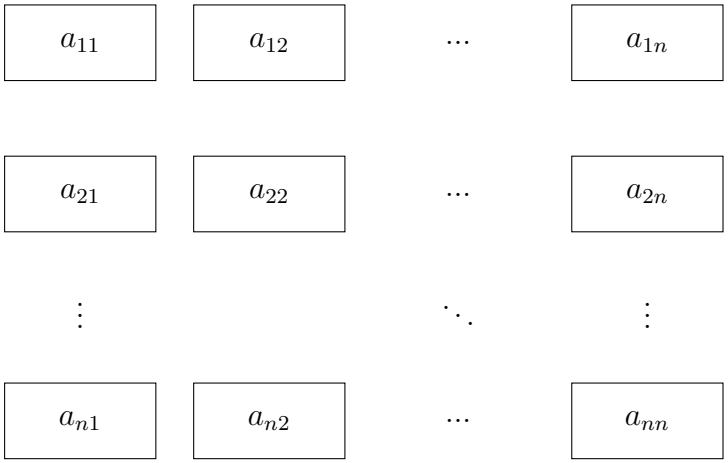


Figura 3.2: Representação da estrutura indexada.
Fonte: Elaboração própria.

De fato, ao armazenar uma matriz esparsa de forma indexada reserva-se espaço na memória para todos os seus elementos, inclusive os nulos (WEISS, 2013; DROZDEK, 2013). Isso leva a um expressivo desperdício de memória, além de operações aritméticas básicas com elementos nulos.

Outra estrutura utilizada nesse trabalho é a de listas encadeadas. Nessa estrutura, tem-se uma sequência de elementos ordenados chamados nós, como mostrado na Figura 3.3.

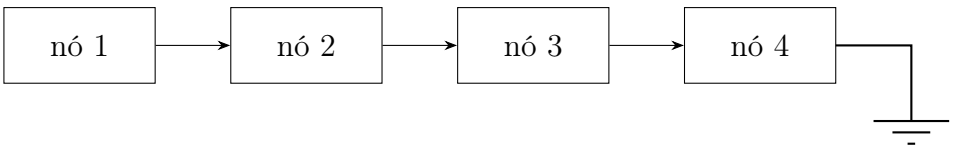


Figura 3.3: Representação de uma lista encadeada.
Fonte: Elaboração própria.

Cada nó carrega no mínimo dois componentes: um dado (ou um valor) e um ponteiro, ver Figura 3.4.

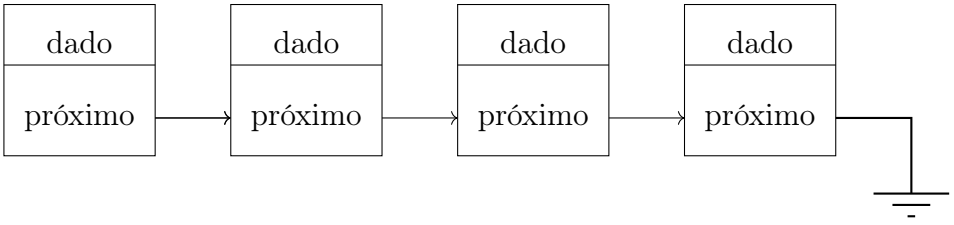


Figura 3.4: Representação dos nós na lista encadeada.
Fonte: Elaboração própria.

O primeiro nó da lista é chamado nó cabeça e o último nó da lista possui ponteiro *NULL*.

A seguir, encontra-se o Código 1, um exemplo genérico de nó (TENENBAUM, LANGSAM, AUGENSTEIN 1995; WEISS, 2013; ROVAI, 2018):

Código 1: Estrutura básica do nó da lista

```
1:  struct no {
2:  int dado;
3:  struct no *proximo;
4: };
```

Fonte: Elaboração própria.

Nesse exemplo, cada nó contém um dado (que nesse caso é um valor inteiro) e um ponteiro que aponta para o próximo nó da lista.

A estrutura de listas utilizada nesse trabalho é a combinação de duas listas apresentada adiante no Código 2:

- Uma lista de linhas ("linhaA" e "linhasA"), formada a partir dos primeiros elementos de cada linha da matriz. Essa lista possui n nós (já que sendo uma matriz quadrada o número de linhas é igual ao de colunas) e é armazenada em um vetor que aqui é chamado de "vetlinhaA". Cada nó de "linhaA" é do tipo "elemento A" e guarda o índice da linha "lA" e um ponteiro para a próxima linha.
- n listas com os elementos não nulos de cada linha da matriz ("elementoA"). Cada elemento contém os índices de linhas e colunas ("iA" e "jA"), além do valor do elemento ("valorA") e um ponteiro para o próximo elemento da lista ("proximoA").

Código 2: Estrutura de listas utilizada

```
struct elementoA {
int iA, jA;
double valorA;
struct elementoA* proximoA;
};
typedef struct elementoA elementoA;
struct linhaA {
elementoA* proxlinhaA;
```

```
int lA;
};
typedef struct linhaA linhaA;
struct linhasA {
linhaA *vetlinhaA;
};
typedef struct linhasA linhasA;
linhaA **vetlinhaA;
```

Fonte: Elaboração própria.

Essa estrutura foi utilizada para armazenar os valores não nulos de todas as matrizes principais: a matriz "problema", aqui chamada de **A**; e as matrizes resultantes de cada fatoração. Na fatoração **LU**, por exemplo, as matrizes **L** e **U** também foram armazenadas dessa forma; assim como as matrizes da fatoração **QR** e de Cholesky.

A seguir a Figura 3.5 apresenta a estrutura de listas encadeadas empregada.

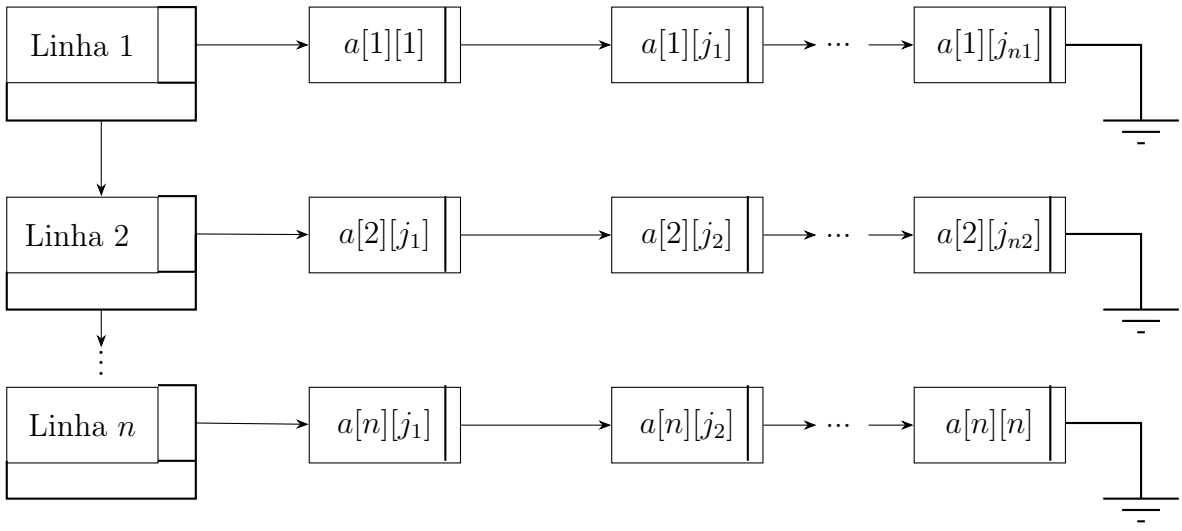


Figura 3.5: Representação da matriz **A** com a estrutura de lista encadeada.
Fonte: Elaboração própria.

Na Figura 3.5, o nó "Linha 1" (assim como Linha 2... Linha n) guarda o índice da linha, o ponteiro para próxima linha (nesse caso Linha 2) e faz parte de "vetlinhaA".

O nó "Linha 1" aponta para "Linha 2" e para o primeiro elemento $a[1][1]$ (ele existe, pois todos os problemas de solução única apresentam pelo menos os elementos da diagonal principal). Ainda na primeira linha, tem-se os outros elementos não nulos presentes na primeira linha da matriz **A**.

Como $a[1][2]$ e outros elementos da matriz podem ser nulos, eles não necessariamente entram na lista. Por isso não foi especificado a coluna do segundo elemento da primeira lista, referente a primeira linha da matriz, tampouco as seguintes.

Na Figura 3.5, não foi colocado na imagem os valores de j, porque sendo A uma matriz esparsa, eles podem existir ou não. O mesmo é válido para "Linha 2" e as linhas seguintes. Na última linha, tem-se a representação da última linha da matriz **A**. A "Linha n", com seus elementos não nulos na sequência, sendo o último o elemento $a[n][n]$ (presente na

diagonal principal). Cada nó apresentado como $a[i][j_n]$, carrega seu índice de linha, índice de coluna, o valor do elemento e um ponteiro que aponta para o próximo nó da lista.

O problema *"bcsstm02"* é uma matriz diagonal de ordem 66. Nessa estrutura, por exemplo, é armazenado como mostrado na Figura 3.6.

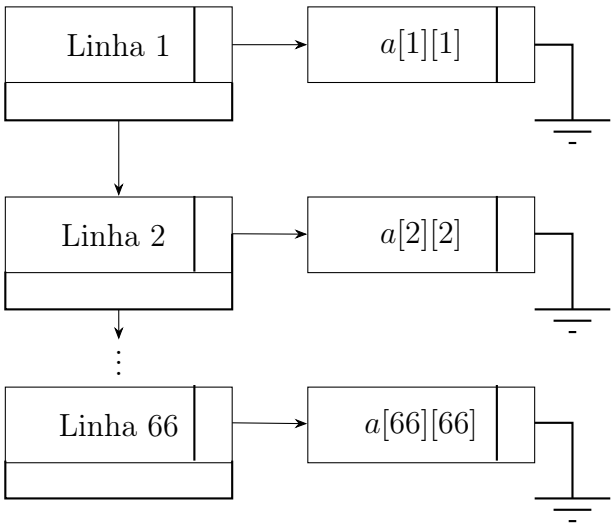


Figura 3.6: Representação do problema *"bcsstm02"* com a estrutura de lista encadeada empregada.

Fonte: Elaboração própria.

Como nesse caso trata-se de uma matriz diagonal, as listas das linhas possuem apenas um elemento, o elemento da diagonal. Logo, há menor volume de dados armazenados na memória, e dessa forma, é possível resolver sistemas maiores.

Nesta pesquisa as estruturas indexada e de listas encadeadas foram utilizadas a fins comparativos, tanto quanto ao tempo de resolução dos problemas como resíduo gerado.

3.2.1 Manipulação das estruturas

Como já foi dito, a estrutura indexada permite um acesso direto aos elementos por meio de seus índices da matriz. Na estrutura de listas, isso não ocorre. Para encontrar um elemento $a[i][j]$ específico é necessário percorrer a lista que representa a linha da matriz, buscado-o até encontrá-lo e então efetuar a operação desejada.

Para realizar esta tarefa é utilizado um ponteiro "atualA", que percorre "vetlinhaA" e os nós da lista de elementos até encontrar o elemento desejado.

Por exemplo, para encontrar o elemento $a[3][2]$, tem-se que realizar as operações apresentadas no Algoritmo 6.

Algoritmo 6: Busca do elemento $a[3][2]$ na estrutura de lista encadeada.

```
1: atualA ← vetlinhaA[3] → proxlinhaA;
2: enquanto (atualA ≠ NULL) faça
3: se atualA->jA = 2 então
4:     atualA ← valorA;
5: atualA ← atualA->proximoA;
6: retorna 0;
```

Fonte: Ziviani (1999), King (2008), Cormen *et al.* (2012).

O vetor "vetlinhaA", posiciona o ponteiro "atualA" na linha correta e o ponteiro "proximoA" percorre a linha até encontrá-lo, caso exista, e assim é feito o acesso a um elemento específico dessa lista.

Para realizar operações, buscam-se os elementos requeridos e caso encontrados (se existir), efetua-se a operação de soma, subtração, multiplicação ou divisão. Quando o elemento não é encontrado na lista durante a busca, tem-se que seu valor é nulo. Se a operação entre os valores buscados é de soma ou subtração e um dos valores não foi encontrado, trata-se de uma soma ou subtração por zero. Nesses casos, não é efetuada operação alguma.

Se a operação entre os valores buscados for uma multiplicação (e um dos valores não foi encontrado), ou a divisão de um valor não encontrado (portanto, igual a zero) por um valor qualquer (diferente de zero); o resultado também é sempre nulo. Não efetuar essas operações resulta em uma economia de tempo computacional para um problema com grande número de elementos.

Se durante uma fatoração (**LU**, **QR** ou de Cholesky), for obtido um valor nulo para um coeficiente de qualquer das matrizes **L**, **U**, **Q**, **R** ou **L** (Cholesky), esse coeficiente é "descartado". Como as matrizes fatoradas (**L** e **U**, **Q** e **R** e **L**) também seguem a mesma estrutura de listas encadeadas utilizada para a matriz **A**, elas não armazenam elementos nulos.

Capítulo 4

Problemas tratados

Este capítulo apresenta os problemas escolhidos para serem resolvidos nesta dissertação, além de algumas de suas características que são relevantes para essa pesquisa.

4.1 Gerador aleatório

Alguns dos problemas utilizados para testes são sistemas lineares obtidos a partir de um gerador aleatório. Esses sistemas foram disponibizados por Silva (2021) e construídos a partir de uma sequência de passos apresentados por Junior (2011). A seguir são apresentados esses passos:

1. Criação de um vetor aleatório $\mathbf{v} \in \mathbb{R}^n$ uniformemente distribuído entre 0 e 1;

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{4.1}$$

2. Cálculo da matriz $\mathbf{W}$, sendo $\mathbf{W} = \mathbf{I} - (2\mathbf{v}\mathbf{v}^T)/(\mathbf{v}^T\mathbf{v})$ e considerando $c = \mathbf{v}^T\mathbf{v}$;

$$\mathbf{W} = \begin{bmatrix} 1 - \dfrac{2v_1^2}{c} & -\dfrac{2v_1v_2}{c} & \cdots & -\dfrac{2v_1v_n}{c} \\ -\dfrac{2v_2v_1}{c} & 1 - \dfrac{2v_2^2}{c} & \cdots & -\dfrac{2v_2v_n}{c} \\ \vdots & \vdots & \ddots & \vdots \\ -\dfrac{2v_nv_1}{c} & -\dfrac{2v_nv_2}{c} & \cdots & 1 - \dfrac{2v_n^2}{c} \end{bmatrix} \tag{4.2}$$

3. Cálculo da matriz diagonal $\mathbf{D}$, cujos elementos são os autovalores de $\mathbf{W}$ obtido

anteriormente;

$$\mathbf{D} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \tag{4.3}$$

4. Cálculo da matriz $\mathbf{A}$, onde $\mathbf{A} = \mathbf{WDW}^T$.

$$a_{ij} = \sum_{k=1}^{n} w_{ik} d_k w_{jk} \qquad i = 1, 2, ..., n, \quad j = 1, 2, ..., n \tag{4.4}$$

5. Gera-se o vetor solução $\mathbf{x}$, e a partir do o produto de $\mathbf{A}$ por $\mathbf{x}$, encontra-se o vetor $\mathbf{b}$.

Seguindo as etapas descritas acima tem-se uma matriz quadrada cheia, densa. Para obter variações significativas na densidade da matriz gerada foram utilizadas as funções "*trial*" e "*triau*" do *software Octave* (2025). Dessa forma, tornando as matrizes triangulares ou tridiagonais, se utilizadas ambas funções de forma associada.

Para os testes, foram escolhidas matrizes com ordem de grandeza de 100 a 8000, cuja distribuição de elementos é completa, triangular ou tridiagonal como é apresentado na Tabela 4.1.

Tabela 4.1: Problemas do gerador aleatório.

Problema	n	nz	D (%)	E (%)
tridiag100	100	298	2, 98	97, 02
triang100	100	5050	50, 5	49, 5
cheia100	100	10000	100	0
tridiag500	500	1498	0,6	99,4
triang500	500	125250	50, 1	49, 9
cheia500	500	250000	100	0
tridiag1000	1000	2998	0,3	99, 7
triang1000	1000	500500	50, 05	49, 95
cheia1000	1000	1000000	100	0
tridiag1500	1500	4498	0,2	99, 8
triang1500	1500	1125750	50, 03	49, 97
cheia1500	1500	2250000	100	0
tridiag2000	2000	5998	0,15	99, 85
triang2000	2000	2001000	50, 02	49, 97
cheia2000	2000	4000000	100	0
tridiag3500	3500	10498	0,08	99, 91
tridiag5000	5000	14998	0,06	99, 94
tridiag6500	6500	19498	0,05	99, 95
tridiag8000	8000	23998	0, 04	99, 96
Média	2015,79	597464,84	39,74	57,63

Fonte: Silva (2021).

Os problemas da Tabela 4.1 estão nomeados conforme sua característica tridiagonal (tridiag.), triangular (triang.) ou cheia, e sua respectiva dimensão ou ordem de grandeza. Nota-se que a densidade das matrizes variou de 0,04% a 100%.

Nas matrizes tridiagonais, tem-se que os elementos não nulos são os elementos da diagonal principal (n elementos) e os elementos da diagonal acima e abaixo da diagonal principal ($n-1$ elementos em cada). Assim, o número de total de elementos não nulos é: $n + (n-1) + (n-1)$. Logo, a densidade dessas matrizes será: $\dfrac{3n-2}{n^2}$; portanto quanto maior n, menor será sua densidade.

Para as matrizes triangulares, inferiores ou superiores o raciocínio é similar. O número de elementos não nulos (abaixo ou acima da diagonal e na diagonal) é um somatório de $1+2+3+...+n = \dfrac{n(n+1)}{2}$, como o número total de elementos é n^2, a densidade dessas matrizes equivale a: $\dfrac{n(n+1)/2}{n^2} = \dfrac{n+1}{2n}$. Dessa forma, quanto maior o valor de n, mais sua densidade se aproxima de 50%.

Nas matrizes cheias não há elementos nulos, por isso são matrizes com densidade igual a 100% e esparsidade nula. Esses problemas (assim como aqueles representados por matrizes triangulares) por serem mais densos tendem a requerer expressivo espaço na memória e não são o foco desse trabalho. O uso de listas encadeadas tende a ser uma alternativa interessante para matrizes esparsas, como será mostrado adiante.

4.2 Repositório *SuiteSparse Matrix Collection*

Outros problemas escolhidos para os testes foram retirados do repositório *online*: *SuiteSparse Matrix Collection* ou *University of Florida Sparse Matrix Collection* de Davis e Hu (2011). Um banco de dados repleto de matrizes reais e esparsas coletadas a partir de aplicações em Engenharias, Matemática, Estatística, Economia e outras pode ser encontrado em `https://sparse.tamu.edu/`.

De modo geral, foram escolhidos problemas com diferentes valores para ordem de grandeza e esparsidade. Esses problemas foram agrupados segundo suas dimensões, nos intervalos: $19 \leq n \leq 1000$, $1080 \leq n \leq 6537$ e $6611 \leq n \leq 21132$. No entanto, vale lembrar que existem sistemas de dimensões bem maiores, como da ordem de milhões de equações.

Alguns dos problemas escolhidos também foram testados por Silva (2021) usando outros métodos numéricos (iterativos) e outra estrutura de dados.

Todos os problemas escolhidos são de matrizes quadradas, com determinantes não nulos, ou seja, problemas de solução única. Em todos os casos, foram escolhidas matrizes esparsas, cuja esparsidade encontra-se entre 77,0% e 99,9%.

Algumas características como o fato da matriz ser simétrica e positiva definida são condições para o uso da fatoração de Cholesky. No entanto, essas características não foram utilizadas como critérios de escolha de uma matriz problema. Ou seja, é esperado que nem todos os problemas sejam resolvidos com a fatoração de Cholesky.

A seguir, as Tabelas 4.2, 4.3 e 4.4 apresentam os problemas em sequência de acordo com a ordem de grandeza, quantidade de elementos não nulos, densidade, esparsidade, número de condição e especificação se a matriz é simétrica e positiva definida. A Tabela 4.2 contém problemas com ordem de grandeza entre 19 e 1000.

Tabela 4.2: Problemas do repositório com $19 \leq n \leq 1000$.

Problema	n	nz	D (%)	E (%)	Núm. condição	Simétrica	Positiva definida
Trefethen_20b	19	83	22,99	77,01	$3,03 \times 10^1$	Sim	Sim
Trefethen_20	20	89	22,25	77,75	$6,3 \times 10^1$	Sim	Sim
bcsstk01	48	224	9,72	90,28	$8,82 \times 10^5$	Sim	Sim
bfw62a	62	450	11,71	88,29	$5,53 \times 10^2$	Não	Não
bcsstm02	66	66	1,51	98,48	8,75	Sim	Sim
Trefethen_150	150	1095	4,87	95,13	$7,69 \times 10^2$	Sim	Sim
Trefethen_200	200	1545	3,86	96,14	$1,09 \times 10^3$	Sim	Sim
Trefethen_300	300	2489	2,76	97,23	$1,77 \times 10^3$	Sim	Sim
bcsstm06	420	420	0,24	99,76	$3,46 \times 10^6$	Sim	Sim
bcsstm07	420	3836	2,17	97,82	$7,61 \times 10^3$	Sim	Sim
Trefethen_500	500	4489	1.79	98,20	$3,18 \times 10^3$	Sim	Sim
Trefethen_700	700	6677	1,36	98,64	$4,71 \times 10^3$	Sim	Sim
sherman1	1000	3750	0,37	99,62	$1,56 \times 10^4$	Sim	Não
Média	300,38	1939,46	6,58	93,41	$3,37 \times 10^5$	-	-

Fonte: Davis e Hu (2011).

Nota-se que dois dos problemas apresentados na Tabela 4.2 não podem ser solucionados pelo método de decomposição de Cholesky, pois sua matriz $\mathbf{A}$ não é simultaneamente simétrica e positiva definida.

A Tabela 4.3 apresenta problemas com ordem de grandeza entre 1080 e 6537.

Tabela 4.3: Problemas do repositório com $1080 \leq n \leq 6537$.

Problema	n	nz	D (%)	E (%)	Núm. condição	Simétrica	Positiva definida
sherman2	1080	23094	1,97	98,02	$9,64 \times 10^{11}$	Não	Não
bcsstm09	1083	1083	0,09	99,91	1×10^{4}	Sim	Sim
bcsstm10	1086	22092	0,98	99,02	1×10^{4}	Sim	Sim
sherman4	1104	3786	0,31	99,69	$2,18 \times 10^{3}$	Não	Não
1138_bus	1138	2596	0,20	99,80	$8,57 \times 10^{6}$	Sim	Sim
rdb1250	1250	7300	0,47	99,53	$1,75 \times 10^{3}$	Não	Não
c-19	2327	12072	0,22	99,78	$6,78 \times 10^{5}$	Sim	Não
cavity10	2597	76367	1,13	98,87	$2,95 \times 10^{6}$	Não	Não
sherman5	3312	20793	0,19	99,81	$1,88 \times 10^{5}$	Não	Não
bcsstm21	3600	3600	0,03	99,97	$2,37 \times 10^{1}$	Sim	Sim
c-26	4307	19422	0,10	99,90	$4,60 \times 10^{8}$	Sim	Não
c-28	4598	17594	0,08	99,92	$5,24 \times 10^{9}$	Sim	Não
sherman3	5005	20033	0,08	99,92	$5,01 \times 10^{17}$	Não	Não
c-29	5033	24382	0,10	99,90	$3,94 \times 10^{10}$	Sim	Não
c-30	5321	35507	0,12	99,87	$2,09 \times 10^{13}$	Sim	Não
c-31	5339	41955	0,15	99,85	$1,13 \times 10^{9}$	Sim	Não
c-32	5975	30223	0,08	99,91	$4,90 \times 10^{7}$	Sim	Não
c-33	6317	31220	0,08	99,92	$1,12 \times 10^{8}$	Sim	Não
c-35	6537	34714	0,08	99,92	$3,33 \times 10^{6}$	Sim	Não
Média	3526,79	21964,74	0,34	99,66	$2,64 \times 10^{16}$	-	-

Fonte: Davis e Hu (2011).

Novamente, dentre os problemas escolhidos, há alguns problemas que não podem ser resolvidos pelo método de Cholesky.

Os problemas de ordem entre 6611 e 21132 estão na Tabela 4.4 e foram os maiores problemas do repositório escolhidos para este trabalho. Na Tabela 4.4, apenas duas matrizes possuem simetria e são positivas definidas. Com isso, podem solucionados pela fatoração de Cholesky.

Tabela 4.4: Problemas do repositório com $6611 \leq n \leq 21132$.

Problema	n	nz	D (%)	E (%)	Núm. condição	Simétrica	Positiva definida
c-34	6611	35472	0,08	99,92	$1,12 \times 10^8$	Sim	Não
c-36	7479	36710	0,06	99,93	$7,64 \times 10^7$	Sim	Não
c-38	8127	42908	0,06	99,93	$1,53 \times 10^4$	Sim	Não
c-37	8204	41440	0,06	99,94	$6,32 \times 10^7$	Sim	Não
c-39	9271	73041	0,08	99,91	$3,47 \times 10^8$	Sim	Não
c-41	9769	55757	0,06	99,94	$4,78 \times 10^{12}$	Sim	Não
c-40	9941	45721	0,05	99,95	$1,11 \times 10^6$	Sim	Não
c-42	10471	60378	0,05	99,94	$1,75 \times 10^7$	Sim	Não
c-44	10728	47864	0,04	99,96	$6,08 \times 10^7$	Sim	Não
c-43	11125	67400	0,05	99,94	$9,16 \times 10^8$	Sim	Não
c-45	13206	93829	0,05	99,95	$5,45 \times 10^{10}$	Sim	Não
pres_poison	14822	356313	0,17	99,83	$2,04 \times 10^6$	Sim	Sim
c-46	14913	72655	0,03	99,97	$4,51 \times 10^4$	Sim	Não
c-47	15343	113372	0,05	99,95	$3,16 \times 10^8$	Sim	Não
bcsstm25	15439	15439	0,01	99,99	$6,06 \times 10^9$	Sim	Sim
c-48	18354	92217	0,03	99,97	$1,19 \times 10^9$	Sim	Não
c-49	21132	89087	0,02	99,98	$6,02 \times 10^8$	Sim	Não
Média	12055	79329,59	0,06	99,94	$2,85 \times 10^{11}$	-	

Fonte: Davis e Hu (2011).

Nota-se pelos dados das Tabelas 4.2, 4.3 e 4.4 que os problemas escolhidos variam sua ordem de grandeza de dezenas a dezenas de milhares de elementos, sendo o maior problema, com ordem de 21132. Ainda nas Tabelas 4.2, 4.3 e 4.4, é apresentado o número de condição de cada uma das matrizes, disponível no repositório *SuiteSparse Matrix Collection*. Esses números são bem maiores que 1, o que indica que esses problemas tendem a apresentar erros expressivos, sendo por isso de difícil solução.

Dentre as matrizes apresentadas, nota-se que nem todas são positiva definida e simétricas. O que separa os problemas passíveis de resolução pelo método Cholesky. Algumas das características comentadas podem ser observadas nas imagens das matrizes, como será mostrado para alguns casos a seguir. As imagens foram retiradas do próprio repositório de problemas *SuiteSparse Matrix Collection* de Davis e Hu (2011), e apresentadas por ordem de grandeza.

As matrizes cujas estruturas formadas pelos elementos não nulos formam imagens muito semelhantes, estão nos apêndices deste trabalho. Nas Figuras 4.1 e 4.2 nota-se que os quadrados que simbolizam os elementos são maiores se comparados as imagens posteriores. Isso se deve ao fato dessas matrizes possuírem ordem de grandeza menor que as seguintes e todas estarem representadas em figuras do mesmo tamanho. Além disso, uma maior proporção colorida na Figura 4.1 indica uma densidade maior, já que espaços brancos representam zero.

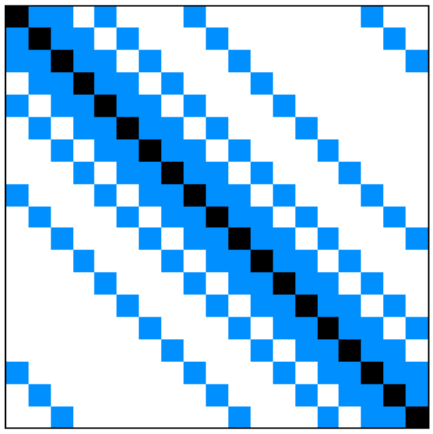


Figura 4.1: Padrão de esparsidade da matriz "*Trefethen_ 20b*".
Fonte: Davis e Hu (2011).

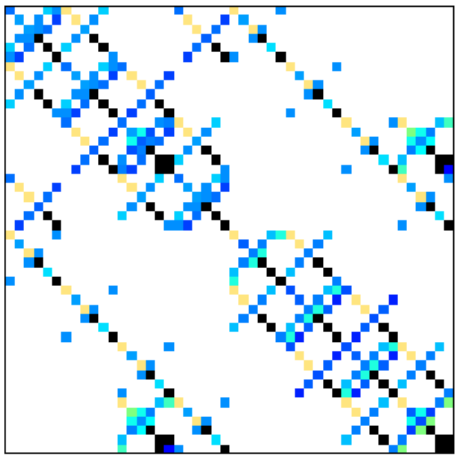


Figura 4.2: Padrão de esparsidade da matriz "*Bcsstk01*".
Fonte: Davis e Hu (2011).

Nos problemas "*Trefethen_*" (Figura 4.1), "*bcsstk01*" (Figura 4.2) "*bcsstm07*" (Figura 4.6 exibida adiante) e outros (como "*sherman_*" presente nas Figuras A.2, A.5 e A.8, "*rdb1250*" na Figura A.3 e "*cavity10*" na Figura A.4 presentes no Apêndice A) observa-se um padrão de linhas diagonais além da diagonal principal. Nessas matrizes tem-se superdiagonais (diagonais acima da diagonal principal) e subdiagonais (diagonais abaixo da diagonal principal).

A Figura 4.3 ilustra o problema "*bfwa62*", o qual não é simétrico (rever Tabela 4.2), embora sua imagem possa sugerir que seja. Isto é, embora sua distribuição visual de elementos seja espelhada em relação à diagonal principal, $a_{ij} \neq a_{ji}$, então sua matriz de coeficientes não é simétrica.

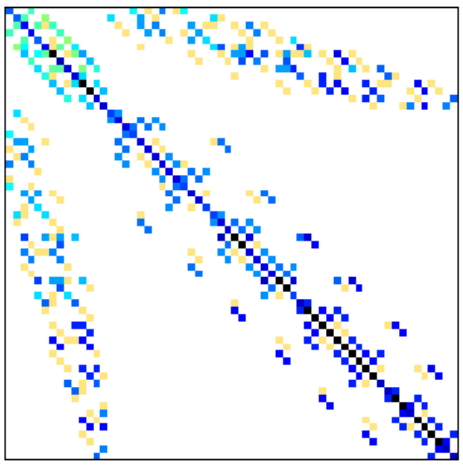


Figura 4.3: Padrão de esparsidade da matriz "*Bfwa62*".
Fonte: Davis e Hu (2011).

As figuras referentes aos problemas "*bcsstm_*" ("*bcsstm02*" ver Figura 4.4 e "*bcsstm25*" na Figura A.17 presente no Apêndice A) são matrizes diagonais; e por isso possuem apenas os elementos da diagonal principal.

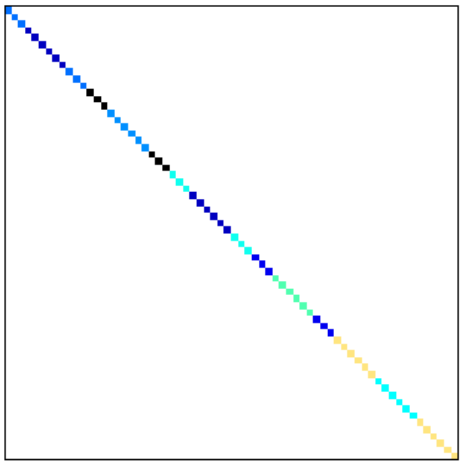


Figura 4.4: Padrão de esparsidade da matriz "*Bcsstm02*".
Fonte: Davis e Hu (2011).

Em "*Trefethen_150*" presente na Figura 4.5 e "*bcsstm07*" na Figura 4.6 nota-se um padrão de linhas diagonais além da diagonal principal, são matrizes com superdiagonais (diagonais acima da diagonal principal) e subdiagonais (diagonais abaixo da diagonal principal).

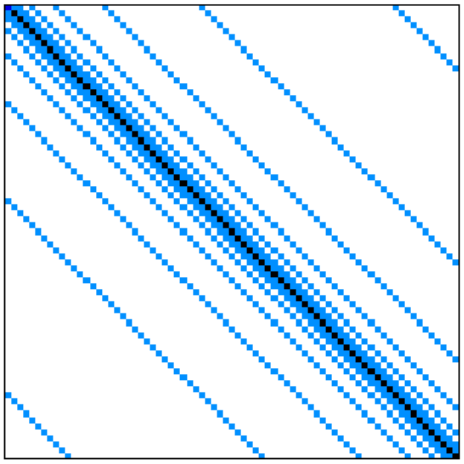


Figura 4.5: Padrão de esparsidade da matriz "*Trefethen_150*".
Fonte: Davis e Hu (2011).

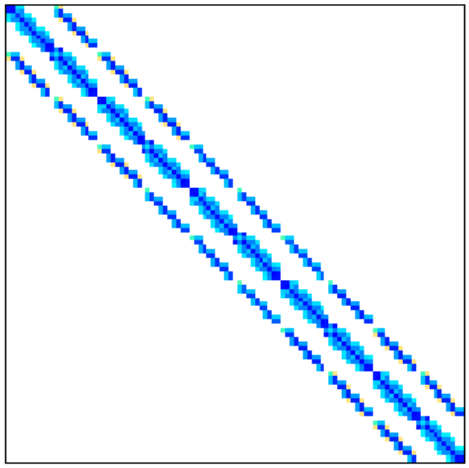


Figura 4.6: Padrão de esparsidade da matriz "*Bcsstm07*".
Fonte: Davis e Hu (2011).

Os problemas "*1138_ bus*" (ver Figura 4.7) e "*c-*" ("*c-19*" ver Figura 4.8, "*c-26*" na Figura A.6, "*c-28*" na Figura A.7, "*c-30*" na Figura A.9, "*c-31*" na Figura A.10, "*c-32*" ver Figura 4.9, "*c-33*" na Figura A.11, "*c-35*" na Figura A.12, "*c-34*" na Figura A.13, "*c-46*" na Figura A.15, "*c-47*" na Figura A.16, "*c-48*" na Figura A.18 e "*c-49*" na Figura A.19), apresentam um padrão de diagonal principal, e blocos superior e/ou inferior com elementos dispersos e/ou linhas diagonais.

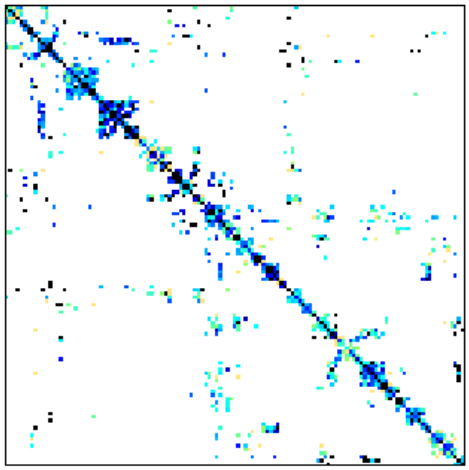


Figura 4.7: Padrão de esparsidade da matriz "*1138_ bus*".
Fonte: Davis e Hu (2011).

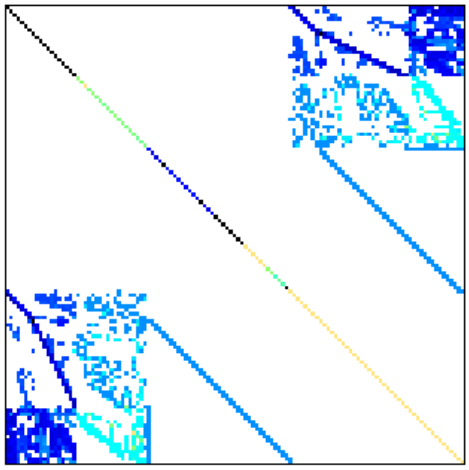


Figura 4.8: Padrão de esparsidade da matriz "*c-19*".
Fonte: Davis e Hu (2011).

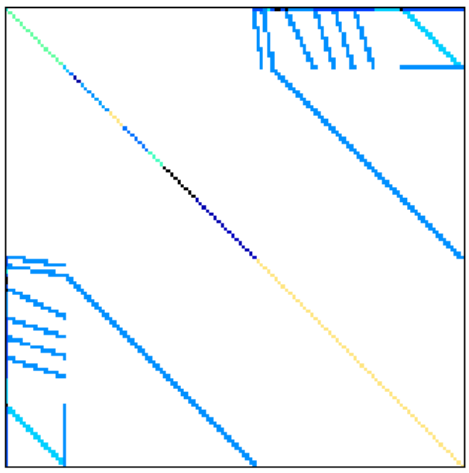


Figura 4.9: Padrão de esparsidade da matriz "*c-32*".
Fonte: Davis e Hu (2011).

Em todos os problemas, ressalta-se a presença da diagonal principal indispensável para que a matriz possua determinante não nulo e solução única possível. Observa-se ainda o tamanho e a esparsidade das matrizes estudadas. Afinal, tais características estão relacionadas ao espaço reservado na memória e ao tempo de execução para a solução de cada problema.

Capítulo 5

Resultados computacionais

Este capítulo aborda os arquivos de entrada e saída de dados, os testes em termos de memória utilizada, tempo de execução para resolução dos problemas e norma do resíduo gerado nas soluções. Após a apresentação dos resultados obtidos a partir de determinado método numérico, há uma comparação em relação as estruturas empregadas (indexada e de listas). Na última subseção, há um comparativo geral entre os métodos de fatoração **LU**, **QR** e de Cholesky e as estruturas indexada e de listas encadeadas.

5.1 Especificações e parâmetros

Todos os resultados apresentados foram obtidos através do compilador Borland, versão 5.02, com o uso de um computador pessoal cujas especificações são: processador Intel(R) Core(TM) i5-5200U CPU 2.20 GHz e 4 Gb de memória RAM.

A entrada de dados foi feita por meio de arquivos de texto contendo os elementos não nulos da matriz. A primeira linha do arquivo contém a ordem de grandeza e o número de elementos não nulos da matriz. Nas linhas seguintes é exibido o número de linha, número de coluna e o valor de cada elemento, nessa ordem. Dessa forma economiza memória, aumenta a eficiência nas operações e facilita o acesso aos elementos por linha, já que a maioria das operações ocorre linha a linha.

A Figura 5.1 apresenta um exemplo de arquivo de entrada, nesse caso, relativo ao problema "*cage3*" disponível em um repositório *online* (`https://sparse.tamu.edu/vanHeukelum/cage3`).

```
5 19
1 1 6.666667e-01
1 2 1.000369e-01
1 3 1.221853e-01
1 4 5.001844e-02
1 5 6.109267e-02
2 1 3.665560e-01
2 2 5.334071e-01
2 4 1.000369e-01
3 1 3.001107e-01
3 3 5.777040e-01
3 5 1.221853e-01
4 1 3.665560e-01
4 2 2.000738e-01
4 4 2.833149e-01
4 5 1.500553e-01
5 1 3.001107e-01
5 3 2.443707e-01
5 4 1.832780e-01
5 5 2.722407e-01
0.000000e+00
0.000000e+00
0.000000e+00
0.000000e+00
0.000000e+00
```

Figura 5.1: Exemplo de arquivo de entrada de dados referente ao problema "*cage3*".

Fonte: Elaboração própria.

Alguns dos problemas retirados do repositório *online* não apresentavam, a princípio, essa formatação. Nesses casos, precisaram ser convertidos a esse formato antes dos testes.

Nos arquivos de saída encontra-se o nome do problema, algumas de suas características como ordem de grandeza, número de elementos não nulos, densidade, esparsidade; os tempos computacionais gastos em cada etapa e o resíduo obtido nas soluções. A seguir, como exemplo, a Figura 5.2 apresenta o arquivo de saída do problema "*cage3*" usando o método de decomposição **LU** com estrutura de dados indexada.

```
Nome do arquivo: cage3.txt

Ordem de grandeza: 5
Quantidade de elementos diferentes de zero em A: 19
Densidade: 76 %
Esparsidade: 24 %
Tempo de leitura:0.000000e+00 segundos.
Tempo de decomposicao:0.000000e+00 segundos.
Tempo de resolução sistema inferior:0.000000e+00 segundos.
Tempo de resolução sistema superior:0.000000e+00 segundos.
Tempo de execução:0.000000e+00 segundos.
Norma do resíduo:0.000000e+00
```

Figura 5.2: Arquivo de saída de dados referente ao problema "*cage3*" - Decomposição **LU** com estrutura indexada.

Fonte: Elaboração própria.

Para testes utilizando listas encadeadas, foi contabilizado também o número de elementos não nulos presentes nas matrizes decompostas para cada método empregado. Ou seja, o número de elementos presentes nas listas que representam as matrizes **L**, **U**, **Q**, **R** e **C**.

A Figura 5.3 é dedicada a um exemplo de saída de dados de problemas que foram resolvidos utilizando decomposição **LU** com estrutura de dados em lista encadeada.

```
Nome do arquivo: cage3.txt

Ordem de grandeza: 5
Quantidade de elementos diferentes de zero em A: 19
Quantidade de elementos em L: 15
Quantidade de elementos em U: 15
Densidade: 76 %
Esparsidade: 24 %
Tempo de leitura:0.000000e+00 segundos.
Tempo de decomposicao:0.000000e+00 segundos.
Tempo de sistema inferior:0.000000e+00 segundos.
Tempo de sistema superior:0.000000e+00 segundos.
Tempo de execução:0.000000e+00 segundos.
Norma do resíduo:0.000000e+00
```

Figura 5.3: Arquivo de saída de dados referente ao problema "*cage3*" - Decomposição **LU** com estrutura de lista encadeada.

Fonte: Elaboração própria.

5.2 Memória e estrutura utilizada

Destaca-se que na estrutura indexada, as matrizes que resultaram da decomposição de **A**, assim como a própria matriz **A**, são armazenadas em *arrays* e consomem n^2 de memória. Primeiramente, os resultados a serem analisados são referentes ao uso da memória na estrutura de dados de lista encadeada.

Na estrutura de listas encadeadas somente os elementos não nulos da matriz **A** e das matrizes da decomposição para os métodos **LU**, **QR** ou de Cholesky são salvos, então o uso de memória é menor. Pois, como relatado anteriormente, o elemento não nulo juntamente com sua posição na linha e coluna, são armazenados em uma estrutura de célula ou nó. Isso é observado ao contar cada lista que representa as matrizes **A**, **L**, **U**, **Q**, **R** e **C** (para Cholesky) nos métodos considerados, como é mostrado a seguir.

Para a lista referente a matriz **A**, o número de nós equivale ao número de elementos não nulos da própria matriz (nz). Quanto às matrizes resultantes da decomposição, independente do método empregado, nota-se que nem sempre elas preservam a esparsidade de **A**. Por exemplo, no caso anterior de "*cage3*" a matriz **A** possui 19 elementos não nulos e as matrizes **L** e **U** possuem 15 elementos em cada uma delas. E isso é comentado adiante, visto que a esparsidade é uma característica importante para resolução de sistemas lineares da ordem de milhares de elementos.

Assim, busca-se avaliar o impacto do uso da estrutura indexada e da estrutura de lista encadeada também nas matrizes decompostas de cada caso (**L**, **U**, **Q**, **R** e **C**). Em matrizes não simétricas e que não sejam positivas definidas, o método de Cholesky não é utilizado. Nesses casos, a tabela apresenta um traço "-" ao invés de valores, devido a incapacidade de resolução do método.

Nas tabelas a seguir adotam-se as siglas:
n: ondem de grandeza do problema;
nz: número de elementos não nulos;
nl: número de elementos da lista que representa a matriz **L** da decomposição **LU**;
nu: número de elementos da lista que representa a matriz **U** da decomposição **LU**;
nq: número de elementos da lista que representa a matriz **Q** da decomposição **QR**;
nr: número de elementos da lista que representa a matriz **R** da decomposição **QR**;
nc: número de elementos da lista que representa a matriz **C** da decomposição de Cholesky.

A Tabela 5.1 apresenta os resultados relativos a memória dos problemas obtidos com o gerador aleatório. Os nomes *"tridiag"* e *"triang"* são abreviações para tridiagonal e triangular, respectivamente. A numeração presente no nome de cada problema indica sua dimensão.

Tabela 5.1: Número de elementos não nulos dos problemas do gerador aleatório.

Problema	n	nz	n^2	nl	nu	nq	nr	nc
tridiag100	100	298	10000	199	199	4246	297	-
triang100	100	5050	10000	100	5050	100	5050	-
cheia100	100	10000	10000	5050	5050	10000	5050	-
tridiag500	500	1498	250000	999	999	33897	1497	-
triang500	500	125250	250000	500	125250	500	125250	-
cheia500	500	250000	250000	125250	125250	250000	125250	-
tridiag1000	1000	2998	1000000	1999	1999	60151	2997	-
triang1000	1000	500500	1000000	1000	500500	1000	500500	-
cheia1000	1000	1000000	1000000	500500	500500	1000000	500500	-
tridiag1500	1500	4498	2250000	2999	2999	89988	4497	-
triang1500	1500	1125750	2250000	1500	1125750	1500	1125750	-
cheia1500	1500	2250000	2250000	1125750	1125750	2250000	1125750	-
tridiag2000	2000	5998	4000000	3999	3999	103386	5997	-
triang2000	2000	2001000	4000000	2000	2001000	2000	2001000	-
cheia2000	2000	4000000	4000000	2001000	2001000	-	-	-
tridiag3500	3500	10498	12250000	6999	6999	185234	10497	-
tridiag5000	5000	14998	25000000	9999	9999	257263	14997	-
tridiag6500	6500	19498	42250000	12999	12999	323818	19497	-
tridiag8000	8000	23998	64000000	15999	15999	-	-	-
Média	2015,79	597464,84	8738421,05	200991,63	398489	269004,88	327904,47	

Fonte: Elaboração própria.

Na fatoração **LU** de matrizes cheias ou tridiagonais nota-se que o número de elementos da matriz **L** e da matriz **U** foram iguais para cada caso estudado. Observa-se que para problemas com matrizes triangulares a matriz fatorada **L** é uma matriz diagonal.

Na fatoração **QR** de matrizes triangulares a matriz **Q** é uma matriz diagonal enquanto a matriz **R** tem o mesmo numero de elementos não nulos da matriz **A**. Já para matrizes cheias, a matriz **Q** também é cheia. Nota-se ainda que o número de elementos não nulos

da matriz fatorada **R** é igual para matriz triangular ou cheia, desde que sejam de mesma dimensão.

Dois dos problemas apresentados na Tabela 5.1 não foram resolvidos por fatoração **QR** dentro do tempo limite de 32 h. Além disso, vale destacar que os problemas do gerador aleatório exibidos na Tabela 5.1 não puderam ser resolvidos pelo método de Cholesky. Destaca-se que as matrizes decompostas têm ordem de grandeza muito inferior com relação as suas respectivas matrizes do problema original, principalmente para a decomposição **LU**.

A seguir estão os problemas oriundos do repositório *online*. Para facilitar a leitura dos dados e comentários, seus resultados foram divididos em 3 tabelas. A Tabela 5.2, apresenta problemas com ordem de grandeza que varia de 19 a 1000.

Tabela 5.2: Número de elementos não nulos dos problemas do repositório para $19 \leq n \leq 1000$.

Problema	n	nz	n^2	nl	nu	nq	nr	nc
Trefethen_20b	19	83	361	19	83	19	83	19
Trefethen_20	20	89	400	20	89	20	89	20
bcsstk01	48	224	2304	224	48	2027	952	877
bfw62a	62	450	3844	1228	1240	3470	1715	-
bcsstm02	66	66	4356	66	66	66	66	66
Trefethen_150	150	1095	22500	150	1095	150	1095	150
Trefethen_200	200	1545	40000	200	1545	200	1545	200
Trefethen_300	300	2489	90000	300	2489	300	2489	300
bcsstm06	420	420	176400	420	420	420	420	420
bcsstm07	420	3836	176400	420	3836	420	3836	420
Trefethen_500	500	4489	250000	500	4489	500	4489	500
Trefethen_700	700	6677	490000	700	6677	700	6677	700
sherman1	1000	3750	1000000	24985	24985	252612	51069	-
Média	300,38	1939,46	173581,92	2248,61	3620,15	20069,54	5732,69	333,82

Fonte: Elaboração própria.

Na maioria dos problemas observados, o processo de decomposição gera pelo menos uma matriz diagonal de mesma ordem de grandeza da matriz original. É o que ocorre em: *"Trefethen_20b"*, *"Trefethen_20"*, *"bcsstk01"*, *"bcsstm02"*, *"Trefethen_150"*, *"Trefethen_200"*, *"Trefethen_300"*, *"bcsstm06"*, *"bcsstm07"*, *"Trefethen_500"* e *"Trefethen_700"*.

Em alguns casos, as fatorações **LU** e **QR** geram matrizes decompostas com mesmo número de elementos não nulos da matriz **A** (preservando sua esparsidade). Isso é observado para os seguintes problemas: *"Trefethen_20b"*, *"Trefethen_20"*, *"bcsstm02"*, *"Trefethen_150"*, *"Trefethen_200"*, *"Trefethen_300"*, *"bcsstm06"*, *"bcsstm07"*, *"Trefethen_500"* e *"Trefethen_700"*.

Tal semelhança não é observada na decomposição de *"bcsstk01"* usando método **QR**; ou em ambas (**LU** e **QR**) para os problemas *"bfw62a"* e *"sherman1"*. Para esses problemas observa-se um aumento significativo de elementos não nulos, o que influencia significativamente no tempo de resolução desses problemas.

Observa-se que os problemas *"bcsstm02"* e *"bcsstm06"* são sistemas lineares cujas matrizes de coeficientes são diagonais assim como as matrizes resultantes de sua decomposição para todos os métodos empregados.

Dos problemas apresentados, dois deles não são passíveis de resolução pelo método Cholesky, são eles: *"bfw62a"* e *"sherman1"*, pois não atendem as condições (propriedades) necessárias para aplicação do método. A seguir, a Tabela 5.3 apresenta problemas do repositório referido, com ordem de grandeza variando de 1080 a 6537.

Tabela 5.3: Número de elementos não nulos dos problemas do repositório para $1080 \leq n \leq 6537$.

Problema	n	nz	n^2	nl	nu	nq	nr	nc
sherman2	1080	23094	1166400	133083	139945	701066	328551	-
bcsstm09	1083	1083	1172889	1083	1083	1083	1083	1083
bcsstm10	1086	22092	1179396	1086	11589	1086	11589	1086
sherman4	1104	3786	1218816	66816	66816	216027	102847	-
1138_bus	1138	2596	1295044	2596	1138	565224	99137	38312
rdb1250	1250	7300	1562500	61347	61347	841972	119037	-
c-19	2327	12072	5414929	2327	12072	2327	12072	-
cavity10	2597	76367	6744409	416603	417806	-	-	-
sherman5	3312	20793	10969344	571413	408556	-	-	-
bcsstm21	3600	3600	12960000	3600	3600	3600	3600	3600
c-26	4307	19422	18550249	19422	4307	-	-	-
c-28	4598	17594	21141604	4598	17594	4598	17594	-
sherman3	5005	20033	25050025	548483	548483	-	-	-
c-29	5033	2 4382	25331089	5033	24382	5033	24382	-
c-30	5321	35507	28313041	5321	35507	5321	35507	-
c-31	5339	41955	28504921	5339	41955	5339	41955	-
c-32	5975	30223	35700625	5975	30223	5975	30223	-
c-33	6317	31220	39904489	6317	31220	6317	31220	-
c-35	6537	34714	42732369	6537	34714	6537	34714	-
Média	3526,79	22517,53	16258533,63	98262,05	99596,68	158100,33	59567,4	11020,25

Fonte: Elaboração própria.

Para esse grupo de problemas apenas quatro apresentam solução via decomposição pelo método de Cholesky, são eles: *"bcsstm09"*, *"bcsstm10"*, *"1138_bus"* e *"bcsstm21"*. *"Bcsstm09"* e *"bcsstm21"* constituem sistemas cujas matrizes de coeficientes são diagonais, assim como a decomposição nos três métodos estudados. Isso que indica que a espasidade desses problemas permanece a mesma nas matrizes decompostas.

As matrizes decompostas de *"1138_bus"* (usando decomposição **LU**), *"c-19"*, *"bcsstm21"*, *"c-26"* (usando decomposição **LU**), *"c-28"*, *"c-29"*, *"c-30"*, *"c-31"*, *"c-32"*, *"c-33"* e *"c-36"* possuem uma matriz diagonal e outra de mesma esparsidade da matriz **A**.

Os problemas *"cavity10"*, *"sherman5"*, *"c-26"* e *"sherman3"* não foram resolvidos dentro do tempo limite de teste estabelecido (32 horas) para o método de decomposição **QR**.

Vale observar que em algumas decomposições, as matrizes geradas possuem um maior número de elementos não nulos que a matriz **A**. Isso ocorre em *"sherman2"*, *"sherman4"*, *"1138_bus"* (na decomposição **QR**), *"rdb1250"*, *"cavity10"*, *"sherman5"* e *"sherman3"*. A Tabela 5.4 apresenta os dados para problemas de dimensão entre 6611 a 21132.

Tabela 5.4: Número de elementos não nulos dos problemas do repositório para $6611 \leq n \leq 21132$.

Problema	n	nz	n^2	nl	nu	nq	nr	nc
c-34	6611	35472	43705321	6611	35472	6611	35472	-
c-36	7479	36710	55935441	7479	36710	7479	36710	-
c-38	8127	42908	66048129	8127	42908	8127	42908	-
c-37	8204	41440	67305616	8204	41440	8204	41440	-
c-39	9271	73041	85951441	9271	62929	9271	62929	-
c-41	9769	55757	95433361	9769	55702	9769	55702	-
c-40	9941	45721	98823481	9941	45721	9941	45721	-
c-42	10471	60378	109641841	10471	60378	10471	60378	-
c-44	10728	47864	115089984	10728	47864	10728	47864	-
c-43	11125	67400	123765625	11125	67392	11125	67392	-
c-45	13206	93829	174398436	13206	93829	13206	93829	-
pres_poisson	14822	356313	219691684	365313	14822	-	-	5061932
c-46	14913	72655	222397569	14913	72655	14913	72655	-
c-47	15343	113372	235407649	15343	113372	15343	113372	-
bcsstm25	15439	15439	238362721	15439	15439	15439	15439	15439
c-48	18354	92217	336869316	18354	92217	-	-	-
c-49	21132	89087	446561424	21132	89086	-	-	-
Média	12055	78800,18	160905237,6	32672,12	58113,88	10759,07	56557,93	2538685,5

Fonte: Elaboração própria.

Para o conjunto de problemas apresentado na Tabela 5.4, nota-se que apenas dois problemas atenderam aos pré-requisitos do método de Cholesky e puderam ser solucionados pelo mesmo. Já os problemas *"pres_poisson"*, *"c-48"* e *"c-49"* não foram resolvidos pelo método de decomposição **QR** usando listas encadeadas dentro do limite de tempo do teste (32 h).

Vale salientar que os problemas *"c-34"*, *"c-36"*, *"c-38"*, *"c-37"*, *"c-39"*, *"c-41"*, *"c-40"*, *"c-42"*, *"c-44"*, *"c-43"*, *"c-45"*, *"c-46"*, *"c-47"*, *"c-48"* e *"c-49"* possuem uma matriz diagonal e uma matriz com mesmo número de elementos não nulos de sua matriz original, tanto na decomposição **LU** como na decomposição **QR**.

No problema *"pres_poisson"* a decomposição **LU** criou matrizes com mesmo número de elementos não nulos da original. Inclusive, embora o problema tenha sido resolvido pelo método de Cholesky, o número de elementos não nulos da matriz gerada foi bem maior que da matriz **A**. Na Tabela 5.4, apenas *"bcsstm25"* é uma matriz diagonal e teve como decomposições matrizes diagonais para todos os métodos estudados (fatoração **LU**, fatoração **QR** e fatoração de Cholesky).

A seguir, tem-se os resultados referentes a cada grupo de problemas observado o método numérico empregado e a estrutura de dados utilizada.

5.3 Resultados do gerador aleatório

Os problemas do gerador aleatório foram solucionados com os métodos de fatoração **LU** e **QR** usando estrutura de dados indexada e de listas encadeadas. O método de Cholesky não pôde ser utilizado para esses problemas.

Para apresentação dos resultados adotam-se as seguintes abreviações:
TL: tempo para leitura dos dados em segundos;
TD: tempo para decomposição da matriz **A** em segundos (decomposição em **L** e **U** na fatoração **LU**, em **Q** e **R** na fatoração **QR** e em **C** na fatoração de Cholesky);
$TRS1$: tempo de resolução do sistema linear 1 em segundos (sistema triangular inferior na fatoração **LU**, ortogonal na fatoração QR ou triangular superior na fatoração de Cholesky);
$TRS2$: tempo de resolução do sistema linear 2 em segundos (sistema triangular superior na fatoração **LU** e **QR** ou triangular inferior na fatoração de Cholesky);
TT: tempo total de execução do programa em segundos;
$||R|| = ||x_{exata} - x_{numerica}||$: resíduo na norma euclidiana.

Por simplificação, onde o tempo obtido foi "$0,000000 \times 10^0$", nas tabelas encontra-se apenas "0*" (zero asterisco). Além disso, os resultados dos testes foram organizados em ordem crescente de acordo com o tamanho do problema e apresentam - além dos tempos e norma do resíduo - algumas de suas características.

A Tabela 5.5 apresenta os resultados obtidos pelos testes para problemas do gerador aleatório utilizando a decomposição **LU** e estrutura de dados indexada.

Tabela 5.5: Decomposição **LU** utilizando estrutura indexada - problemas do gerador aleatório.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
tridiag100	100	298	2,98	97,02	0*	0*	0*	0*	0*	$4,21 \times 10^{-16}$
triang100	100	5050	50,50	49,50	0*	0*	0*	0*	0*	$1,70 \times 10^{-15}$
cheia100	100	10000	100	0	0*	0*	0*	0*	0*	$2,22 \times 10^{-15}$
tridiag500	500	1498	0,60	99,40	$1,50 \times 10^{-2}$	$3,13 \times 10^{-1}$	0*	0*	$3,28 \times 10^{-1}$	$1,09 \times 10^{-15}$
triang500	500	125250	50,10	49,90	$9,40 \times 10^{-2}$	$4,53 \times 10^{-1}$	0*	0*	$5,47 \times 10^{-1}$	$9,12 \times 10^{-15}$
cheia500	500	250000	100	0	$1,87 \times 10^{-1}$	$2,19 \times 10^{-1}$	0*	0*	$4,06 \times 10^{-1}$	$8,49 \times 10^{-15}$
tridiag1000	1000	2998	0,30	99,70	$3,20 \times 10^{-2}$	$2,46 \times 10^{0}$	$1,60 \times 10^{-2}$	0*	$2,51 \times 10^{0}$	$1,65 \times 10^{-15}$
triang1000	1000	500500	50,05	49,95	$4,54 \times 10^{-1}$	$6,32 \times 10^{0}$	0*	$1,50 \times 10^{-2}$	$6,79 \times 10^{0}$	$1,80 \times 10^{-14}$
cheia1000	1000	1000000	100	0	$7,34 \times 10^{-1}$	$2,12 \times 10^{0}$	0*	0*	$2,86 \times 10^{0}$	$1,65 \times 10^{-14}$
tridiag1500	1500	4498	0,20	99,80	$4,70 \times 10^{-2}$	$9,67 \times 10^{0}$	$3,10 \times 10^{-2}$	0*	$9,75 \times 10^{0}$	$1,97 \times 10^{-15}$
triang1500	1500	1125750	50,03	49,97	$1,03 \times 10^{0}$	$2,74 \times 10^{1}$	$3,10 \times 10^{-2}$	$1,60 \times 10^{-2}$	$2,85 \times 10^{1}$	$2,51 \times 10^{-14}$
cheia1500	1500	2250000	100	0	$2,00 \times 10^{0}$	$9,71 \times 10^{0}$	0*	$1,60 \times 10^{-2}$	$1,17 \times 10^{1}$	$2,69 \times 10^{-14}$
tridiag2000	2000	5998	0,15	99,85	$7,80 \times 10^{-2}$	$2,45 \times 10^{1}$	$6,20 \times 10^{-2}$	$1,60 \times 10^{-2}$	$2,47 \times 10^{1}$	$2,28 \times 10^{-15}$
triang2000	2000	2001000	50,02	49,97	$1,59 \times 10^{0}$	$7,19 \times 10^{1}$	$6,20 \times 10^{-2}$	0*	$7,36 \times 10^{1}$	$3,35 \times 10^{-14}$
cheia2000	2000	4000000	100	0	$3,06 \times 10^{0}$	$2,30 \times 10^{1}$	$1,60 \times 10^{-2}$	0*	$2,60 \times 10^{1}$	$3,53 \times 10^{-14}$
tridiag3500	3500	10498	0,08	99,91	$1,87 \times 10^{-1}$	$1,60 \times 10^{2}$	$3,28 \times 10^{-1}$	$3,10 \times 10^{-2}$	$1,60 \times 10^{2}$	$2,99 \times 10^{-15}$
tridiag5000	5000	14998	0,06	99,94	$3,60 \times 10^{-1}$	$4,68 \times 10^{2}$	$9,07 \times 10^{-1}$	$6,20 \times 10^{-2}$	$4,70 \times 10^{2}$	$3,59 \times 10^{-15}$
tridiag6500	6500	19498	0,05	99,95	$5,78 \times 10^{-1}$	$1,11 \times 10^{3}$	$1,96 \times 10^{0}$	$9,40 \times 10^{-2}$	$1,12 \times 10^{3}$	$4,21 \times 10^{-15}$
tridiag8000	8000	23998	0,04	99,96	$8,91 \times 10^{-1}$	$2,23 \times 10^{3}$	$3,66 \times 10^{0}$	$1,87 \times 10^{-1}$	$2,23 \times 10^{3}$	$4,50 \times 10^{-15}$
Média	2015,79	597464,84	39,74	60,25	$5,97 \times 10^{-1}$	$2,18 \times 10^{2}$	$3,72 \times 10^{-1}$	$2,00 \times 10^{2}$	$2,19 \times 10^{2}$	$1,05 \times 10^{-14}$

Fonte: Elaboração própria.

Na Tabela 5.5 tem-se que o tempo para solução dos problemas variou de $0,000000 \times 10^0 s$ a $2,23 \times 10^3 s$ (ou 37 minutos) e a norma do resíduo obtido nas soluções apresentadas foi mínimo de $4,21 \times 10^{-16}$ e máxima de $3,53 \times 10^{-14}$.

Nota-se ainda que para os problemas de mesma ordem de grandeza e diferentes densidades, em todos os casos apresentados com densidade maior que 100, os problemas com matrizes triangulares tiveram um tempo total maior em relação a sistemas lineares com matrizes cheias ou tridiagonais. Isso ocorreu principalmente devido ao tempo de decomposição dessas matrizes ser alto, cerca de 99,54% do tempo total. Mesmo para matrizes cheias, onde o tempo de leitura foi maior, o tempo de decomposição foi inferior ao tempo de decomposição de matrizes triangulares. Assim, para problemas de mesma ordem de

grandeza, os problemas solucionados em menos tempo foram: os tridiagonais, cheios e triangulares, respectivamente.

A Tabela 5.6 mostra os problemas do gerador aleatório resolvidos com o método de fatoração **LU** e a estrutura de dados de lista encadeada.

Tabela 5.6: Decomposição **LU** utilizando estrutura de listas encadeadas - problemas do gerador aleatório.

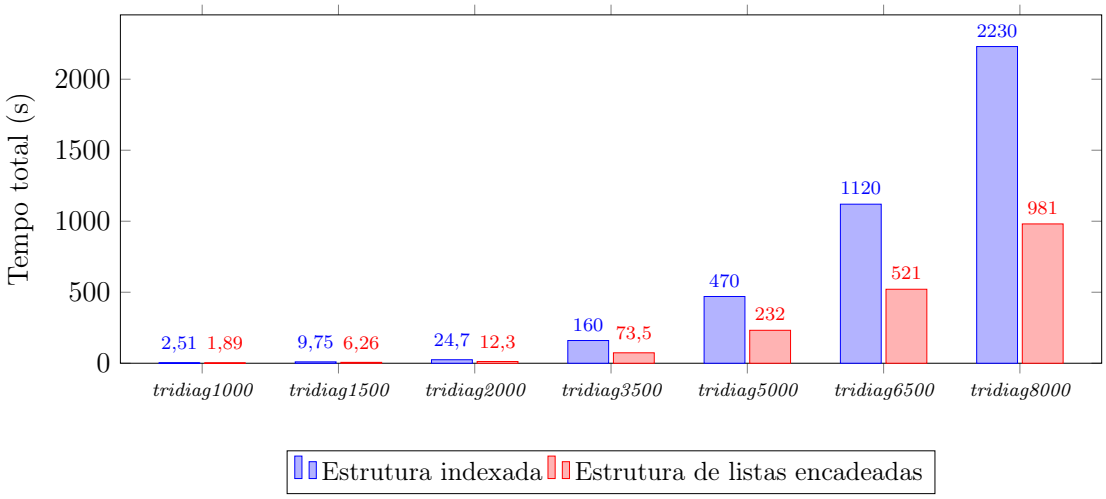
Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
tridiag100	100	298	2,98	97,02	0*	0*	0*	0*	0*	$4,21 \times 10^{-16}$
triang100	100	5050	50,50	49,50	$1,50 \times 10^{-2}$	$3,20 \times 10^{-2}$	0*	0*	$4,70 \times 10^{-2}$	$1,70 \times 10^{-15}$
cheia100	100	10000	100	0	0*	$6,20 \times 10^{-2}$	0*	0*	$6,20 \times 10^{-2}$	$2,22 \times 10^{-15}$
tridiag500	500	1498	0,60	99,40	0*	$2,03 \times 10^{-1}$	0*	0*	$2,03 \times 10^{-1}$	$1,09 \times 10^{-15}$
triang500	500	125250	50,10	49,90	$1,41 \times 10^{-1}$	$1,35 \times 10^{1}$	0*	0*	$1,37 \times 10^{1}$	$9,12 \times 10^{-15}$
cheia500	500	250000	100	0	$2,81 \times 10^{-1}$	$4,00 \times 10^{1}$	0*	0*	$4,03 \times 10^{1}$	$8,49 \times 10^{-15}$
tridiag1000	1000	2998	0,30	99,70	0*	$1,89 \times 10^{0}$	0*	0*	$1,89 \times 10^{0}$	$1,65 \times 10^{-15}$
triang1000	1000	500500	50,05	49,95	$7,66 \times 10^{-1}$	$2,97 \times 10^{2}$	0*	0*	$2,97 \times 10^{2}$	$1,80 \times 10^{-14}$
cheia1000	1000	1000000	100	0	$1,54 \times 10^{0}$	$8,77 \times 10^{2}$	0*	0*	$8,78 \times 10^{2}$	$1,65 \times 10^{-14}$
tridiag1500	1500	4498	0,20	99,80	0*	$6,26 \times 10^{0}$	0*	0*	$6,26 \times 10^{0}$	$1,97 \times 10^{-15}$
triang1500	1500	1125750	50,03	49,97	$2,14 \times 10^{0}$	$1,51 \times 10^{3}$	0*	0*	$1,51 \times 10^{3}$	$2,51 \times 10^{-14}$
cheia1500	1500	2250000	100	0	$5,46 \times 10^{0}$	$4,98 \times 10^{3}$	$3,10 \times 10^{-2}$	0*	$4,99 \times 10^{3}$	$2,69 \times 10^{-14}$
tridiag2000	2000	5998	0,15	99,85	$1,60 \times 10^{-2}$	$1,23 \times 10^{1}$	0*	0*	$1,23 \times 10^{1}$	$2,28 \times 10^{-15}$
triang2000	2000	2001000	50,02	49,97	$3,92 \times 10^{0}$	$4,18 \times 10^{3}$	0*	$1,60 \times 10^{-2}$	$4,19 \times 10^{3}$	3.35×10^{-14}
cheia2000	2000	4000000	100	0	$1,05 \times 10^{1}$	$1,53 \times 10^{4}$	$4,70 \times 10^{-2}$	$1,60 \times 10^{-2}$	$1,54 \times 10^{4}$	$3,53 \times 10^{-14}$
tridiag3500	3500	10498	0,08	99,91	0*	$7,35 \times 10^{1}$	$3,20 \times 10^{-2}$	0*	$7,35 \times 10^{1}$	$2,99 \times 10^{-15}$
tridiag5000	5000	14998	0,06	99,94	$1,60 \times 10^{-2}$	$2,31 \times 10^{2}$	$3,10 \times 10^{-2}$	0*	$2,32 \times 10^{2}$	$3,59 \times 10^{-15}$
tridiag6500	6500	19498	0,05	99,95	$3,10 \times 10^{-2}$	$5,21 \times 10^{2}$	$6,30 \times 10^{-2}$	0*	$5,21 \times 10^{2}$	$4,21 \times 10^{-15}$
tridiag8000	8000	23998	0,04	99,96	$3,20 \times 10^{-2}$	$9,81 \times 10^{2}$	$9,40 \times 10^{-2}$	0*	$9,81 \times 10^{2}$	$4,50 \times 10^{-15}$
Média	2015,79	597464,84	39,74	60,25	1,31	$1,527 \times 10^{3}$	$1,57 \times 10^{-2}$	$1,68 \times 10^{-3}$	$1,53 \times 10^{3}$	$1,05 \times 10^{-14}$

Fonte: Elaboração própria.

O tempo de decomposição exerce grande influência no tempo total de resolução dos problemas (cerca de 99,80%), visto que em muitos problemas o tempo de leitura ou mesmo de resolução dos sistemas gerados é próximo a zero. Comparando as Tabelas 5.5 e 5.6 nota-se uma redução nos tempos de solução dos problemas tridiagonais mantendo a norma dos resíduos muito semelhante quando usada a estrutura de listas encadeadas.

A Figura 5.4 compara o tempo total para solução dos problemas tridiagonais com dimensão entre 1000 e 8000 do gerador aleatório.

Figura 5.4: Tempo de resolução de problemas tridiagonais utilizando decomposição **LU**, estrutura indexada × listas encadeadas.

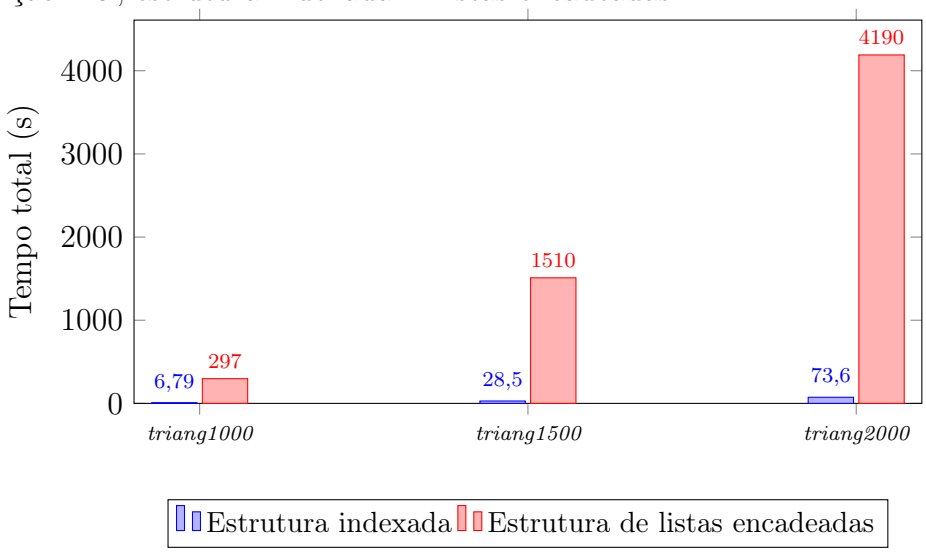


Fonte: Elaboração própria.

Nota-se que para os problemas tridiagonais estudados com fatoração **LU**, a estrutura de listas encadeadas fornece resultados semelhantes em termos da norma de resíduo porém, com tempo de resolução significativamente menor, como esperado.

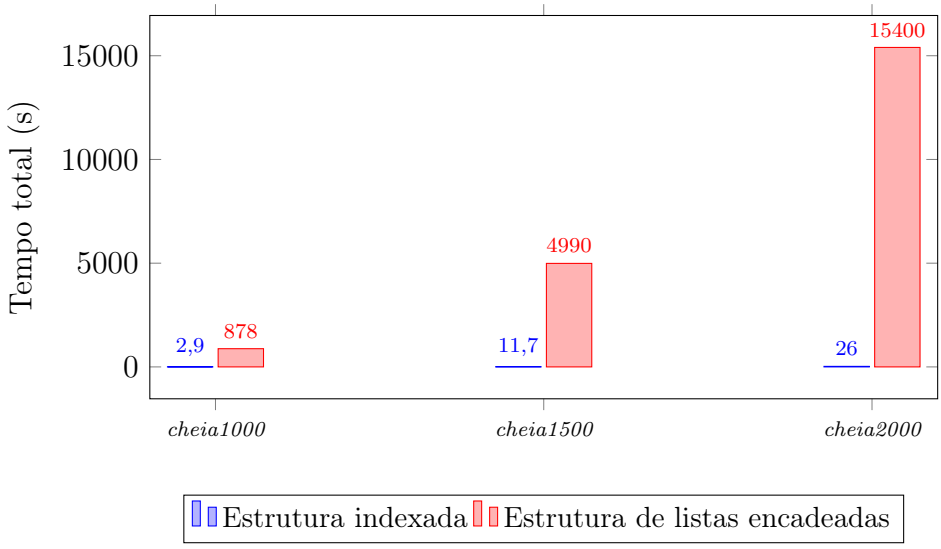
Quando a densidade de elementos cresce muito (de 50% a 100%) a estrutura lista encadeada tem desempenho significativamente inferior a estrutura indexada, como pode ser visto na Figura 5.5 e na Figura 5.6.

Figura 5.5: Tempo de resolução de problemas triangulares ($D \approx 50\%$) utilizando decomposição **LU**, estrutura indexada × listas encadeadas.



Fonte: Elaboração própria.

Figura 5.6: Tempo de resolução de problemas cheios ($D = 100\%$) utilizando decomposição **LU**, estrutura indexada × listas encadeadas.



Fonte: Elaboração própria.

Portanto, à medida que o número de elementos não nulos aumenta muito, como ocorre em matrizes triangulares ou cheias, a estrutura de listas passa a gastar um tempo total de resolução significativamente maior que a estrutura indexada. Assim, para matrizes densas, essa não é a melhor opção de estrutura, como mostrado nas Figuras 5.5 com matrizes triangulares e 5.6 com matrizes cheias.

A seguir, encontram-se na Tabela 5.7 os resultados dos problemas do gerador aleatório solucionados com o método de fatoração **QR** e estrutura de dados indexada.

Tabela 5.7: Decomposição **QR** utilizando estrutura indexada - problemas do gerador aleatório.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
tridiag100	100	298	2,98	97,02	0*	$1,50 \times 10^{-2}$	0*	0*	$1,50 \times 10^{-2}$	$7,54 \times 10^{-16}$
triang100	100	5050	50,50	49,50	$1,60 \times 10^{-2}$	0*	0*	0*	$1,60 \times 10^{-2}$	$1,70 \times 10^{-15}$
cheia100	100	10000	100	0	0*	0*	0*	0*	0*	$3,34 \times 10^{-15}$
tridiag500	500	1498	0,60	99,40	0*	$7,03 \times 10^{-1}$	0*	0*	$7,03 \times 10^{-1}$	$1,60 \times 10^{-15}$
triang500	500	125250	50,10	49,90	$1,41 \times 10^{-1}$	$5,47 \times 10^{-1}$	0*	0*	$6,88 \times 10^{-1}$	$1,36 \times 10^{-14}$
cheia500	500	250000	100	0	$1,87 \times 10^{-1}$	$5,63 \times 10^{-1}$	0*	0*	$7,50 \times 10^{-1}$	$1,62 \times 10^{-14}$
tridiag1000	1000	2998	0,30	99,70	$3,10 \times 10^{-2}$	$9,82 \times 10^{0}$	0*	0*	$9,85 \times 10^{0}$	$2,43 \times 10^{-15}$
triang1000	1000	500500	50,05	49,95	$4,53 \times 10^{-1}$	$1,28 \times 10^{1}$	0*	0*	$1,33 \times 10^{1}$	$1,80 \times 10^{-14}$
cheia1000	1000	1000000	100	0	$7,35 \times 10^{-1}$	$7,51 \times 10^{0}$	0*	0*	$8,25 \times 10^{0}$	$4,43 \times 10^{-14}$
tridiag1500	1500	4498	0,20	99,80	$6,30 \times 10^{-2}$	$4,55 \times 10^{1}$	0*	0*	$4,56 \times 10^{1}$	$2,91 \times 10^{-15}$
triang1500	1500	1125750	50,03	49,97	$1,06 \times 10^{0}$	$6,46 \times 10^{1}$	0*	$1,60 \times 10^{-2}$	$6,57 \times 10^{1}$	$2,51 \times 10^{-14}$
cheia1500	1500	2250000	100	0	$2,02 \times 10^{0}$	$4,29 \times 10^{1}$	0*	0*	$4,49 \times 10^{1}$	$5,68 \times 10^{-14}$
tridiag2000	2000	5998	0,15	99,85	$9,40 \times 10^{-2}$	$1,09 \times 10^{2}$	0*	$1,60 \times 10^{-2}$	$1,09 \times 10^{2}$	$3,25 \times 10^{-15}$
triang2000	2000	2001000	50,02	49,97	$1,59 \times 10^{0}$	$1,61 \times 10^{2}$	0*	$1,50 \times 10^{-2}$	$1,63 \times 10^{2}$	$3,35 \times 10^{-14}$
cheia2000	2000	4000000	100	0	$3,12 \times 10^{0}$	$1,06 \times 10^{2}$	0*	0*	$1,09 \times 10^{2}$	$7,81 \times 10^{-14}$
tridiag3500	3500	10498	0,08	99,91	$2,34 \times 10^{-1}$	$8,06 \times 10^{2}$	0*	$3,20 \times 10^{-2}$	$8,06 \times 10^{2}$	$4,42 \times 10^{-15}$
tridiag5000	5000	14998	0,06	99,94	$4,69 \times 10^{-1}$	$3,23 \times 10^{3}$	0*	$7,90 \times 10^{-2}$	$3,23 \times 10^{3}$	$5,33 \times 10^{-15}$
tridiag6500	6500	19498	0,05	99,95	$7,34 \times 10^{-1}$	$8,55 \times 10^{3}$	0*	$1,41 \times 10^{-1}$	$8,55 \times 10^{3}$	$6,08 \times 10^{-15}$
tridiag8000	8000	23998	0,04	99,96	-	-	-	-	-	-
Média	2015,79	597464,84	39,74	60,25	$6,08 \times 10^{-1}$	$7,30 \times 10^{2}$	0*	$1,66 \times 10^{-2}$	$7,31 \times 10^{2}$	$1,76 \times 10^{-14}$

Fonte: Elaboração própria.

A Tabela 5.7 mostra que o tempo total de execução ficou entre $0,000000 \times 10^0 s$ e $8,55 \times 10^3 s$ cerca de 2 horas e 22 minutos (excluindo o problema *"tridiag800"* que não foi resolvido dentro do tempo limite de teste, 32 h). A norma do resíduo variou de $7,54 \times 10^{-16}$ a $7,81 \times 10^{-14}$ para o pior caso com matriz cheia de dimensão 2000.

Também nota-se novamente que o tempo de decomposição da matriz representa a maior parte do tempo total da solução (em torno de 99,86%), já que é bem maior que os tempos de leitura ou solução dos sistemas criados. Agora, a Tabela 5.8 apresenta os resultados para os mesmos problemas resolvidos usando a estrutura de listas encadeadas.

Tabela 5.8: Decomposição **QR** utilizando estrutura de listas encadeadas - problemas do gerador aleatório.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
tridiag100	100	298	2,98	97,02	0*	$9,40 \times 10^{-2}$	0*	0*	$9,40 \times 10^{-2}$	$7,45 \times 10^{-16}$
triang100	100	5050	50,50	49,50	$1,60 \times 10^{-2}$	$6,20 \times 10^{-2}$	0*	0*	$7,80 \times 10^{-2}$	$1,70 \times 10^{-15}$
cheia100	100	10000	100	0	0*	$5,47 \times 10^{-1}$	0*	0*	$5,47 \times 10^{-1}$	$3,34 \times 10^{-15}$
tridiag500	500	1498	0,60	99,40	0*	$3,17 \times 10^{1}$	$1,10 \times 10^{-1}$	0*	$3,18 \times 10^{1}$	$1,60 \times 10^{-15}$
triang500	500	125250	50,10	49,90	$1,72 \times 10^{-1}$	$4,73 \times 10^{1}$	0*	0*	$4,75 \times 10^{1}$	$9,12 \times 10^{-15}$
cheia500	500	250000	100	0	$4,84 \times 10^{-1}$	$8,01 \times 10^{2}$	$3,20 \times 10^{0}$	$1,60 \times 10^{-2}$	$8,05 \times 10^{2}$	$1,62 \times 10^{-14}$
tridiag1000	1000	2998	0,30	99,70	$1,60 \times 10^{-2}$	$2,31 \times 10^{2}$	$3,44 \times 10^{-1}$	0*	$2,31 \times 10^{2}$	$2,43 \times 10^{-15}$
triang1000	1000	500500	50,05	49,95	$1,27 \times 10^{0}$	$1,00 \times 10^{3}$	$1,60 \times 10^{-2}$	0*	$1,00 \times 10^{3}$	$1,81 \times 10^{-14}$
cheia1000	1000	1000000	100	0	$3,68 \times 10^{0}$	$1,43 \times 10^{4}$	$2,79 \times 10^{1}$	0*	$1,43 \times 10^{4}$	$4,43 \times 10^{-14}$
tridiag1500	1500	4498	0,20	99,80	$1,60 \times 10^{-2}$	$9,21 \times 10^{2}$	$9,22 \times 10^{-1}$	0*	$9,21 \times 10^{2}$	$2,91 \times 10^{-15}$
triang1500	1500	1125750	50,03	49,97	$4,22 \times 10^{0}$	$5,06 \times 10^{3}$	$1,50 \times 10^{-2}$	0*	$5,07 \times 10^{3}$	$2,51 \times 10^{-14}$
cheia1500	1500	2250000	100	0	$1,26 \times 10^{1}$	$6,81 \times 10^{4}$	$1,03 \times 10^{2}$	$1,60 \times 10^{-2}$	$6,83 \times 10^{4}$	$5,68 \times 10^{-14}$
tridiag2000	2000	5998	0,15	99,85	0*	$1,70 \times 10^{3}$	$1,44 \times 10^{0}$	0*	$1,70 \times 10^{3}$	$3,25 \times 10^{-15}$
triang2000	2000	2001000	50,02	49,97	$8,35 \times 10^{0}$	$1,47 \times 10^{4}$	0*	$1,60 \times 10^{-2}$	$1,47 \times 10^{4}$	3.35×10^{-14}
cheia2000	2000	4000000	100	0	-	-	-	-	-	-
tridiag3500	3500	10498	0,08	99,91	$1,60 \times 10^{-2}$	$1,31 \times 10^{4}$	$6,79 \times 10^{0}$	0*	$1,32 \times 10^{4}$	$4,42 \times 10^{-15}$
tridiag5000	5000	14998	0,06	99,94	$1,50 \times 10^{-2}$	$4,19 \times 10^{4}$	$1,45 \times 10^{1}$	0	$4,20 \times 10^{4}$	$5,33 \times 10^{-15}$
tridiag6500	6500	19498	0,05	99,95	$3,20 \times 10^{-2}$	$9,43 \times 10^{4}$	$2,37 \times 10^{1}$	0*	$9,43 \times 10^{4}$	$6,08 \times 10^{-15}$
tridiag8000	8000	23998	0,04	99,96	-	-	-	-	-	-
Média	2015,79	597464,84	39,74	60,25	1,82	$1,51 \times 10^{4}$	$1,07 \times 10^{1}$	$2,82 \times 10^{-3}$	$1,51 \times 10^{4}$	$1,38 \times 10^{-14}$

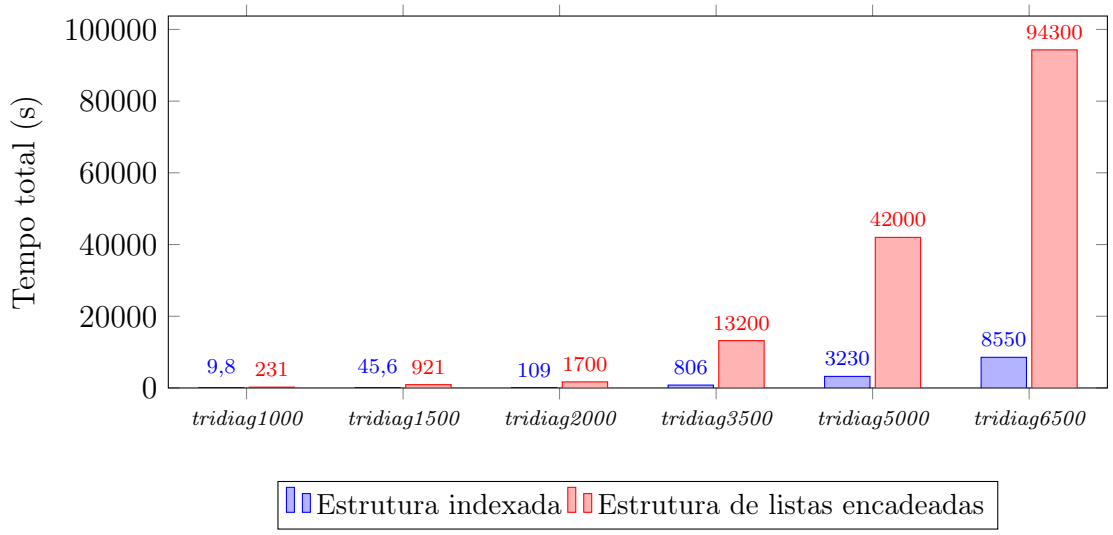
Fonte: Elaboração própria.

Nota-se da Tabela 5.8 que os tempos foram significativamente maiores que os anteriores utilizando a estrutura indexada, por exemplo. O problema mais rápido foi solucionado em $7,80 \times 10^{-2}$, enquanto alguns sequer foram resolvidos dentro do tempo limite de teste de 32 horas, como *"cheia2000"* e *"tridiag8000"*.

A norma dos resíduos encontrados permanece bem semelhante àquela obtida com a estrutura indexada, variando entre $7,45 \times 10^{-16}$ a $5,68 \times 10^{-14}$. Observa-se que, no geral, a decomposição **QR** necessitou de mais tempo para resolver os problemas em relação a decomposição **LU** (veja Tabela 5.6).

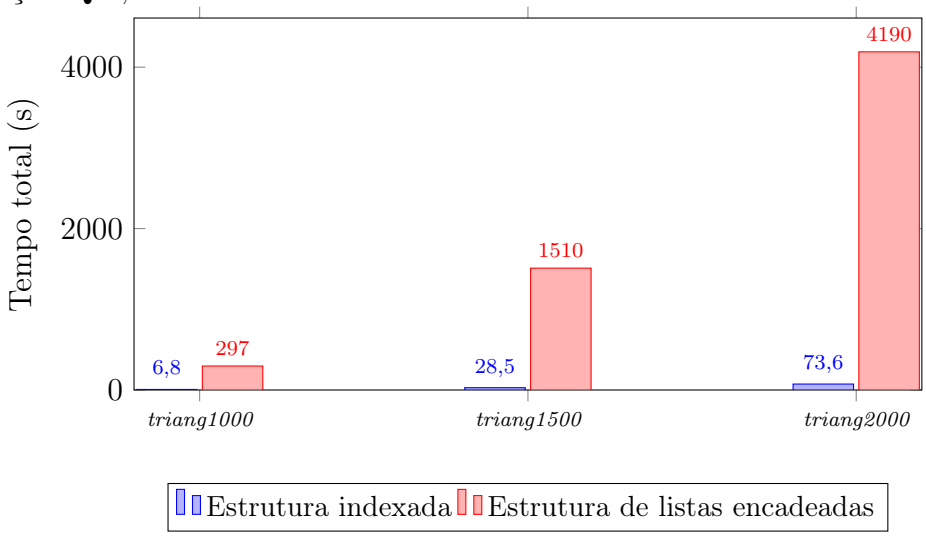
As Figuras 5.7, 5.8 e 5.9 mostram como o tempo de resolução dos problemas em todas as matrizes (tridiagonal, triangular e cheia) usando a fatoração **QR** com estrutura de listas encadeadas foi superior a estrutura indexada.

Figura 5.7: Tempo de resolução de problemas tridiagonais utilizando decomposição **QR**, estrutura indexada × listas encadeadas.



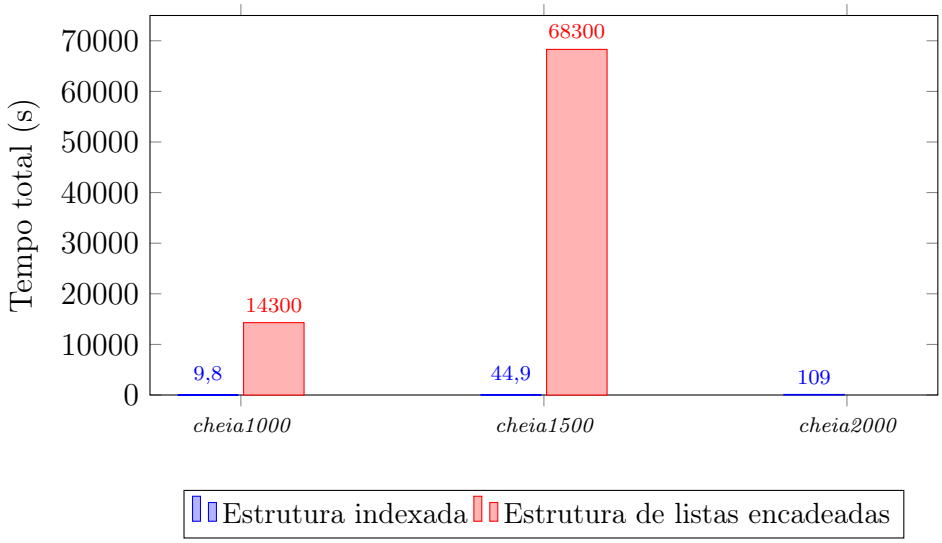
Fonte: Elaboração própria.

Figura 5.8: Tempo de resolução de problemas triangulares ($D \approx 50\%$) utilizando decomposição **QR**, estrutura indexada × listas encadeadas.



Fonte: Elaboração própria.

Figura 5.9: Tempo de resolução de problemas cheios ($D \approx 100\%$) utilizando decomposição **QR**, estrutura indexada $\times$ listas encadeadas.



Fonte: Elaboração própria.

Para as matrizes cheias como mostrado na Figura 5.9, o método **QR** com a estrutura de listas encadeadas sequer foi capaz de resolver o problema *"cheia2000"* dentro do intervalo de teste de 32 h. Esse mesmo problema foi resolvido em 109 segundos com a estrutura indexada. Esses resultados mostram que o uso das listas encadeadas em conjunto com o método de fatoração **QR**, não é uma alternativa interessante para esta classe de problemas.

No geral, a estrutura de lista encadeada foi melhor que a estrutura indexada com a decomposição **LU** para os problemas tridiagonais, como foi mostrado na Figura 5.4.

5.4 Resultados *SuiteSparse Matrix Collection*

As Tabelas 5.9, 5.10 e 5.11 apresentam os resultados computacionais para o método de decomposição **LU** com a estrutura de dados indexada. Posteriormente, as Tabelas 5.12, 5.13 e 5.14 apresentam os resultados com o mesmo método númerico, porém, com a estrutura de dados de lista encadeada sucessivamente para os métodos **QR** e de Cholesky. Por simplificação, onde o tempo obtido foi "$0,000000 \times 10^{0}$", na tabela encontra-se apenas "0*" (zero asterisco). A Tabela 5.9 apresenta os resultados obtidos pelos testes para problemas que variam a ordem de grandeza de 19 a 1000.

Tabela 5.9: Decomposição **LU** utilizando estrutura indexada - problemas do repositório com $19 \leq n \leq 1000$.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
Trefethen_ 20b	19	83	22,99	77,01	0*	0*	0*	0*	0*	$1,33 \times 10^{-15}$
Trefethen_ 20	20	89	22,25	77,75	0*	0*	0*	0*	0*	$1,53 \times 10^{-15}$
bcsstk01	48	224	9,72	90,28	0*	0*	0*	0*	0*	0*
bfw62a	62	450	11,71	88,29	0*	0*	0*	0*	0*	0*
bcsstm02	66	66	1,51	98,48	0*	0*	0*	0*	0*	$1,78 \times 10^{-15}$
Trefethen_ 150	150	1095	4,87	95,13	0*	$1,60 \times 10^{-2}$	0*	0*	$1,60 \times 10^{-2}$	$5,77 \times 10^{-14}$
Trefethen_ 200	200	1545	3,86	96,14	0*	$1,50 \times 10^{-2}$	0*	0*	$1,50 \times 10^{-2}$	$1,59 \times 10^{-13}$
Trefethen_ 300	300	2489	2,76	97,23	0*	$7,80 \times 10^{-2}$	0*	0*	$7,80 \times 10^{-2}$	$6,29 \times 10^{-13}$
bcsstm06	420	420	0,24	99,76	0*	$1,25 \times 10^{-1}$	0*	0*	$1,25 \times 10^{-1}$	$9,14 \times 10^{-13}$
bcsstm07	420	3836	2,17	97,82	$1,60 \times 10^{-2}$	$1,40 \times 10^{-1}$	0*	0*	$1,56 \times 10^{-1}$	$2,22 \times 10^{-12}$
Trefethen_ 500	500	4489	1,79	98,20	$1,60 \times 10^{-2}$	$4,69 \times 10^{-1}$	0*	0*	$4,85 \times 10^{-1}$	$1,32 \times 10^{-12}$
Trefethen_ 700	700	6677	1,36	98,64	$1,60 \times 10^{-2}$	1,33	0*	$1,60 \times 10^{-2}$	1,36	$3,01 \times 10^{-12}$
sherman1	1000	3750	0,37	99,62	$3,10 \times 10^{-2}$	3,41	$1,60 \times 10^{-2}$	0*	3,45	0*
Média	300,38	1939,46	6,58	93,41	$6,08 \times 10^{-3}$	$4,29 \times 10^{-1}$	$1,23 \times 10^{-3}$	$1,23 \times 10^{-3}$	$4,37 \times 10^{-1}$	$6,39 \times 10^{-13}$

Fonte: Elaboração própria.

A norma dos resíduos gerados pelas soluções encontradas é muito boa, varia de $0,000000 \times 10^{0}$ (ou "0*" como mostrado na Tabela 5.9) a $3,01 \times 10^{-12}$, no pior cenário, com o problema *"trefethen_ 700"*.

Vale destacar ainda, que de acordo com os resultados, o tempo total de resolução dos problemas foi fortemente influenciado pelo tempo de decomposição da matriz **A**, nas matrizes **L** e **U**. Já que o tempo de leitura de dados e resolução dos sistemas lineares superior e inferior foram majoritariamente quase nulos. Para o problema *"sherman1"*, por exemplo, tempo de decomposição foi de 3,14 s enquanto o tempo de resolução foi de 3,45 s.

Em problemas com ordem de grandeza 1000 (como *"sherman1"*), o tempo de leitura por exemplo, foi mais de cem vezes menor que o tempo empenhado na decomposição de **A**.

Observa-se que os problemas *"bcsstm06"* e *"bcsstm07"* possuem a mesma ordem de grandeza. No entanto, no problema *"bcsstm06"* a matriz é diagonal, consequentemente, possui um tempo de resolução um pouco menor. A Tabela 5.10 apresenta os resultados para os problemas médios resolvidos com o método de decomposição **LU** utilizando a estrutura indexada.

Tabela 5.10: Decomposição **LU** utilizando estrutura indexada - problemas do repositório com $1080 \leq n \leq 6537$.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
sherman2	1080	23094	1,98	98,02	$3,20 \times 10^{-2}$	5,28	0*	$3,10 \times 10^{-2}$	5,34	$1,24 \times 10^{-5}$
bcsstm09	1083	1083	0,09	99,91	$3,20 \times 10^{-2}$	2,78	0*	0*	2,81	$3,73 \times 10^{-13}$
bcsstm10	1086	11589	0,98	99,02	$3,20 \times 10^{-2}$	3,94	$1,60 \times 10^{-2}$	0*	3,98	$7,77 \times 10^{-11}$
sherman4	1104	3786	0,31	99,69	$3,10 \times 10^{-2}$	4,69	$1,60 \times 10^{-2}$	0*	4,73	$6,33 \times 10^{-12}$
1138_bus	1138	2596	0,20	99,80	$3,10 \times 10^{-2}$	3,39	0*	0*	3,42	0*
rdb1250	1250	7300	0,47	99,53	$3,20 \times 10^{-2}$	$1,06 \times 10^{1}$	0*	$1,60 \times 10^{-2}$	$1,07 \times 10^{1}$	0*
c-19	2327	12072	0,22	99,78	$9,40 \times 10^{-2}$	$3,76 \times 10^{1}$	$7,80 \times 10^{-2}$	$4,70 \times 10^{-2}$	$3,78 \times 10^{1}$	$8,67 \times 10^{-17}$
cavity10	2597	76367	1,13	98,87	$1,56 \times 10^{-1}$	$1,54 \times 10^{2}$	$1,25 \times 10^{-1}$	$4,70 \times 10^{-2}$	$1,55 \times 10^{2}$	$7,63 \times 10^{-25}$
sherman5	3312	20793	0,19	99,81	$2,03 \times 10^{-1}$	$2,95 \times 10^{2}$	$1,41 \times 10^{-1}$	$2,03 \times 10^{-1}$	$2,95 \times 10^{2}$	$7,65 \times 10^{-9}$
bcsstm21	3600	3600	0,03	99,97	$2,03 \times 10^{-1}$	$1,55 \times 10^{2}$	$9,30 \times 10^{-2}$	$1,10 \times 10^{-1}$	$1,55 \times 10^{2}$	$3,26 \times 10^{-12}$
c-26	4307	19422	0,10	99,90	$2,81 \times 10^{-1}$	$3,01 \times 10^{2}$	$2,18 \times 10^{-1}$	$1,41 \times 10^{-1}$	$3,02 \times 10^{2}$	$2,16 \times 10^{-14}$
c-28	4598	17594	0,08	99,92	$3,59 \times 10^{-1}$	$4,13 \times 10^{2}$	$6,30 \times 10^{-2}$	$4,60 \times 10^{-2}$	$4,13 \times 10^{2}$	0*
sherman3	5005	20033	0,08	99,92	$3,59 \times 10^{-1}$	$8,15 \times 10^{2}$	$4,07 \times 10^{-1}$	$5,00 \times 10^{-1}$	$8,16 \times 10^{2}$	$7,20 \times 10^{-10}$
c-29	5033	24382	0,10	99,90	$4,22 \times 10^{-1}$	$5,76 \times 10^{2}$	$7,80 \times 10^{-2}$	$7,80 \times 10^{-2}$	$5,76 \times 10^{2}$	0*
c-30	5321	35507	0,12	99,87	$4,06 \times 10^{-1}$	$5,97 \times 10^{2}$	$6,30 \times 10^{-2}$	$7,80 \times 10^{-2}$	$5,98 \times 10^{2}$	0*
c-31	5339	41955	0,15	99,85	$4,07 \times 10^{-1}$	$6,06 \times 10^{2}$	$6,20 \times 10^{-2}$	$6,30 \times 10^{-2}$	$6,06 \times 10^{2}$	0*
c-32	5975	30223	0,08	99,91	$5,00 \times 10^{-1}$	$8,62 \times 10^{2}$	$9,40 \times 10^{-2}$	$9,40 \times 10^{-2}$	$8,62 \times 10^{2}$	0*
c-33	6317	31220	0,08	99,92	$5,78 \times 10^{-1}$	$1,03 \times 10^{3}$	$1,10 \times 10^{-1}$	$7,80 \times 10^{-2}$	$1,03 \times 10^{3}$	0*
c-35	6537	34714	0,08	99,92	$6,10 \times 10^{-1}$	$1,26 \times 10^{3}$	$1,25 \times 10^{-1}$	$1,25 \times 10^{-1}$	$1,26 \times 10^{3}$	0*
Média	3526,78	21964,74	0,34	99,66	$2,51 \times 10^{-1}$	$3,75 \times 10^{2}$	$8,89 \times 10^{-2}$	$8,72 \times 10^{-2}$	$3,75 \times 10^{2}$	$6,53 \times 10^{-7}$

Fonte: Elaboração própria.

Nota-se um aumento gradual do tempo de leitura do aquivo de entrada de dados e também de resolução dos sistemas lineares gerados se comparados aos problemas anteriores. O tempo de decomposição segue sendo o que mais influencia o tempo total de resolução dos problemas, pois é aproximadamente 100% (ou 99,94%) do tempo total. Este passou de poucos segundos para alguns minutos. A norma dos resíduos gerados durante os testes manteve-se entre quase zero e $1,24 \times 10^{-5}$, considerado um excelente resultado.

O problema *"sherman2"*, cuja norma do resíduo foi de $1,24 \times 10^{-5}$, obteve a pior norma de resíduo dentre os problemas da Tabela 5.10. Este não era o problema com maior ordem de grandeza e nem maior número de elementos não nulos; no entanto, outras características próprias do problema podem ter influenciado nesse resultado (como alto número de condição, por exemplo).

A maioria dos problemas da classe *"c-"* obtiveram norma do resíduo quase nula, o que evidencia a qualidade (acurácia) da solução encontrada, principalmente para os problemas com ordem superior a 5000.

No problema *"bcsstm21"*, sua matriz de coeficientes é diagonal e, provavelmente, por esta característica seu tempo total foi quase metade do *"sherman5"*, mesmo tendo ordem de grandeza maior. A seguir, tem-se a Tabela 5.11, que mostra os resultados dos testes de decomposição **LU** utilizando estrutura indexada para os maiores problemas.

A sigla "EM" significa estouro de memória, indicando que o resultado de alguma operação excedeu a capacidade de representação para esse tipo de dado. Nota-se que embora os problemas *"c-34"* e *"c-36"* tenham sido solucionados em, aproximadamente, meia hora com norma de resíduo nula para o número de casas decimais adotado, os demais problemas tiveram estouro de memória e não foram resolvidos. Logo, os problemas (aqui testados) com ordem de grandeza superior a 7479, não puderam ser solucionados utilizando fatoração **LU**, com a estrutura indexada. Pois, como dito, esta estrutura armazena todos os elementos nulos e estes são utilizados em cálculos desnecessários.

Tabela 5.11: Decomposição **LU** utilizando estrutura indexada - problemas do repositório com $6611 \leq n \leq 21132$.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
c-34	6611	35472	0,08	99,92	$6,25 \times 10^{-1}$	$1,21 \times 10^{3}$	$9,40 \times 10^{-2}$	$9,30 \times 10^{-2}$	$1,21 \times 10^{3}$	0*
c-36	7479	36710	0,06	99,93	2,83	$1,88 \times 10^{3}$	$1,56 \times 10^{-1}$	$1,56 \times 10^{-1}$	$1,88 \times 10^{3}$	0*
c-38	8127	42908	0,06	99,93	EM	EM	EM	EM	EM	EM
c-37	8204	41440	0,06	99,94	EM	EM	EM	EM	EM	EM
c-39	9271	73041	0,08	99,91	EM	EM	EM	EM	EM	EM
c-41	9769	55757	0,06	99,94	EM	EM	EM	EM	EM	EM
c-40	9941	45721	0,05	99,95	EM	EM	EM	EM	EM	EM
c-42	10471	60378	0,05	99,94	EM	EM	EM	EM	EM	EM
c-44	10728	47864	0,04	99,96	EM	EM	EM	EM	EM	EM
c-43	11125	67400	0,05	99,94	EM	EM	EM	EM	EM	EM
c-45	13206	93829	0,05	99,95	EM	EM	EM	EM	EM	EM
pres_poison	14822	356313	0,17	99,83	EM	EM	EM	EM	EM	EM
c-46	14913	72655	0,03	99,97	EM	EM	EM	EM	EM	EM
c-47	15343	113372	0,05	99,95	EM	EM	EM	EM	EM	EM
bcsstm25	15439	15439	0,01	99,99	EM	EM	EM	EM	EM	EM
c-48	18354	92217	0,03	99,97	EM	EM	EM	EM	EM	EM
c-49	21132	89087	0,02	99,98	EM	EM	EM	EM	EM	EM
Média	12055	78800,18	$5,59 \times 10^{-2}$	99,94	1,73	1545	0,12	0,12	1545	0*

Fonte: Elaboração própria.

Como relatado anteriormente, outros fatores além da ordem de grandeza, podem influenciar nos resultados. Logo, não seria correto generalizar que todas as matrizes com ordem de grandeza superior a 7479, não podem ser resolvidas pela decomposição **LU** e estrutura indexada. Já que, matrizes diagonais por exemplo, costumam apresentar melhores resultados (em termos de tempo computacional) se comparadas a matrizes mais densas. No entanto, dentro do conjunto de problemas escolhido, essa foi a ordem de grandeza limite para o método de decomposição **LU** com estrutura indexada.

A Tabela 5.12 apresentada a seguir, traz os resultados para os problemas pequenos resolvidos com método de decomposição **LU** utilizando a estrutura de listas encadeadas.

Tabela 5.12: Decomposição **LU** utilizando estrutura de listas encadeadas - problemas do repositório com $19 \leq n \leq 1000$.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
Trefethen_20b	19	83	22,99	77,01	0*	0*	0*	0*	0*	$1,33 \times 10^{-15}$
Trefethen_20	20	89	22,25	77,75	0*	0*	0*	0*	0*	$1,54 \times 10^{-15}$
bcsstk01	48	224	9,72	90,28	0*	0*	0*	0*	0*	0*
bfw62a	62	450	11,71	88,29	0*	0*	0*	0*	0*	0*
bcsstm02	66	66	1,51	98,48	0*	0*	0*	0*	0*	$1,78 \times 10^{-15}$
Trefethen_150	150	1095	4,87	95,13	0*	$1,6 \times 10^{-2}$	0*	0*	$1,6 \times 10^{-2}$	$5,78 \times 10^{-14}$
Trefethen_200	200	1545	3,86	96,14	0*	$3,1 \times 10^{-2}$	0*	0*	$3,1 \times 10^{-2}$	$1,59 \times 10^{-13}$
Trefethen_300	300	2489	2,76	97,23	0*	$1,25 \times 10^{-1}$	0*	0*	$1,25 \times 10^{-1}$	$6,29 \times 10^{-13}$
bcsstm06	420	420	0,24	99,76	0*	$7,8 \times 10^{-2}$	0*	0*	$7,8 \times 10^{-2}$	$9,14 \times 10^{-13}$
bcsstm07	420	3836	2,17	97,82	0*	$4,06 \times 10^{-1}$	0*	0*	$4,06 \times 10^{-1}$	$2,22 \times 10^{-12}$
Trefethen_500	500	4489	1,79	98,20	$1,5 \times 10^{-2}$	$6,1 \times 10^{-1}$	0*	0*	$6,25 \times 10^{-1}$	$1,32 \times 10^{-12}$
Trefethen_700	700	6677	1,36	98,64	0*	1,78	0*	0*	1,78	$3,01 \times 10^{-12}$
sherman1	1000	3750	0,37	99,62	0*	$1,5 \times 10^{1}$	0*	0*	$1,5 \times 10^{1}$	0*
Média	300,38	1939,46	6,58	93,41	$1,15 \times 10^{-3}$	1,39	0*	0*	1,39	$6,39 \times 10^{-13}$

Fonte: Elaboração própria.

No geral, a Tabela 5.12 apresenta um crescimento no tempo total para resolução dos problemas de acordo com a ordem de grandeza, sendo o maior tempo de 15 segundos.

O tempo de leitura foi quase nulo na maioria dos problemas. Os tempos relacionados a resolução dos sistemas das matrizes decompostas também foi aproximadamente nulo em todos os testes. Novamente, observa-se o tempo total sendo influenciado, principalmente, pelo tempo de decomposição da matriz.

Para todos os problemas o resíduo foi da ordem de 10^{-12} ou menor, o que retrata a boa qualidade das soluções encontradas. Observa-se que os problemas onde há matriz diagonal mostraram melhores resultados se comparados a outros problemas com ordem de grandeza semelhante ou muito próxima, é o caso de *"bcsstm02"* e *"bcsstm06"*.

Analisando a Tabela 5.9 que mostra os resultados da decomposição **LU** com estrutura de dados indexada, nota-se um leve aumento no tempo máximo de solução dos problemas. Já que com a estrutura indexada o tempo máximo gasto foi de 3,45 segundos e com a estrutura de listas encadeadas o tempo máximo foi de 15 segundos ambos para o problema *"sherman1"*. Neste caso, a lista encadeada não proporciona uma redução no tempo computacional, pois os problemas são pequenos, com ordem de grandeza na casa de centenas.

Quanto à norma do resíduo gerado nas soluções não há diferença significativa entre os resultados das Tabelas 5.9 e 5.12, pois ambas possuem a mesma ordem de grandeza com ínfimas diferenças. A seguir, a Tabela 5.13 mostra os resultados para os problemas médios do repositório *online* usando o método de fatoração **LU** e estrutura de dados de listas encadeadas.

Tabela 5.13: Decomposição **LU** utilizando estrutura de listas encadeadas - problemas do repositório com $1080 \leq n \leq 6537$.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
sherman2	1080	23094	1,98	98,02	$3,1 \times 10^{-2}$	$1,20 \times 10^{2}$	0*	0*	$1,20 \times 10^{2}$	$1,25 \times 10^{-5}$
bcsstm09	1083	1083	0,09	99,91	0*	1,5	0*	0*	1,5	$3,73 \times 10^{-13}$
bcsstm10	1086	11589	0,98	99,02	0*	7,25	0*	0*	7,25	$7,76 \times 10^{-11}$
sherman4	1104	3786	0,31	99,69	0*	$4,64 \times 10^{1}$	0*	0*	$4,64 \times 10^{1}$	$6,32 \times 10^{-12}$
1138_bus	1138	2596	0,20	99,80	0*	1,78	0*	0*	1,78	0*
rdb1250	1250	7300	0,47	99,53	0*	$5,33 \times 10^{1}$	0*	$1,5 \times 10^{-2}$	$5,33 \times 10^{1}$	0*
c-19	2327	12072	0,22	99,78	$1,6 \times 10^{-2}$	$3,99 \times 10^{1}$	$1,6 \times 10^{-2}$	0*	4×10^{1}	$8,67 \times 10^{-17}$
cavity10	2597	76367	1,13	98,87	$7,8 \times 10^{-2}$	$2,29 \times 10^{3}$	$1,6 \times 10^{-2}$	0*	$2,29 \times 10^{3}$	$7,62 \times 10^{-25}$
sherman5	3312	20793	0,19	99,81	$3,1 \times 10^{-2}$	$3,74 \times 10^{3}$	$1,5 \times 10^{-2}$	0*	$3,74 \times 10^{3}$	$7,65 \times 10^{-9}$
bcsstm21	3600	3600	0,03	99,97	0*	$5,56 \times 10^{1}$	$1,5 \times 10^{-2}$	0*	$5,56 \times 10^{1}$	$3,26 \times 10^{-12}$
c-26	4307	19422	0,10	99,90	$1,5 \times 10^{-2}$	$1,45 \times 10^{2}$	$3,1 \times 10^{-2}$	0*	$1,45 \times 10^{2}$	$2,15 \times 10^{-14}$
c-28	4598	17594	0,08	99,92	$1,50 \times 10^{-2}$	$3,40 \times 10^{2}$	3,20	0*	$3,40 \times 10^{2}$	0*
sherman3	5005	20033	0,08	99,92	$3,10 \times 10^{-2}$	$5,22 \times 10^{3}$	$3,10 \times 10^{-2}$	0*	$5,22 \times 10^{3}$	$7,20 \times 10^{-10}$
c-29	5033	24382	0,10	99,90	$3,1 \times 10^{-2}$	$4,41 \times 10^{2}$	$3,2 \times 10^{-2}$	0*	$4,42 \times 10^{2}$	0*
c-30	5321	35507	0,12	99,87	$6,30 \times 10^{-2}$	$5,41 \times 10^{2}$	$3,20 \times 10^{-2}$	0*	$5,41 \times 10^{2}$	0*
c-31	5339	41955	0,15	99,85	$4,70 \times 10^{-2}$	$7,04 \times 10^{2}$	$3,20 \times 10^{-2}$	0*	$7,04 \times 10^{2}$	0*
c-32	5975	30223	0,08	99,91	$4,70 \times 10^{-2}$	$6,78 \times 10^{2}$	$4,70 \times 10^{-2}$	0*	$6,78 \times 10^{2}$	0*
c-33	6317	31220	0,08	99,92	$3,10 \times 10^{-2}$	$9,70 \times 10^{2}$	$4,70 \times 10^{-2}$	0*	$9,70 \times 10^{2}$	0*
c-35	6537	34714	0,08	99,92	$3,10 \times 10^{-2}$	$9,73 \times 10^{2}$	$4,70 \times 10^{-2}$	0*	$9,73 \times 10^{2}$	0*
Média	3526,79	21964,74	0,34	99,66	$2,46 \times 10^{-2}$	$8,61 \times 10^{2}$	$1,87 \times 10^{-1}$	$7,89 \times 10^{-4}$	$8,61 \times 10^{2}$	$6,58 \times 10^{-7}$

Fonte: Elaboração própria.

Nota-se que o tempo de decomposição segue maior em relação aos demais tempos, quase equivalente a 100% do tempo de resolução.

Da Tabela 5.13 nota-se que os problemas apresentados foram resolvidos em um tempo total que varia entre 1,5 segundos a, aproximadamente, 1 hora e 27 minutos. A norma do resíduo gerado teve um valor máximo de $1,25 \times 10^{-5}$. De uma forma geral, o tempo total cresceu de acordo com a ordem de grandeza e o número de elementos não nulos, com

algumas exceções como *"c-19"* e *"c-29"*.

O problema *"sherman2"* possui a menor dimensão da Tabela 5.13 e não possui o menor tempo de solução, provavelmente devido ao seu número de elementos não nulos ou outras características do problema, como alto número de condição. Ressalta-se ainda que ele teve o pior resíduo dentre os problemas observados na Tabela 5.13.

O problema *"1138_ bus"*, apresentou um tempo de resolução menor que o *"sherman4"*, ambos com a mesma magnitude de ordem de grandeza. Contudo, o número de elementos não nulos de *"1138_ bus"* é bem menor que em *"sherman4"*, o que justifica tal desempenho. Por seu turno, os problemas *"c-19"* e *"c-29"* obtiveram tempos de resolução menor que os problemas que os antecedem na Tabela 5.13 (*"rdb1250"* e *"sherman3"*), mesmo possuindo maior ordem de grandeza e maior número de elementos não nulos.

O problema *"bcsstm21"* cuja matriz é diagonal foi resolvido em pouco tempo se comparado a outros problemas com ordem de grandeza próxima. Esse comportameno pode ser justificado pelo baixo número de elementos não nulos.

Ao comparar os tempos das Tabelas 5.13 (decomposição **LU** com estrutura de listas encadeadas) e 5.10 (decomposição **LU** com estrutura indexada) nota-se um aumento do tempo total máximo de execução dos problemas mantendo a norma do resíduo constante. A média do tempo de resolução dos problemas cresceu 120%.

Vale destacar que *"cavity10"*, *"sherman5"* e *"sherman3"* apresentaram tempos expressivos comparativamente (inclusive se considerado os tempos de resolução desses problemas com a estrutura indexada). A seguir, estão apresentados na Tabela 5.14 os resultados da decomposição **LU** utilizando lista encadeada para os maiores problemas.

Tabela 5.14: Decomposição **LU** utilizando estrutura de listas encadeadas - problemas do repositório com $6611 \leq n \leq 21132$.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
c-34	6611	35472	0,08	99,92	$4,70 \times 10^{-2}$	$1,05 \times 10^{3}$	$6,30 \times 10^{-2}$	0*	$1,05 \times 10^{3}$	0*
c-36	7479	36710	0,06	99,93	$4,70 \times 10^{-2}$	$1,56 \times 10^{3}$	$6,30 \times 10^{-2}$	0*	$1,56 \times 10^{3}$	0*
c-38	8127	42908	0,06	99,93	$4,70 \times 10^{-2}$	$1,84 \times 10^{3}$	$6,20 \times 10^{-2}$	0*	$1,84 \times 10^{3}$	$2,00 \times 10^{-2}$
c-37	8204	41440	0,06	99,94	$4,60 \times 10^{-2}$	$2,49 \times 10^{3}$	$7,80 \times 10^{-2}$	$1,60 \times 10^{-2}$	$2,49 \times 10^{3}$	$3,12 \times 10^{-3}$
c-39	9271	73041	0,08	99,91	$7,80 \times 10^{-2}$	$3,52 \times 10^{3}$	$1,25 \times 10^{-1}$	0*	$3,52 \times 10^{3}$	$4,63 \times 10^{-3}$
c-41	9769	55757	0,06	99,94	$4,70 \times 10^{-2}$	$4,12 \times 10^{3}$	$1,09 \times 10^{-1}$	0*	$4,12 \times 10^{3}$	$3,10 \times 10^{-2}$
c-40	9941	45721	0,05	99,95	6.30×10^{-2}	$3,23 \times 10^{3}$	$1,41 \times 10^{-1}$	0*	$3,23 \times 10^{3}$	$4,21 \times 10^{-2}$
c-42	10471	60378	0,05	99,94	$7,80 \times 10^{-2}$	$4,49 \times 10^{3}$	$1,25 \times 10^{-1}$	0*	$4,49 \times 10^{3}$	$7,16 \times 10^{-2}$
c-44	10728	47864	0,04	99,96	$4,70 \times 10^{-2}$	$4,07 \times 10^{3}$	$1,25 \times 10^{-1}$	0*	$4,08 \times 10^{3}$	$7,72 \times 10^{-3}$
c-43	11125	67400	0,05	99,94	$1,09 \times 10^{-1}$	$5,33 \times 10^{3}$	$1,40 \times 10^{-1}$	0*	$5,33 \times 10^{3}$	$3,81 \times 10^{-2}$
c-45	13206	93829	0,05	99,95	$1,57 \times 10^{-1}$	$9,58 \times 10^{3}$	$2,03 \times 10^{-1}$	0*	$9,58 \times 10^{3}$	$2,05 \times 10^{-1}$
pres_poisson	14822	365313	0,17	99,83	$5,00 \times 10^{-1}$	$6,58 \times 10^{3}$	$2,97 \times 10^{-1}$	0*	$6,58 \times 10^{3}$	0*
c-46	14913	72655	0,03	99,97	$9,30 \times 10^{-2}$	$1,13 \times 10^{4}$	$2,50 \times 10^{-1}$	0*	$1,13 \times 10^{4}$	$2,89 \times 10^{-2}$
c-47	15343	113372	0,05	99,95	$1,25 \times 10^{-1}$	$1,39 \times 10^{4}$	$2,66 \times 10^{-1}$	0*	$1,39 \times 10^{4}$	$2,05 \times 10^{-2}$
bcsstm25	15439	15439	0,01	99,99	$3,10 \times 10^{-2}$	$4,86 \times 10^{3}$	$2,97 \times 10^{-1}$	0*	$4,86 \times 10^{3}$	$2,98 \times 10^{-8}$
c-48	18354	92217	0,03	99,97	$7,80 \times 10^{-2}$	$2,28 \times 10^{4}$	$4,69 \times 10^{-1}$	0*	$2,28 \times 10^{4}$	$2,75 \times 10^{-2}$
c-49	21132	89087	0,02	99,98	$9,40 \times 10^{-2}$	$3,48 \times 10^{4}$	$6,09 \times 10^{-1}$	0*	$3,48 \times 10^{4}$	$1,14 \times 10^{-2}$
Média	12055	79329,59	0,05	99,94	$9,92 \times 10^{-2}$	$7,97 \times 10^{3}$	$2,01 \times 10^{-1}$	$9,41 \times 10^{-4}$	$7,97 \times 10^{3}$	$3,01 \times 10^{-2}$

Fonte: Elaboração própria.

Nota-se que o tempo de resolução do sistema linear superior foi quase nulo em quase todos os problemas, exceto em *"c-37"*.

O tempo total de solução dos problemas é fortemente influenciado pelo tempo de decomposição da matriz **A**, próximo a 100%. Todos os problemas foram resolvidos com um tempo entre (pouco mais de) 17 minutos e 9 horas e quarenta minutos (com *"c-49"*).

O tempo médio aumentou, pois a dimensão e número de elementos não nulos cresceu.

Assim, observa-se o expressivo crescimento do tempo total de resolução dos problemas em ordens de grandeza superiores. No entanto, vale destacar que antes com estrutura indexada muitos desses problemas não foram solucionados (ver Tabela 5.11).

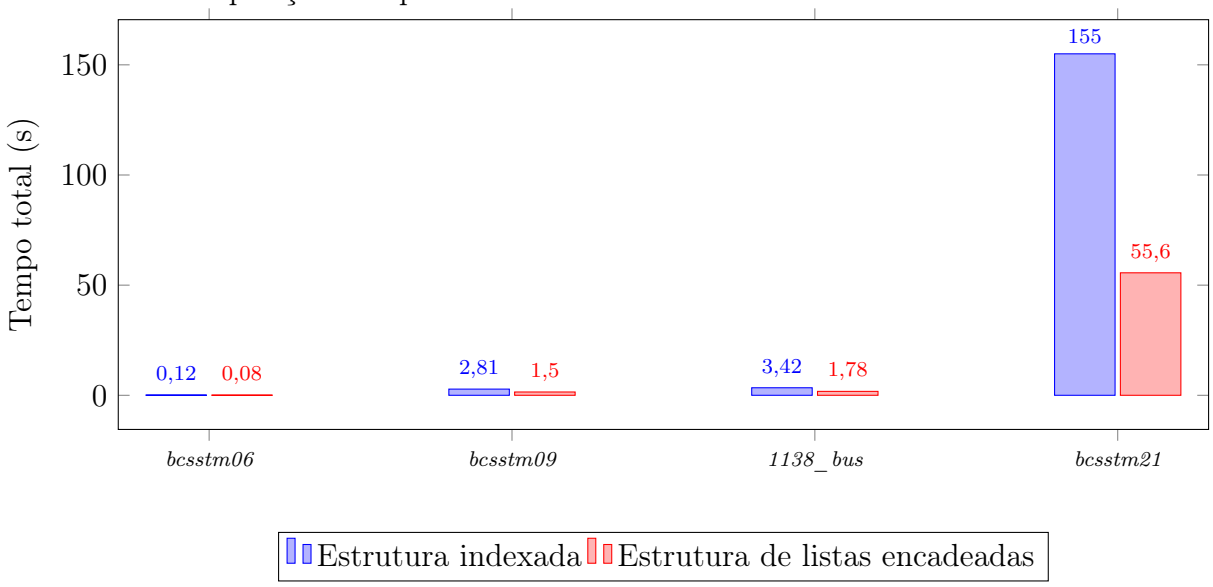
Nesse aspecto, algumas exceções merecem atenção: como *"c-40"* e *"c-44"*. Esses são problemas que foram resolvidos em tempos menores que seu antecessor da tabela (*"c-41"* e *"c-42"*). Nesses casos isso ocorreu provavelmente devido ao número de elementos não nulos que é inferior ao número de elementos não nulos do problema anterior.

Outro resultado relevante é o do problema *"pres_poisson"*. Ele também é resolvido em um tempo cerca de 30% menor que o problema anterior (*"c-45"*). No entanto, nesse caso além de ter uma ordem de grandeza maior, ele também apresenta um número de elementos não nulos superior três vezes o anterior. Tudo isso sugere a influência de outras características próprias do problema, como número de condição.

Na Tabela 5.14, a norma do resíduo máxima foi de $2,05 \times 10^{-1}$, um valor bem superior aos apresentados anteriormente. No entanto, vale lembrar que do problema *"c-38"* adiante a estrutura indexada não foi capaz de fornecer solução alguma, devido aos estouros de memória (*"overflow"*).

A seguir, a Figura 5.10 compara o tempo de solução de alguns problemas (que tiveram resíduos baixos) como *"bcsstm06"*, *"bcsstm09"*, *"1138_bus"* e *"bcsstm21"* para a fatoração **LU** com as estruturas de dados indexada e listas encadeadas. Ressalta-se que todos os problemas da Figura 5.10 possuem densidade menor ou igual 0,24% e espasidade igual ou superior a 99,76%.

Figura 5.10: Comparação do tempo total de resolução de problemas do repositório *online* utilizando decomposição **LU** para estrutura indexada e de listas encadeadas.



Fonte: Elaboração própria.

A Figura 5.10 evidencia a vantagem de usar a estrutura de dados de listas encadeadas diante da redução do tempo total para solução dos problemas, mantendo a norma do resíduo similar. Vale lembrar também que dentro do conjunto de problemas retirados do respositório *online* que foram utilizados para testes, a estrutura indexada resolveu problemas com ordem máxima de 8127, enquando a estrutura de listas resolveu problemas de dimensão 21132.

A seguir, são apresentados os resultados para os problemas tratados, solucionados com o método de fatoração **QR** usando primeiro a estrutura de dados indexada (nas Tabelas 5.15 e 5.16) e, posteriormente, a estrutura de listas encadeadas (Tabelas 5.17, 5.18 e 5.19).

A Tabela 5.15 mostra os resultados para os problemas pequenos retirados do repositório *online* resolvidos com fatoração **QR** e estrutura de listas encadeadas.

Tabela 5.15: Decomposição **QR** utilizando estrutura indexada - problemas do repositório com $19 \leq n \leq 1000$.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
Trefethen_20b	19	83	22,99	77,01	0*	0*	0*	0*	0*	1.33×10^{-15}
Trefethen_20	20	89	22,25	77,75	0*	0*	0*	0*	0*	1.54×10^{-15}
bcsstk01	48	224	9,72	90,28	0*	0*	0*	0*	0*	0*
bfw62a	62	450	11,71	88,29	0*	0*	0*	0*	0*	0*
bcsstm02	66	66	1,51	98,48	0*	$1,50 \times 10^{-2}$	0*	0*	$1,50 \times 10^{-2}$	$1,78 \times 10^{-15}$
Trefethen_150	150	1095	4,87	95,13	0*	$6,30 \times 10^{-2}$	$1,50 \times 10^{-2}$	0*	$7,80 \times 10^{-2}$	$5,77 \times 10^{-14}$
Trefethen_200	200	1545	3,86	96,14	0*	$1,88 \times 10^{-1}$	0*	0*	$1,88 \times 10^{-1}$	$1,59 \times 10^{-13}$
Trefethen_300	300	2489	2,76	97,23	0*	$6,25 \times 10^{-1}$	0*	0*	$6,25 \times 10^{-1}$	$6,29 \times 10^{-13}$
bcsstm06	420	420	0,24	99,76	$1,50 \times 10^{-2}$	$3,13 \times 10^{-1}$	0*	0*	$3,28 \times 10^{-1}$	$9,14 \times 10^{-14}$
bcsstm07	420	3836	2,17	97,82	0*	1,05	$1,60 \times 10^{-2}$	0*	1,06	$2,22 \times 10^{-12}$
Trefethen_500	500	8478	1,79	98,20	$1,50 \times 10^{-2}$	2,75	$1,60 \times 10^{-2}$	0*	2,78	$1,32 \times 10^{-12}$
Trefethen_700	700	6677	1,36	98,64	$1,60 \times 10^{-2}$	$1,00 \times 10^{1}$	$1,6 \times 10^{-2}$	0*	$1,01 \times 10^{1}$	$3,01 \times 10^{-12}$
sherman1	1000	3750	0,37	99,62	$3,20 \times 10^{-2}$	$3,12 \times 10^{1}$	$1,50 \times 10^{-2}$	0*	$3,13 \times 10^{1}$	0*
Média	300,38	2246,31	6,58	93,41	$6,00 \times 10^{-3}$	3,55	$6,00 \times 10^{-3}$	0*	3,57	$5,76 \times 10^{-13}$

Fonte: Elaboração própria.

Nota-se que todos os problemas da Tabela 5.15 apresentam tempo de leitura e de resolução do sistema linear 2 (matriz triangular superior) e do sistema linear 1 (cuja matriz **Q** é ortogonal) quase nulos. Com isso, assim como observado nas tabelas anteriores, o tempo total de resolução segue fortemente influenciado pelo tempo de decomposição da matriz **A**, em **Q** e **R**. Nesse caso, o tempo de decomposição correspode a cerca de 99,44% do tempo de resolução. O tempo total de resolução dos problemas cresce de acordo com a ordem de grandeza dos problemas estudados (salvo algumas exceções) sendo o tempo total máximo de 31,3 segundos para o problema *"sherman1"*.

O problema *"bcsstm06"* de ordem 420 é uma matriz diagonal, por isso apresenta tempo menor que seu antecessor de ordem 300 (*"Trefethen_300"*). Considera-se que os resultados apresentam boa qualidade da solução, pois a norma do resíduo foi bem pequena, com valor máximo de $3,01 \times 10^{-12}$. A seguir encontra-se a Tabela 5.16 com os resultados dos problemas para os problemas médios do repositório *online* resolvidos com fatoração **QR** e estrutura indexada.

Tabela 5.16: Decomposição **QR** utilizando estrutura indexada - problemas do repositório com $1080 \leq n \leq 6537$.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
sherman2	1080	23094	1,98	98,02	$4,70 \times 10^{-2}$	$5,77 \times 10^{1}$	$1,50 \times 10^{-2}$	0*	$5,78 \times 10^{1}$	$1,34 \times 10^{-1}$
bcsstm09	1083	1083	0,09	99,91	$1,50 \times 10^{-2}$	$1,07 \times 10^{1}$	$1,60 \times 10^{-2}$	$1,50 \times 10^{-2}$	$1,08 \times 10^{1}$	$3,73 \times 10^{-13}$
bcsstm10	1086	22092	0,98	99,02	$3,20 \times 10^{-2}$	$4,71 \times 10^{-1}$	$1,40 \times 10^{-1}$	$1,60 \times 10^{-2}$	$4,73 \times 10^{1}$	$7,77 \times 10^{-11}$
sherman4	1104	3786	0,31	99,69	$3,10 \times 10^{-2}$	$3,18 \times 10^{1}$	$3,10 \times 10^{-2}$	$3,10 \times 10^{-2}$	$3,18 \times 10^{1}$	$1,84 \times 10^{-11}$
1138_bus	1138	2596	0,20	99,80	$3,20 \times 10^{-2}$	$3,17 \times 10^{1}$	$1,50 \times 10^{-2}$	0*	$3,18 \times 10^{1}$	0*
rdb1250	1250	7300	0,47	99,53	$3,20 \times 10^{-2}$	$1,48 \times 10^{2}$	$1,50 \times 10^{-2}$	$1,60 \times 10^{-2}$	$1,48 \times 10^{2}$	0*
c-19	2327	12072	0,22	99,78	$1,25 \times 10^{-1}$	$1,89 \times 10^{2}$	$2,19 \times 10^{-1}$	$1,25 \times 10^{-1}$	$1,89 \times 10^{2}$	$8,67 \times 10^{-17}$
cavity10	2597	76367	1,13	98,87	$1,88 \times 10^{-1}$	$2,64 \times 10^{3}$	0*	$1,25 \times 10^{-1}$	$2,64 \times 10^{3}$	$5,98 \times 10^{-23}$
sherman5	3312	20793	0,19	99,81	$2,34 \times 10^{-}$	$2,07 \times 10^{3}$	0*	$2,97 \times 10^{-1}$	$2,07 \times 10^{3}$	$1,61 \times 10^{-8}$
bcsstm21	3600	3600	0,03	99,97	$2,34 \times 10^{-1}$	$8,79 \times 10^{2}$	$2,65 \times 10^{-1}$	$1,88 \times 10^{-1}$	$8,79 \times 10^{2}$	$3,26 \times 10^{-12}$
c-26	4307	19422	0,10	99,90	EM	EM	EM	EM	EM	EM
c-28	4598	17594	0,08	99,92	EM	EM	EM	EM	EM	EM
sherman3	5005	20033	0,08	99,92	EM	EM	EM	EM	EM	EM
c-29	5033	24382	0,10	99,90	EM	EM	EM	EM	EM	EM
c-30	5321	35507	0,12	99,87	EM	EM	EM	EM	EM	EM
c-31	5339	41955	0,15	99,85	EM	EM	EM	EM	EM	EM
c-32	5975	30223	0,08	99,91	EM	EM	EM	EM	EM	EM
c-33	6317	31220	0,08	99,92	EM	EM	EM	EM	EM	EM
c-35	6537	34714	0,08	99,92	EM	EM	EM	EM	EM	EM
Média	3526,79	22517,53	0,34	99,66	$9,70 \times 10^{-2}$	$6,05 \times 10^{2}$	$7,16 \times 10^{-2}$	$8,13 \times 10^{-2}$	$6,10 \times 10^{2}$	$1,34 \times 10^{-2}$

Fonte: Elaboração própria.

A exceção de *"sherman2"*, as demais normas do resíduo foram bem pequenas. Obtiveram inclusive o mesmo valor que na decomposição **LU** com estrutura indexada e listas encadeadas. Os problemas *"bcsstm09"* e *"bcsstm21"* são matrizes diagonais e por isso apresentam tempos menores que seus antecessores; ou seja, são problemas com menos elementos não nulos. Como possuem menos elementos, eles consequentemente são resolvidos em menos tempo.

O tempo total de resolução dos problemas cresceu de acordo com a ordem de grandeza, a exceção foram os problemas *"bcsstm09"*, *"sherman5"* e *"bcsstm21"*. *"Sherman5"* apresentou um tempo de resolução total inferior a *"cavity10"* (seu antecessor). Isso se deve possivelmente ao fato de *"cavity10"* ter um maior número de elementos não nulos, necessitando efetuar mais operações e, consequentemente, levando mais tempo para ser solucionado.

De *"c-26"* adiante na Tabela 5.16, incluindo inclusive o restante dos problemas com dimensão entre 6611 e 21132, todos apresentaram "estouro" de memória indicado mais uma vez por "EM" (*overflow*) e não puderam ser solucionados através da estrutura indexada. O método de fatoração QR usando estrutura indexada não resolveu (dentre os problemas estudados) problemas com ordem de grandeza superior a 3600.

Vale lembrar que usando o método de fatoração **LU** e estrutura de dados indexada foram resolvidos problemas de ordem de grandeza até 7479 (ou até o problema *"c-36"*). A Tabela 5.17 mostra os resultados dos problemas do repositório para o método de fatoração **QR** com a estrutura de listas encadeadas.

Tabela 5.17: Decomposição **QR** utilizando estrutura de listas encadeadas - problemas do repositório com $19 \leq n \leq 1000$.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
Trefethen_20b	19	83	22,99	77,01	0*	0*	0*	0*	0*	$1,33 \times 10^{-15}$
Trefethen_20	20	89	22,25	77,75	$1,60 \times 10^{-2}$	$1,60 \times 10^{-2}$	0*	$1,60 \times 10^{-2}$	$4,80 \times 10^{-2}$	$1,54 \times 10^{-15}$
bcsstk01	48	224	9,72	90,28	0*	0*	0*	0*	0*	0*
bfw62a	62	450	11,71	88,29	0*	$3,20 \times 10^{-2}$	0*	$3,20 \times 10^{-2}$	$6,40 \times 10^{-2}$	0*
bcsstm02	66	66	1,51	98,48	0*	0*	0*	0*	0*	$1,78 \times 10^{-15}$
Trefethen_150	150	1095	4,87	95,13	0*	$6,30 \times 10^{-2}$	0*	$6,30 \times 10^{-2}$	$1,26 \times 10^{-1}$	$5,77 \times 10^{-14}$
Trefethen_200	200	1545	3,86	96,14	0*	$1,40 \times 10^{-1}$	0*	$1,40 \times 10^{-1}$	$2,80 \times 10^{-1}$	$1,59 \times 10^{-13}$
Trefethen_300	300	2489	2,76	97,23	0*	$4,84 \times 10^{-1}$	0*	$4,84 \times 10^{-1}$	$9,68 \times 10^{-1}$	$6,29 \times 10^{-13}$
bcsstm06	420	420	0,24	99,76	$1,60 \times 10^{-2}$	1,08	0*	1,08	2,17	$9,14 \times 10^{-13}$
bcsstm07	420	3836	2,17	97,82	0*	1,89	0*	1,89	3,78	$2,22 \times 10^{-12}$
Trefethen_500	500	4489	1,79	98,20	0*	2,45	0*	2,47	4,92	$1,32 \times 10^{-12}$
Trefethen_700	700	6677	1,36	98,64	0*	8,47	0*	8,47	$1,69 \times 10^{1}$	$3,01 \times 10^{-12}$
sherman1	1000	3750	0,37	99,62	0*	$2,01 \times 10^{3}$	0*	$2,02 \times 10^{3}$	$4,03 \times 10^{3}$	0*
Média	300,38	1939,46	6,58	93,41	$2,46 \times 10^{-3}$	$1,55 \times 10^{2}$	0*	$1,56 \times 10^{2}$	$3,12 \times 10^{2}$	$6,39 \times 10^{-13}$

Fonte: Elaboração própria.

O tempo total de resolução dos problemas cresceu de acordo com a ordem de grandeza, exceto em algumas situações como em: *"bcsstk01"* e *"bcsstm02"*. Comparando com a Tabela 5.15, nota-se um aumento de quase cem vezes na média dos tempos para resolução dos problemas mantendo a ordem de grandeza da norma dos resíduos encontrados. A Tabela 5.18 apresenta os resultados da fatoração **QR** utilizando listas encadeadas para os problemas médios do repositório *online*.

Tabela 5.18: Decomposição **QR** utilizando estrutura de listas encadeadas - problemas do repositório com $1080 \leq n \leq 6537$.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
sherman2	1080	23094	1,98	98,02	$1,60 \times 10^{-2}$	$7,20 \times 10^{3}$	0*	$7,22 \times 10^{3}$	$1,44 \times 10^{4}$	$1,34 \times 10^{-1}$
bcsstm09	1083	1083	0,09	99,91	0*	$1,86 \times 10^{1}$	0*	$1,86 \times 10^{1}$	$3,73 \times 10^{1}$	$3,73 \times 10^{-13}$
bcsstm10	1086	22092	0,98	99,02	$1,60 \times 10^{-2}$	$3,97 \times 10^{1}$	0*	$3,97 \times 10^{1}$	$7,94 \times 10^{1}$	$7,77 \times 10^{-11}$
sherman4	1104	3786	031	99,69	0*	$3,12 \times 10^{3}$	0*	$3,12 \times 10^{3}$	$6,25 \times 10^{3}$	$1,84 \times 10^{-11}$
1138_bus	1138	2596	0,20	99,80	$3,10 \times 10^{-2}$	$5,68 \times 10^{3}$	0*	$5,69 \times 10^{3}$	$1,14 \times 10^{4}$	0*
rdb1250	1250	7300	0,47	99,53	$1,60 \times 10^{-2}$	$1,05 \times 10^{4}$	0*	$1,06 \times 10^{4}$	$2,11 \times 10^{4}$	0*
c-19	2327	12072	0,22	99,78	$1,60 \times 10^{-2}$	$2,23 \times 10^{2}$	0*	$2,23 \times 10^{2}$	$4,47 \times 10^{2}$	$8,67 \times 10^{-17}$
cavity10	2597	76367	1,13	98,87	-	-	-	-	-	-
sherman5	3312	20793	0,19	99,81	-	-	-	-	-	-
bcsstm21	3600	3600	0,03	99,97	0*	$6,92 \times 10^{2}$	0*	$6,92 \times 10^{2}$	$1,38 \times 10^{3}$	$3,26 \times 10^{-12}$
c-26	4307	19422	0,10	99,90	-	-	-	-	-	-
c-28	4598	17594	0,08	99,92	$1,50 \times 10^{-2}$	$1,66 \times 10^{3}$	0*	$1,66 \times 10^{3}$	$3,33 \times 10^{3}$	0*
sherman3	5005	20033	0,08	99,92	-	-	-	-	-	-
c-29	5033	24382	0,10	99,90	$3,10 \times 10^{-2}$	$2,43 \times 10^{3}$	0*	$2,44 \times 10^{3}$	$4,87 \times 10^{3}$	0*
c-30	5321	35507	0,12	99,87	$1,09 \times 10^{-1}$	$2,76 \times 10^{3}$	0*	$2,76 \times 10^{3}$	$5,53 \times 10^{3}$	0*
c-31	5339	41955	0,15	99,85	$4,70 \times 10^{-2}$	$3,13 \times 10^{3}$	0*	$3,13 \times 10^{3}$	$6,25 \times 10^{3}$	0*
c-32	5975	30223	0,08	99,91	$9,40 \times 10^{-2}$	$3,55 \times 10^{3}$	0*	$3,55 \times 10^{3}$	$7,09 \times 10^{3}$	0*
c-33	6317	31220	0,08	99,92	$4,70 \times 10^{-2}$	$4,07 \times 10^{3}$	0*	$4,07 \times 10^{3}$	$8,15 \times 10^{3}$	0*
c-35	6537	34714	0,08	99,92	$3,20 \times 10^{-2}$	$5,21 \times 10^{3}$	0*	$5,21 \times 10^{3}$	$1,04 \times 10^{4}$	0*
Média	3526,79	22517,53	0,34	99,66	$3,13 \times 10^{-2}$	$3,35 \times 10^{3}$	0*	$3,36 \times 10^{3}$	$6,71 \times 10^{3}$	$8,93 \times 10^{-3}$

Fonte: Elaboração própria.

Na Tabela 5.18, os "traços" indicam problemas que não chegaram em uma solução dentro do tempo máximo que foi utilizado como limite para testes, 32 h. O menor tempo registrado na Tabela 5.18 foi do problema *"bcsstm09"*: 37,3 segundos. Inclusive o tempo de resolução total na Tabela 5.18 cresceu juntamente com a ordem de grandeza dos problemas. Fogem a essa regra apenas alguns dos primeiros problemas apresentados como *"sherman2"*, *"sherman4"*, *"1138_bus"* e *"rdb1250"*.

Vale lembrar que problemas com ordem de grandeza superior a 3600 (ou superior ao *"bcsstm21"*) não foram solucionados pela fatoração **QR** com a estrutura indexada. Ou seja, com a estrutura de dados apropriada, no caso lista encadedada o método **QR** resolveu os problemas mantendo a norma do resíduo quase nula. O maior valor para norma de resíduo encontrado na Tabela 5.18 foi de $1,34 \times 10^{-1}$ com o problema *"sherman2"*. Inclusive, esse é o problema de menor ordem de grandeza da tabela.

Na Tabela 5.19 encontram-se os resultados da fatoração **QR** usando listas encadeadas para os maiores problemas dentre os que foram selecionados do repositório. Observa-se que neste caso, apenas três problemas não foram solucionados dentro do tempo limite (32 h), são eles: *"pres_poison"*, *"c-48"* e *"c-49"*.

Tabela 5.19: Decomposição **QR** utilizando estrutura de listas encadeadas - problemas do repositório com $6611 \leq n \leq 21132$.

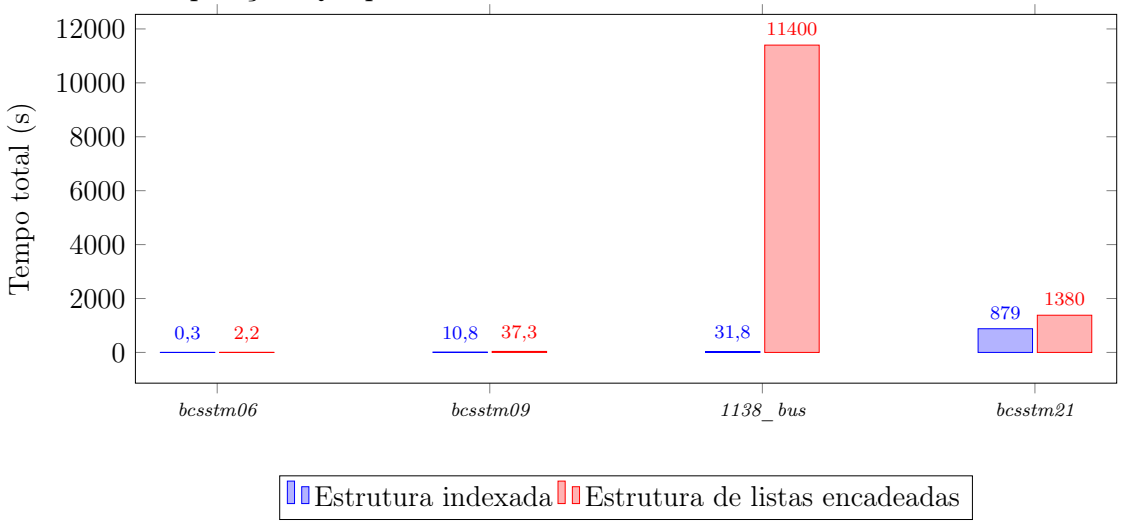
Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRSLI$ (s)	$TRSLS$ (s)	TT (s)	$\|\|R\|\|$
c-34	6611	35472	0,08	99,92	$3,20 \times 10^{2}$	$4,91 \times 10^{3}$	0*	$4,91 \times 10^{3}$	$9,81 \times 10^{3}$	0*
c-36	7479	36710	0,06	99,93	$4,60 \times 10^{-2}$	$6,69 \times 10^{3}$	0*	$6,70 \times 10^{3}$	$1,34 \times 10^{4}$	0*
c-38	8127	42908	0,06	99,93	$4,70 \times 10^{-2}$	$9,74 \times 10^{3}$	0*	$9,74 \times 10^{3}$	$1,95 \times 10^{4}$	$2,01 \times 10^{-2}$
c-37	8204	41440	0,06	99,94	$4,60 \times 10^{-2}$	$1,09 \times 10^{4}$	0*	$1,09 \times 10^{4}$	$2,19 \times 10^{4}$	$3,12 \times 10^{-3}$
c-39	9271	73041	0,08	99,91	$7,80 \times 10^{-2}$	$1,65 \times 10^{4}$	0*	$1,65 \times 10^{4}$	$3,31 \times 10^{4}$	$4,63 \times 10^{-3}$
c-41	9769	55757	0,08	99,91	$6,20 \times 10^{-2}$	$1,79 \times 10^{4}$	0*	$1,79 \times 10^{4}$	$3,58 \times 10^{4}$	$3,10 \times 10^{-2}$
c-40	9941	45721	0,05	99,95	$9,40 \times 10^{-2}$	$1,58 \times 10^{4}$	0*	$1,58 \times 10^{4}$	$3,16 \times 10^{4}$	$4,21 \times 10^{-2}$
c-42	10471	60378	0,05	99,94	$1,25 \times 10^{-1}$	$2,03 \times 10^{4}$	0*	$2,03 \times 10^{4}$	$4,06 \times 10^{4}$	$7,16 \times 10^{-2}$
c-44	10728	47864	0,04	99,96	$4,70 \times 10^{-2}$	$2,18 \times 10^{4}$	0*	$2,18 \times 10^{4}$	$4,35 \times 10^{4}$	$7,72 \times 10^{-3}$
c-43	11125	67400	0,05	99,94	$2,19 \times 10^{-1}$	$2,44 \times 10^{4}$	0*	$2,44 \times 10^{4}$	$4,88 \times 10^{4}$	$3,80 \times 10^{-2}$
c-45	13206	93829	0,05	99,95	$3,44 \times 10^{-1}$	$3,87 \times 10^{4}$	0*	$3,87 \times 10^{4}$	$7,74 \times 10^{4}$	$2,05 \times 10^{-1}$
pres_poison	14822	356313	0,17	99,83	-	-	-	-	-	-
c-46	14913	72655	0,03	99,97	$1,41 \times 10^{-1}$	$5,83 \times 10^{4}$	0*	$5,84 \times 10^{4}$	$1,17 \times 10^{5}$	$2,89 \times 10^{-2}$
c-47	15343	113372	0,05	99,95	$2,03 \times 10^{-1}$	$7,09 \times 10^{4}$	0*	$7,09 \times 10^{4}$	$1,42 \times 10^{5}$	$2,05 \times 10^{-2}$
bcsstm25	15439	15439	0,01	99,99	$1,50 \times 10^{-2}$	$5,61 \times 10^{4}$	0*	$5,61 \times 10^{4}$	$1,12 \times 10^{5}$	$2,98 \times 10^{-8}$
c-48	18354	92217	0,03	99,97	-	-	-	-	-	-
c-49	21132	89087	0,02	99,98	-	-	-	-	-	-
Média	12055	78800,18	0,06	99,94	$2,29 \times 10^{1}$	$2,66 \times 10^{4}$	0*	$2,66 \times 10^{4}$	$5,33 \times 10^{4}$	$3,38 \times 10^{-2}$

Fonte: Elaboração própria.

Nota-se que os tempos de solução dos problemas cresceram significativamente se comparados a Tabela 5.14, onde os mesmos problemas são solucionados com fatoração **LU** e estrutura de dados de listas encadeadas. No entanto, as normas dos resíduos encontrados permanecem as mesmas. A pior norma de resíduo foi de $2,05 \times 10^{-1}$ relativa ao problema *"c-45"*. E as normas, neste caso não foram muito boas, mas podem ser consideradas razoáveis, pois os problemas são relativamente grandes e nenhum deles pôde ser solucionado com estrutura indexada.

A Figura 5.11 compara os tempos totais de resolução de alguns problemas que puderam ser resolvidos com ambas estruturas (indexada e de listas encadeadas) e obtiveram resíduos semelhantes usando a fatoração **QR**.

Figura 5.11: Comparação do tempo total de resolução de problemas do repositório *online* utilizando decomposição **QR** para estrutura indexada e de listas encadeadas.



Fonte: Elaboração própria.

Para o método de fatoração **QR**, nota-se que a estrutura de listas encadeadas embora tenha resolvido um número maior de problemas se comparada a estrutura indexada, quando os problemas puderam ser resolvidos por ambas as estruturas, as listas encadeadas gastaram um tempo maior para fazê-lo.

O próximo método numérico apresentado será a fatoração de Cholesky. Para apresentação dos resultados usando as estruturas indexada e de listas encadeadas foram excluídos das tabelas a seguir os problemas que, por limitação do método, não podem ser resolvidos, ou seja, os problemas em que a matriz **A** não é simétrica e positiva definida simultaneamente, e dessa forma não podem ser resolvidos pelo método de Cholesky.

A Tabela 5.20 mostra os resultados obtidos dos testes utilizando o método de fatoração de Cholesky com estrutura de dados indexada para os problemas do repositório *online*.

Tabela 5.20: Decomposição Cholesky utilizando estrutura indexada - problemas do repositório.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRS1$ (s)	$TRS2$ (s)	TT (s)	$\|\|R\|\|$
Trefethen_20b	19	83	22,99	77,01	0*	0*	0*	0*	0*	1,65
Trefethen_20	20	89	22,25	77,75	0*	0*	0*	0*	0*	1,93
bcsstk01	48	224	9,72	90,28	0*	0*	0*	0*	0*	0*
bcsstm02	66	66	1,51	98,48	0*	0*	0*	0*	0*	$1,32 \times 10^{-14}$
Trefethen_150	150	1095	4,87	95,13	0*	0*	0*	0*	0*	$1,26 \times 10^{1}$
Trefethen_200	200	1545	3,86	96,14	0*	0*	0*	0*	0*	$3,20 \times 10^{1}$
Trefethen_300	300	2489	2,76	97,23	0*	$1,60 \times 10^{-2}$	0*	0*	$1,60 \times 10^{-2}$	$9,54 \times 10^{1}$
bcsstm06	420	420	0,24	99,76	0*	$4,70 \times 10^{-2}$	0*	0*	$4,70 \times 10^{-2}$	$2,79 \times 10^{-12}$
bcsstm07	420	3836	2,17	97,82	0*	$4,70 \times 10^{-2}$	0*	0*	$4,70 \times 10^{-2}$	$1,59 \times 10^{4}$
Trefethen_500	500	8478	1,79	98,20	$1,60 \times 10^{-2}$	$7,80 \times 10^{-2}$	0*	$1,50 \times 10^{-2}$	$1,09 \times 10^{-1}$	$1,81 \times 10^{2}$
Trefethen_700	700	6677	1,36	98,64	$1,60 \times 10^{-2}$	$2,34 \times 10^{-1}$	0*	0*	$2,50 \times 10^{-1}$	$2,82 \times 10^{2}$
bcsstm09	1083	1083	0,09	99,91	$3,10 \times 10^{-2}$	$8,60 \times 10^{-1}$	0*	$1,50 \times 10^{-2}$	$9,06 \times 10^{-1}$	$1,36 \times 10^{-12}$
bcsstm10	1086	22092	0,98	99,02	$1,50 \times 10^{-2}$	$8,91 \times 10^{-1}$	0*	$3,10 \times 10^{-2}$	$9,37 \times 10^{-1}$	$7,73 \times 10^{4}$
1138_bus	1138	2596	0,20	99,80	$1,60 \times 10^{-2}$	1,03	0*	0*	1,05	0*
bcsstm21	3600	3600	0,03	99,97	$1,40 \times 10^{-1}$	$3,14 \times 10^{1}$	$1,25 \times 10^{-1}$	$1,87 \times 10^{-1}$	$3,18 \times 10^{1}$	$7,84 \times 10^{-12}$
pres_poison	14822	356313	0,17	99,83	EM	EM	EM	EM	EM	EM
bcsstm25	15439	15439	0,01	99,99	EM	EM	EM	EM	EM	EM
Média	2353,59	25066,18	4,41	95,58	$1,56 \times 10^{-2}$	2,31	$8,33 \times 10^{-3}$	$1,65 \times 10^{-2}$	2,34	$6,25 \times 10^{3}$

Fonte: Elaboração própria.

Os problemas que foram resolvidos levaram um tempo total máximo de 31,8 segundos, que pode ser considerado rápido, se comparado aos métodos anteriores. No entanto, a norma dos resíduos foi alta em muitos dos casos estudados e consequentemente sua média também. Por isso, algumas soluções sequer podem ser consideradas. Destaca-se como bons resultados os apresentados aos problemas: *"bcsstk01"*, *"bcsstm02"*, *"bcsstm06"*, *"bcsstm09"*, *"1138_bus"* e *"bcsstm21"*, apenas.

Assim como nos testes anteriores, há um crescimento no tempo total à medida que a ordem de grandeza do problema aumenta. O tempo total de teste permanece bastante influenciado pelo tempo de decomposição da matriz (cerca de 98,72% do tempo total). Por fim, os problemas *"pres_poison"* e *"bcsstm25"* não puderam ser solucionados devido ao estouro de memória.

A Tabela 5.21 apresenta os resultados obtidos para os mesmos problemas utilizando o mesmo método com a estrutura de listas encadeadas.

Tabela 5.21: Decomposição Cholesky utilizando estrutura de listas encadeadas - problemas do repositório.

Problema	n	nz	D (%)	E (%)	TL (s)	TD (s)	$TRSLI$ (s)	$TRSLS$ (s)	TT (s)	$\|\|R\|\|$
Trefethen_20b	19	83	22,99	77,01	0*	0*	0*	0*	0*	$1,65$
Trefethen_20	20	89	22,25	77,75	0*	0*	0*	0*	0*	$1,93$
bcsstk01	48	224	9,72	90,28	0*	0*	0*	0*	0*	0*
bcsstm02	66	66	1,51	98,48	0*	0*	0*	0*	0*	$1,33 \times 10^{-14}$
Trefethen_150	150	1095	4,87	95,13	0*	0*	0*	0*	0*	$1,26 \times 10^{1}$
Trefethen_200	200	1545	3,86	96,14	0*	0*	0*	0*	0*	$3,20 \times 10^{1}$
Trefethen_300	300	2489	2,76	97,23	0*	0*	0*	0*	0*	$9,54 \times 10^{1}$
bcsstm06	420	420	0,24	99,76	0*	0*	0*	0*	0*	$2,79 \times 10^{-12}$
bcsstm07	420	3836	2,17	97,82	0*	0*	0*	0*	0*	$1,59 \times 10^{4}$
Trefethen_500	500	8478	1,79	98,20	0*	0*	0*	0*	0*	$1,81 \times 10^{2}$
Trefethen_700	700	6677	1,36	98,64	0*	0*	0*	0*	0*	$2,82 \times 10^{2}$
bcsstm09	1083	1083	0,09	99,91	0*	0*	0*	0*	0*	$1,36 \times 10^{-12}$
bcsstm10	1086	22092	0,98	99,02	$1,60 \times 10^{-2}$	$1,60 \times 10^{-2}$	0*	0*	$3,20 \times 10^{-2}$	$7,72 \times 10^{4}$
1138_bus	1138	2596	0,20	99,80	$1,60 \times 10^{-2}$	$6,20 \times 10^{-2}$	0*	$6,30 \times 10^{-2}$	$1,41 \times 10^{-1}$	0*
bcsstm21	3600	3600	0,03	99,97	1.60×10^{-2}	$1,25 \times 10^{-1}$	0*	0*	$1,41 \times 10^{-1}$	$7,84 \times 10^{-12}$
pres_poison	14822	356313	0,17	99,83	$4,69 \times 10^{-1}$	$1,16 \times 10^{2}$	$5,79 \times 10^{-1}$	$1,18 \times 10^{2}$	$2,36 \times 10^{2}$	0*
bcsstm25	15439	15439	0,01	99,99	$1,60 \times 10^{-2}$	2,44	0*	$3,60 \times 10^{-1}$	2,81	$1,04 \times 10^{-7}$
Média	2353,59	25066,18	4,41	95,58	$3,13 \times 10^{-2}$	6,98	$3,41 \times 10^{-2}$	6,97	$1,41 \times 10^{1}$	$5,51 \times 10^{3}$

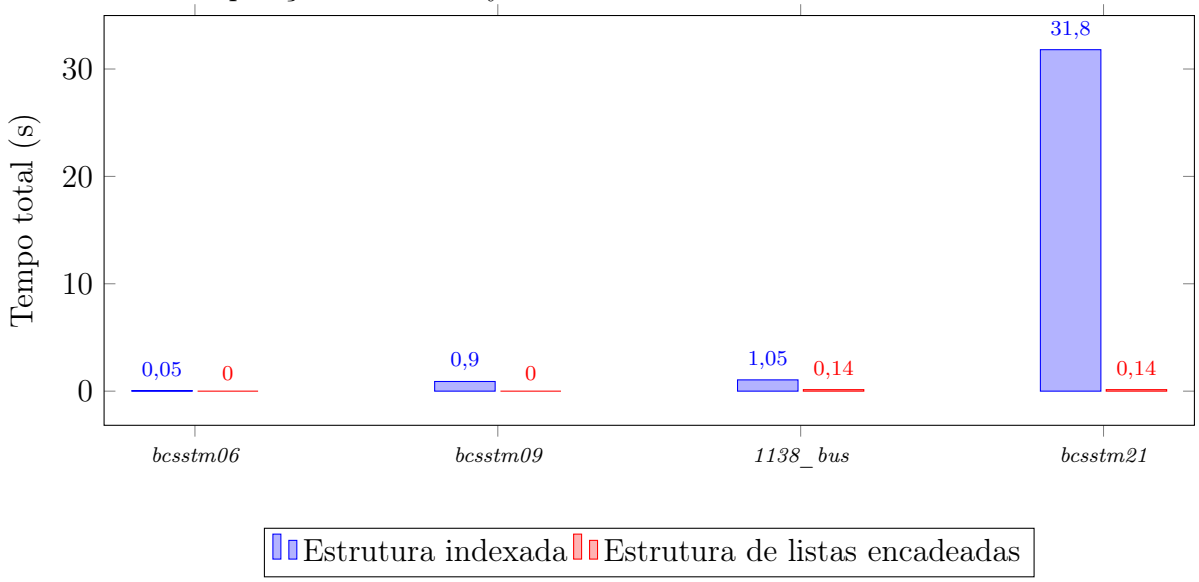
Fonte: Elaboração própria.

Nota-se que neste caso, o tempo total para a solução da maioria dos problemas foi bem próximo de zero, a exceção de *"pres_poison"* que foi menos de 4 minutos. Entretanto, a norma do resíduo de muitas soluções foi alta (assim como na estrutura indexada), sendo ineficiente em muitos casos, isto é, a solução encontrada pelo método está muito distante da solução analítica.

Dessa forma, o método conseguiu resolver de maneira eficiente apenas alguns problemas como: *"bcsstk01"*, *"bcsstm02"*, *"bcsstm06"*, *"bcsstm09"*, *"1138_bus"*, *"bcsstm21"*, *"pres_poison"* e *"bcsstm25"*, obtendo-se nesses casos uma norma do resíduo máxima de $1,04 \times 10^{-7}$. Vale destacar que o uso da fatoração de Cholesky com estrutura de listas encadeadas permitiu solucionar os problemas *"pres_poison"* e *"bcsstm25"* em um ótimo tempo se comparado à fatoração **LU** ou **QR** (usando estrutura indexada ou listas encadeadas); como apresentado nas Tabelas 5.11, 5.14 e 5.19.

A Figura 5.12 permite comparar os tempos de resolução para problemas em que o método de Cholesky forneceu soluções satisfatórias, utilizando estrutura indexada e de listas encadeadas.

Figura 5.12: Comparação do tempo total de resolução de problemas do repositório *online* utilizando decomposição de Cholesky com estrutura indexada e de listas encadeadas.



Fonte: Elaboração própria.

Embora o método de Cholesky tenha falhado ao fornecer solução para alguns dos problemas estudados, nos problemas em que forneceu uma solução válida usando listas encadeadas, o tempo foi bem menor que o tempo encontrado nos demais métodos. Isso pode ser observado comparando os valores apresentados nas Figuras 5.10, 5.11 e 5.12.

A seguir, foram selecionados alguns problemas passíveis de resolução nos três métodos (fatoração **LU**, **QR** e de Cholesky) em que os testes apresentaram resultados válidos a fins comparativos de tempo de resolução.

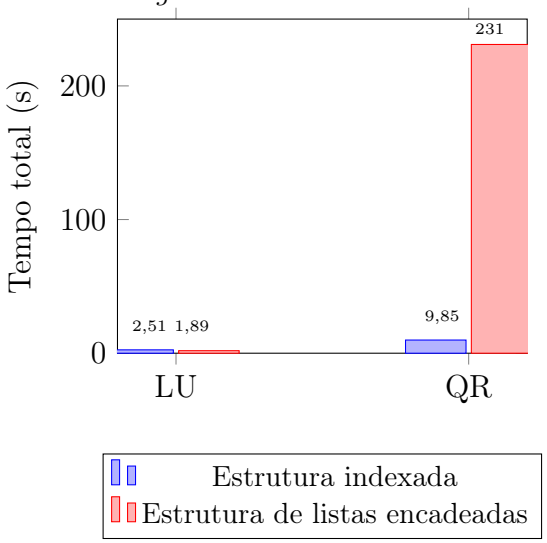
5.5 Comparações dos métodos utilizados

Dentre os problemas tratados, quando utiliza-se a estrutura de dados indexada, o estouro de memória (*"overflow"*) é frequente acima de determinados valores de ordem de grandeza para os métodos **QR**, **LU** e de Cholesky. Em contrapartida, não houve estouro de memória quando utiliza-se a estrutura de listas encadeadas. No entanto, nem todos os resultados obtidos podem ser considerados bons resultados, devido a norma do resíduo das soluções. Consideram-se bons resultados aqueles menores que 10^{-5} e ótimos os menores de 10^{-10}.

Entre os bons resultados, ou seja, aqueles cujo resíduo foi nulo ou muito próximo disso, são exibidos a seguir gráficos comparativos quanto ao tempo total (em segundos) para solução de alguns dos problemas estudados utilizando os três métodos numéricos estudados.

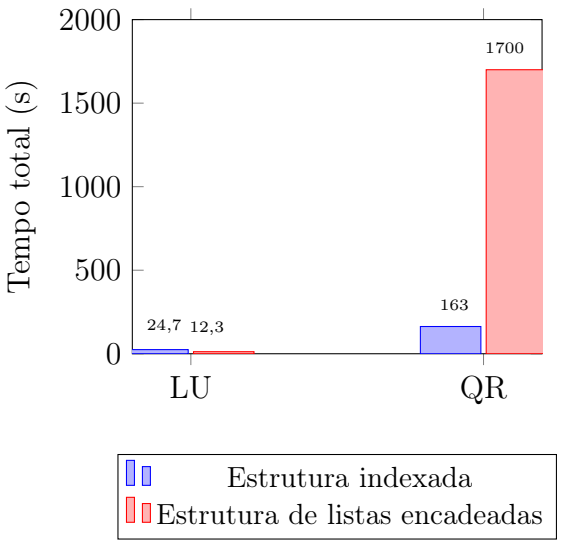
O primeiros problemas apresentados são referentes ao gerador aleatório. Os problemas do gerador aleatório foram resolvidos apenas pelos métodos de fatoração **LU** e **QR**, como comentado anteriormente. A seguir, as Figuras 5.13 e 5.14 mostram os tempos para solução dos problemas *"tridiag1000"* e *"tridiag2000"*, respectivamente.

Figura 5.13: Tempo de resolução do problema *"tridiag1000"*.



Fonte: Elaboração própria.

Figura 5.14: Tempo de resolução do problema *"tridiag2000"*.



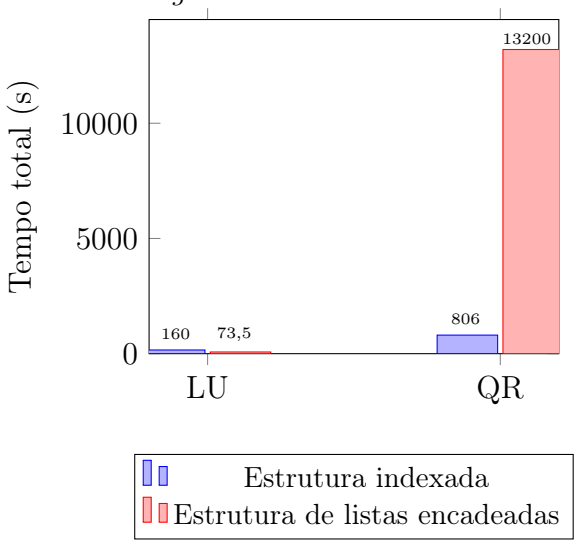
Fonte: Elaboração própria.

Para *"tridiag1000"* o tempo de resolução utilizando fatoração **LU** e listas encadeadas corresponde a 75% do tempo utilizando o mesmo método e a estrutura indexada. Se comparado ao tempo de resolução com fatoração **QR**, é mais de cinco vezes menor para estrutura indexada e mais de cem vezes menor com uso de listas encadeadas.

Nota-se que para*"tridiag2000"* o tempo de solução com a fatoração **LU** e estrutura de listas encadeadas foi cerca de metade do tempo se utilizado o mesmo método e a estrutura indexada. Quando comparado aos tempos de resolução com fatoração **QR**, os valores são ainda maiores. Usando fatoração **QR** e a estrutura indexada o tempo é cerca de treze vezes maior e com listas encadeadas chega a 138 vezes maior que a fatoração **LU** e listas encadeadas.

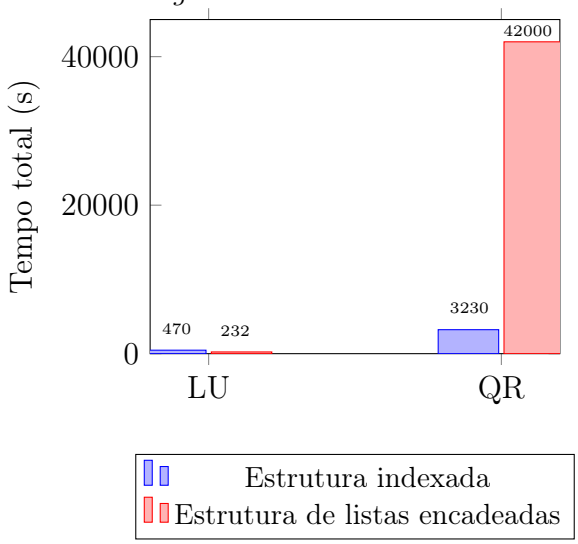
Logo, para ambos, o método de fatoração **LU** com uso da estrutura de listas encadeadas forneceu uma solução válida em bem menos tempo. O mesmo ocorre para os problemas a seguir: *"tridiag3500"* apresentado na Figura 5.15, *"tridiag5000"* na Figura 5.16 e *"tridiag6500"* na Figura 5.17.

Figura 5.15: Tempo de resolução do problema *"tridiag3500"*.



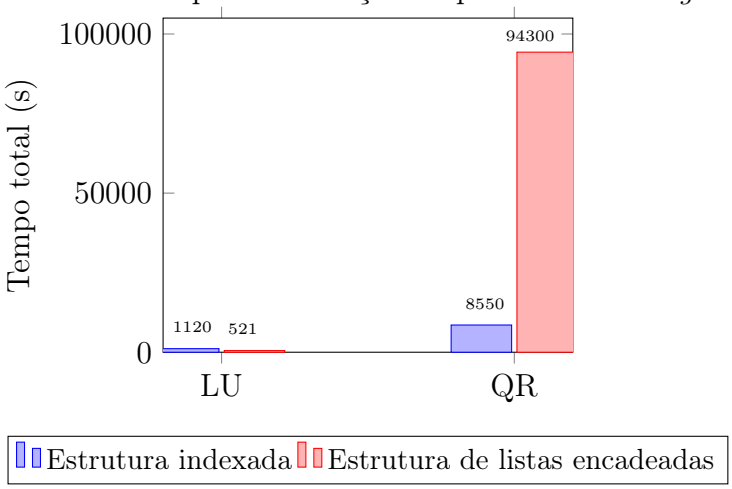
Fonte: Elaboração própria.

Figura 5.16: Tempo de resolução do problema *"tridiag5000"*.



Fonte: Elaboração própria.

Figura 5.17: Tempo de resolução do problema *"tridiag6500"*.



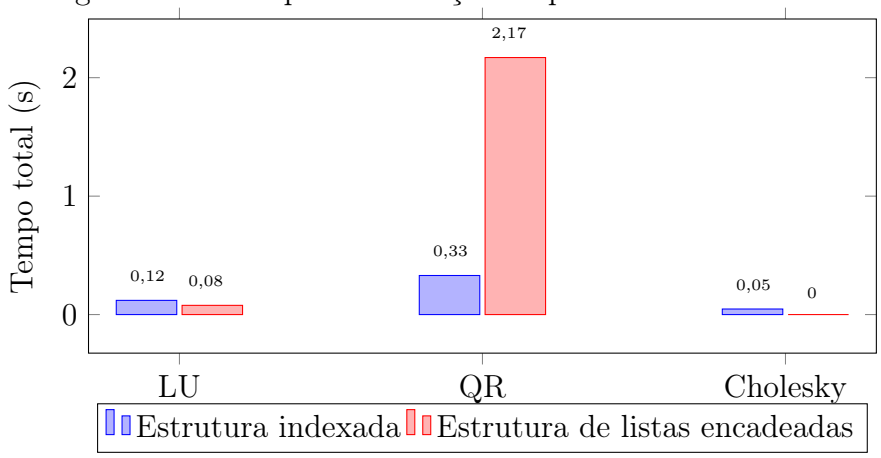
Fonte: Elaboração própria.

Assim, para os problemas do gerador aleatório tem-se que a fatoração **LU** com uso de listas encadeadas foi capaz de fornecer soluções válidas para um maior número de problemas em um tempo muito menor que o método **QR** (usando estrutura indexada ou de listas encadeadas). No geral, a estrutura de listas encadeadas não foi interessante, em termos de tempo computacional, quando o método numérico utilizado foi fatoração **QR**.

Quanto aos problemas do repositório *online* considera-se inicialmente o problema *"bcsstk01"*, pois obteve tempo total e resíduo aproximadamente nulo em todos os testes. Isso mostra que para problemas pequenos (com ordem de grandeza na casa das dezenas), os três métodos **LU**, **QR** ou de Cholesky são eficientes.

A Figura 5.18 mostra os tempos de resolução para o problema *"bcsstm06"*.

Figura 5.18: Tempo de resolução do problema *"bcsstm06"*.

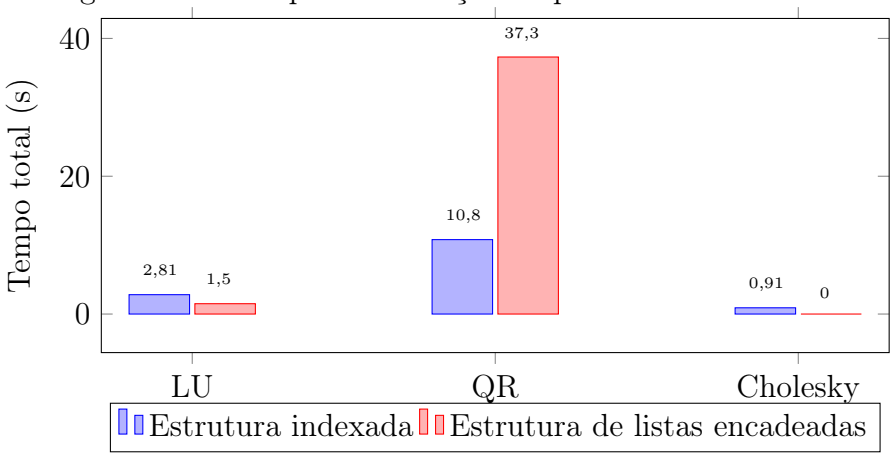


Fonte: Elaboração própria.

Para o problema *"bcsstm06"*, nota-se que o menor tempo foi obtido com método de fatoração de Cholesky (com listas encadeadas e estrutura indexada), seguido da fatoração **LU** com listas encadeadas e estrutura indexada e por último fatoração **QR** com listas encadeadas. Nota-se, como já comentado anteriormente, que a estrutura de listas encadeadas não é uma alternativa interessante quando usado o método de fatoração **QR**, pois esse é o único caso em que as listas encadeadas oferecem um tempo computacional maior que a estrutura indexada.

A Figura 5.19 mostra o gráfico com os tempos para o problema *"bcsstm09"*, cuja ordem de grandeza é de 1083.

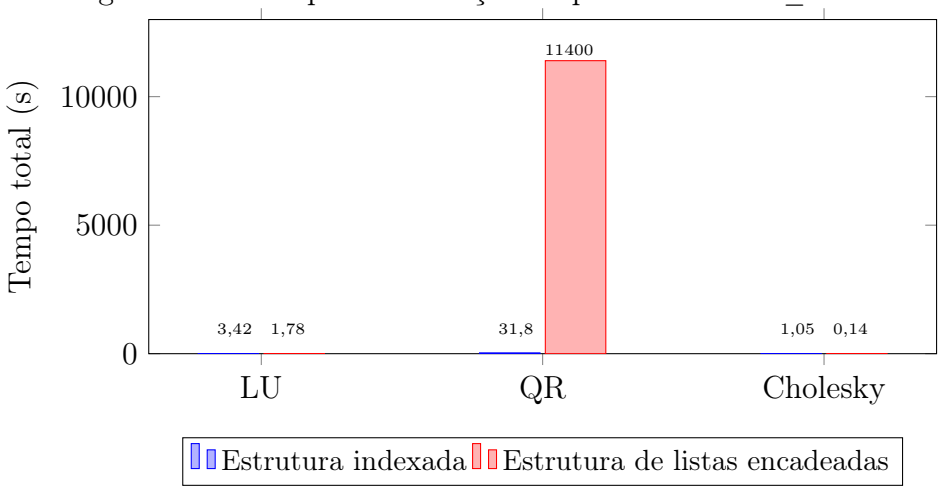
Figura 5.19: Tempo de resolução do problema *"bcsstm09"*.



Fonte: Elaboração própria.

Nota-se que a fatoração de Cholesky seguida da fatoração **LU** resolveram o problema *"bcsstm09"* em um tempo melhor que a fatoração **QR**. Em termos de estrutura, para fatoração **LU** e Cholesky, as listas encadeadas foram uma boa alternativa a estrutura indexada. A Figura 5.20 fornece o comparativo entre os tempos do problema *"1138_ bus"*, cuja ordem de grandeza é de 1138.

Figura 5.20: Tempo de resolução do problema *"1138_bus"*.



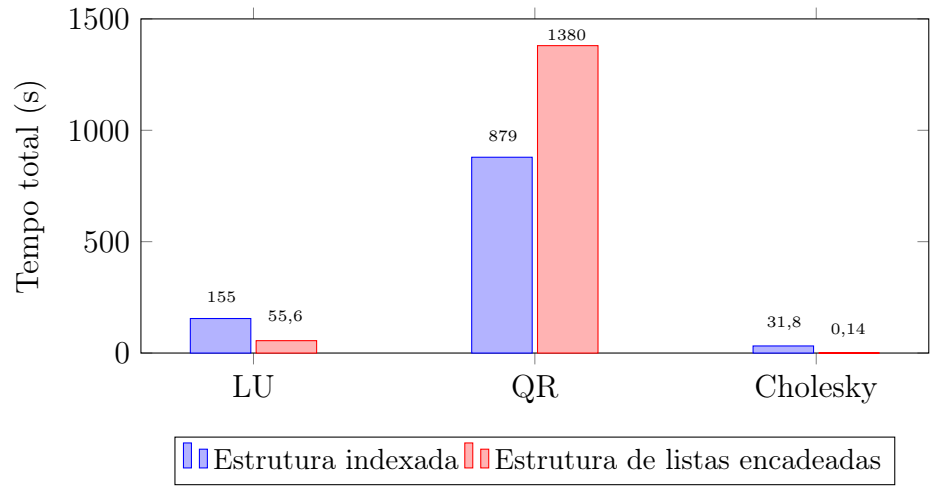
Fonte: Elaboração própria.

À medida que a ordem de grandeza do problema aumenta, fica mais evidente que o método de fatoração **QR** teve o pior desempenho dentre os três métodos estudados, principalmente considerando a estrutura de listas encadeadas.

Para o problema *"1138_bus"*, enquanto o tempo de resolução usando a fatoração de Cholesky ou **LU** levou menos de 4 segundos, a fatoração **QR** utilizando listas encadeadas demorou mais de 3 horas para resolver o mesmo problema.

A Figura 5.21 mostra os tempos de resolução do problema *"bcsstm21"*, cuja ordem de grandeza é 3600.

Figura 5.21: Tempo de resolução do problema *"bcsstm21"*.

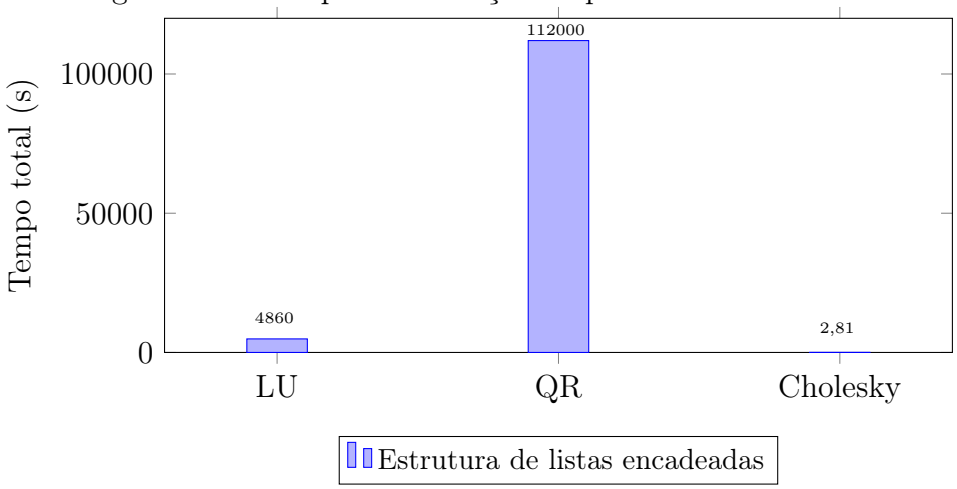


Fonte: Elaboração própria.

Assim, como nos problemas mostrados anteriormente, os tempos de resolução utilizando fatoração de Cholesky e fatoração **LU** foram inferior ao tempo da fatoração **QR** para o problema *"bcssstm21"*. E o uso da estrutura de listas encadeadas foi eficiente para os métodos numéricos de fatoração **LU** e de Cholesky.

A Figura 5.22 refere-se ao gráfico do problema *"bcsstm25"*, de ordem 15439.

Figura 5.22: Tempo de resolução do problema *"bcsstm25"*.



Fonte: Elaboração própria.

O problema *"bcsstm25"*, assim como outros apresentados anteriormente nas Tabelas 5.11 e 5.16 não pôde ser solucionado com a estrutura de dados indexada em nenhum dos métodos testados (**LU**, **QR** e de Cholesky). No entanto, foi resolvido utilizando listas encadeadas com todos os três métodos numéricos. Pela Figura 5.22, nota-se que os melhores tempos foram obtidos com o método de Cholesky e fatoração **LU**.

Dessa forma, como observado nas tabelas e gráficos anteriores, no geral o método numérico com melhor desempenho nos testes foi fatoração **LU**, tendo em vista a qualidade dos resultados apresentados em todos os problemas testados (com Cholesky muitos resíduos foram ruins) e o tempo de resolução dos mesmos.

Com relação a estrutura de dados, como já comentado anteriormente, as listas foram mais eficientes que a estrutura indexada quando usadas com os métodos de fatoração **LU** ou Cholesky. Para fatoração **QR**, o uso de listas encadeadas não mostrou ser uma boa opção em termos de tempo computacional.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

Resolver sistemas lineares com ordem de grandeza da ordem de milhares de elementos é um desafio. Isso se dá devido ao fato desses problemas em sua maioria causarem um estouro de memória (*"overflow"*), quando utilizada a estrutura indexada. Dessa forma, é útil investigar outras estruturas e métodos que sejam capazes de fornecer soluções razoáveis.

Neste trabalho foram comparados fatoração **LU**, **QR** e Cholesky utilizando estrutura de dados indexada e de lista encadeada. Na estrutura indexada, os elementos são armazenados e acessados a partir de um índice. Com a estrutura de listas encadeadas, apenas os elementos não nulos foram armazenados, evitando cálculos desnecessários e melhorando o desempenho computacional. Essa estratégia permitiu "evitar" estouros de memória e solucionar problemas com milhares de elementos.

Para realizar os testes foram utilizados 68 problemas com ordens de grandeza entre 19 e 21132. Dos quais 19 foram obtidos a partir de um gerador aleatório e 49 problemas são oriundos do repositório *online SuiteSparse Matrix Collection*.

Este trabalho mostrou que o uso da estrutura de dados de lista encadeada é uma alternativa muito viável, visto que fornece solução a um conjunto variado de problemas que muitas vezes a estrutura indexada não atende.

Vale salientar que além da estrutura, a escolha do método númerico influencia significavamente na qualidade da solução e no tempo de resolução do problema. Em termos de tempo computacional, quando se trata de método numérico, a fatoração **QR** obteve os maiores tempos quando comparada a fatoração **LU** e Cholesky. Ressalta-se que embora as soluções apresentadas na fatoração **QR** e fatoração **LU** tenham obtido uma norma de resíduo semelhante, de uma forma geral, a fatoração **LU** resolveu os mesmos problemas em um tempo menor.

A fatoração Cholesky apresentou resultados de qualidade bastante variada, visto que a norma de alguns resíduos foi ruim. Não mostrou dessa forma, ser um método eficiente para boa parte dos problemas estudados, mesmo que tenha obtido alguns resultados bons em tempo inferior aos demais métodos estudados.

6.2 Trabalhos Futuros

Embora a fatoração **LU**, aliada à estrutura de dados de listas encadeadas, tenha solucionado um grande número de problemas, algumas das soluções apresentaram resíduos consideráveis. Dessa forma, propõe-se, como trabalho futuro, o refinamento dessas soluções, em especial daquelas obtidas com o uso da decomposição **LU** e listas encadeadas.

Nesse contexto, visando à melhoria do desempenho, uma alternativa adicional consiste no aprimoramento de determinadas rotinas do código, com o objetivo de reduzir o tempo computacional.

Ademais, como evidenciado neste trabalho, alguns métodos de resolução de sistemas lineares mostraram-se mais eficientes que outros. Assim, para fins comparativos, sugere-se a implementação de outros métodos numéricos — sejam diretos ou iterativos — utilizando a estrutura de listas encadeadas. Essa abordagem permitirá analisar o desempenho desses métodos na resolução de sistemas lineares que surgem como subproblemas em aplicações práticas.

APÊNDICE A – Repositório *SuiteSparse Matrix Collection*

Alguns dos problemas estudados foram retirados do repositório *online SuiteSparse Matrix Collection* mantido por Davis e Hu (2011). A seguir, encontra-se a representação visual de algumas dessas matrizes de coeficientes.

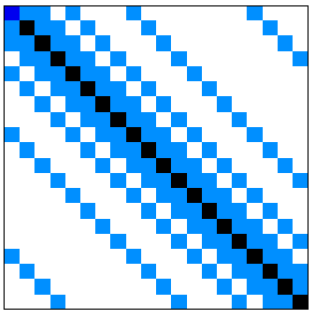


Figura A.1: Padrão de esparsidade da matriz *"Trefethen_20"*.
Fonte: Davis e Hu (2011).

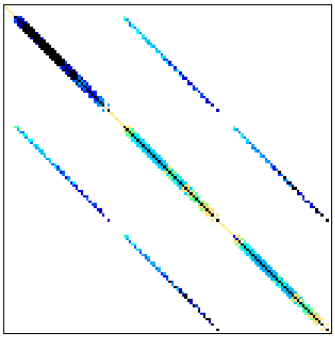


Figura A.2: Padrão de esparsidade da matriz *"sherman4"*.
Fonte: Davis e Hu (2011).

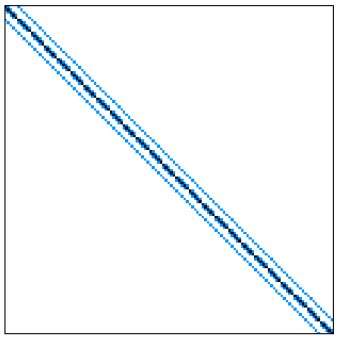


Figura A.3: Padrão de esparsidade da matriz *"rdb1250"*.
Fonte: Davis e Hu (2011).

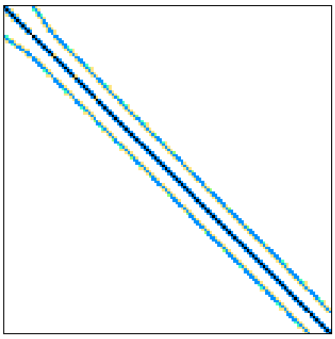


Figura A.4: Padrão de esparsidade da matriz *"cavity10"*.
Fonte: Davis e Hu (2011).

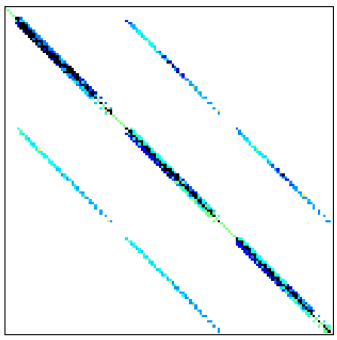


Figura A.5: Padrão de esparsidade da matriz *"sherman5"*.

Fonte: Davis e Hu (2011).

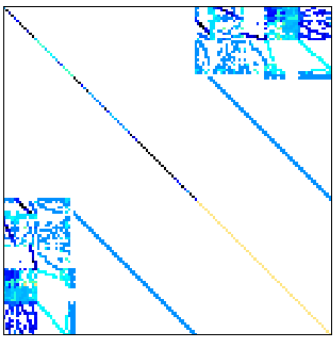


Figura A.6: Padrão de esparsidade da matriz *"c-26"*.

Fonte: Davis e Hu (2011).

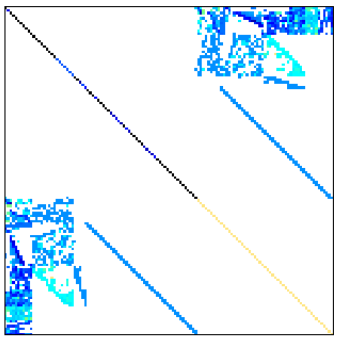


Figura A.7: Padrão de esparsidade da matriz *"c-28"*.

Fonte: Davis e Hu (2011).

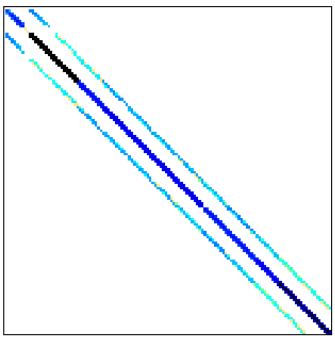


Figura A.8: Padrão de esparsidade da matriz *"sherman3"*.

Fonte: Davis e Hu (2011).

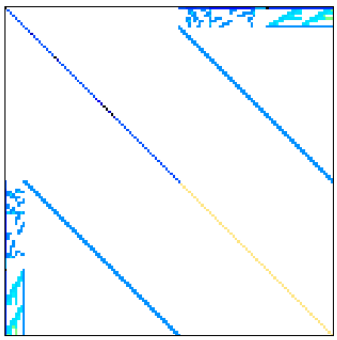


Figura A.9: Padrão de esparsidade da matriz *"c-30"*.

Fonte: Davis e Hu (2011).

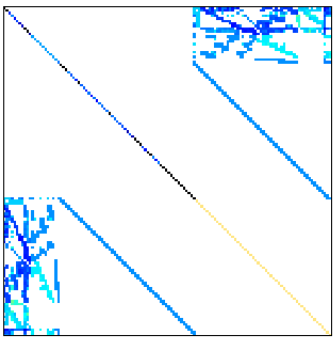


Figura A.10: Padrão de esparsidade da matriz *"c-31"*.

Fonte: Davis e Hu (2011).

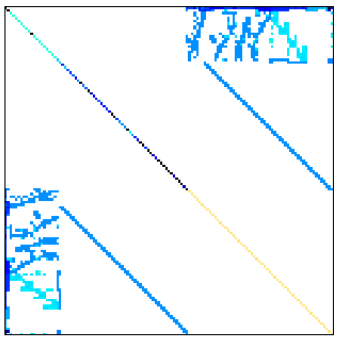


Figura A.11: Padrão de esparsidade da matriz *"c-33"*.

Fonte: Davis e Hu (2011).

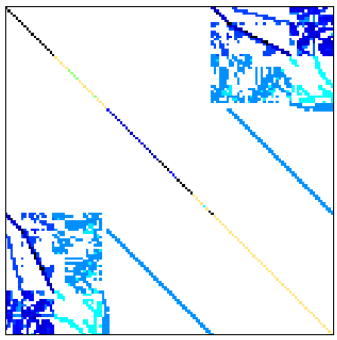


Figura A.12: Padrão de esparsidade da matriz *"c-35"*.

Fonte: Davis e Hu (2011).

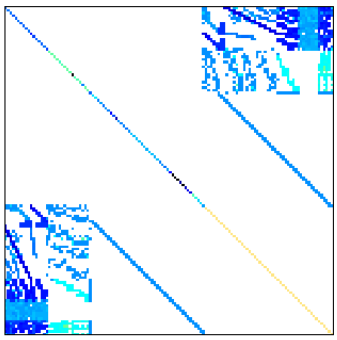


Figura A.13: Padrão de esparsidade da matriz *"c-34"*.

Fonte: Davis e Hu (2011).

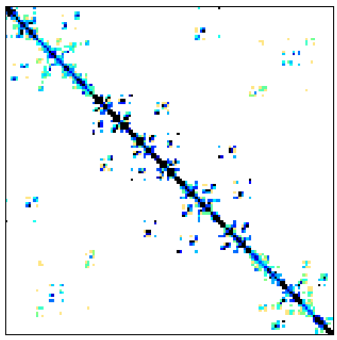


Figura A.14: Padrão de esparsidade da matriz *"press_poisson"*.

Fonte: Davis e Hu (2011).

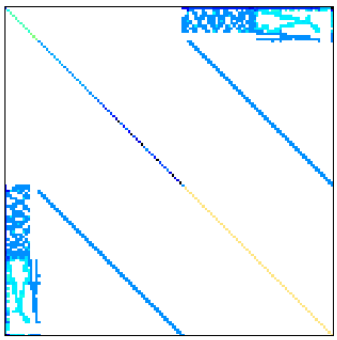


Figura A.15: Padrão de esparsidade da matriz *"c-46"*.

Fonte: Davis e Hu (2011).

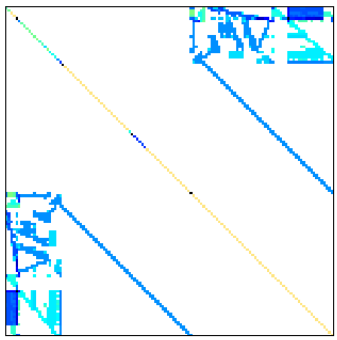


Figura A.16: Padrão de esparsidade da matriz *"c-47"*.

Fonte: Davis e Hu (2011).

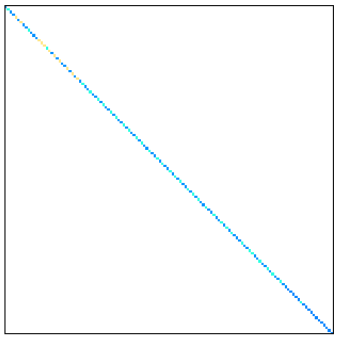


Figura A.17: Padrão de esparsidade da matriz *"bcsstm25"*.

Fonte: Davis e Hu (2011).

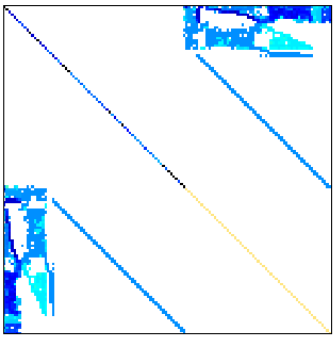


Figura A.18: Padrão de esparsidade da matriz *"c-48"*.

Fonte: Davis e Hu (2011).

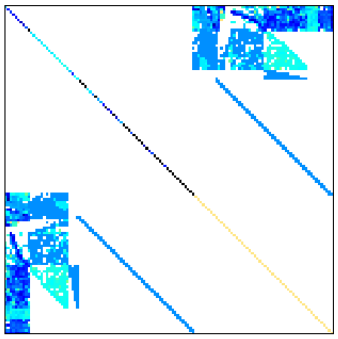


Figura A.19: Padrão de esparsidade da matriz *"c-49"*.

Fonte: Davis e Hu (2011).

Referências

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

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

BURDEN, R. L.; FAIRES, J. D. **Numerical Analysis**. 9. ed. Boston: Cengage Learning, 2011.

CHAPRA, S. C.; CANALE, R. P. **Numerical methods for engineers**. 7. ed. New York: McGraw-Hill Education, 2015.

COELHO, M. A. O. **Métodos Iterativos para Resolver Sistemas de Equações Algébricas Lineares em Estruturas Esparsas**. 2014. Diss. (Mestrado) – Universidade Federal Fluminense.

COLLA, E. C. **Aplicação de técnicas de fatoração de matrizes esparsas para inferência em redes bayesianas**. 2007. Diss. (Mestrado) – Instituto de Matematica e Estatística da Universidade de Sao Paulo.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Algoritmos Teoria e Prática**. Tradução: Arlete Simille Marques. 3. ed. Rio de Janeiro: Elsevier, 2012.

COSTA, E. C. **Estudo de Fluxo de Potência com Aplicação de Métodos Diretos na Resolução de Sistemas de Equações Lineares**. 2008. Diss. (Mestrado) – Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas.

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

DATTA, B. N. **Numerical linear algebra and applications**. 2. ed. Philadelphia: Society for Industrial e Applied Mathematics (SIAM), 2010.

DAVIS, T. A. **Direct Methods for Sparse Linear Systems**. Philadelphia: Society for Industrial e Applied Mathematics (SIAM), 2006.

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

DEMMEL, J. W. **Applied Numerical Linear Algebra**. Philadelphia: Society for Industrial e Applied Mathematics (SIAM), 1997.

DROZDEK, A. **Data Structures and Algorithms in C++**. 4. ed. Boston: Cengage Learning, 2013.

FRANGO, J. V. S. **O Método dos Gradientes Conjugados Precondicionado com Estrutura de Dados CSR**. 2022. Diss. (Mestrado) – Universidade Federal Fluminense.

GNU PROJECT. **GNU Octave: High-Level Language for Numerical Computations**. Boston, 2025. Versão 9.2.0. Disponível em: `https://www.gnu.org/software/octave/`. Acesso em: 4 jun. 2025.

GOLUB, G. H.; LOAN, C. F. V. **Matrix Computations**. 4. ed. Baltimore: Johns Hopkins University Press, 2013.

HEATH, M. T. **Scientific Computing An Introductory Survey**. 2. ed. Philadelphia: Society for Industrial e Applied Mathematics (SIAM), 2002.

JUNIOR, N. G. B. **Métodos Iterativos para Sistemas Lineares**. 2011. Diss. (Mestrado) – Campinas: Departamento de Matemática Aplicada IMECC– UNICAMP.

KING, K. N. **C Programming A Modern Approach**. 2. ed. Nova York: W. W. Norton Company, 2008.

LEVORATO, G. B. P. **Matrizes, Determinantes e Sistemas Lineares: Aplicações na Engenharia e Economia**. 2017. Diss. (Mestrado) – Instituto de Geociências e Ciências Exatas da Universidade Estadual Paulista Júlio de Mesquita Filho.

PESCADOR, A.; POSSAMAI, J. P.; POSSAMAI, C. R. Aplicação de Álgebra Linear na Engenharia. In: ANAIS do Congresso Brasileiro de Educação em Engenharia. [S.l.: s.n.], 2011.

ROVAI, K. R. **Algoritmos e estrutura de dados**. Londrina: Editora e Distribuidora Educacional S.A, 2018.

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

SAAD, Y. **Iterative Methods for Sparse Linear Systems**. 2. ed. Philadelphia: Society for Industrial e Applied Mathematics (SIAM), 2003.

SCILAB ENTERPRISES. **Scilab: Free and Open Source Software for Numerical Computation**. Paris, 2023. Versão 6.1.1. Disponível em: `https://www.scilab.org`. Acesso em: 4 jun. 2025.

SILVA, L. H. **Métodos Iterativos para Sistemas Lineares: Pré-condicionadores, Estruturas de Dados e Outras Técnicas**. 2021. Diss. (Mestrado) – Universidade Federal Fluminense.

TENENBAUM, A. M.; LANGSAM, Y.; AUGENSTEIN, M. J. **Estruturas de dados usando C**. Tradução: Teresa Cristina Félix de Souza. Revisão técnica e adaptação dos programas Roberto Carlos Mayer. 1. ed. São Paulo: Makron Books, 1995.

TREFETHEN, L. N.; BAU, D. **Numerical Linear Algebra**. Philadelphia: Society for Industrial e Applied Mathematics (SIAM), 1997.

WATKINS, D. S. **Fundamentals of Matrix Computations**. 2. ed. Hoboken: Wiley-Interscience, 2002.

WEISS, M. A. **Data Structures and Algorithm Analysis in C++**. 4. ed. Boston: Pearson, 2013.

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