Comparação de Métodos de Armazenamento de Matrizes para o Melhoramento da Simulação em Tempo Real de Redes Elétricas

Raphael Dantas Pinho [*] **Pedro A. de Alcântara** [**]
Luciano Sales Barros [***] **Camila Mara Vital Barros** [***]

[*] *Programa de Pós-graduação em Modelagem Matemática e Computacional, Universidade Federal da Paraíba, PB, (e-mail: raphael.dantas@academico.ufpb.br)*
[**] *Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal da Paraíba, PB, (e-mail: pedro.alcantara@cear.ufpb.br)*
[***] *Departamento de Sistemas de Computação, Universidade Federal da Paraíba, PB, (e-mails: lsalesbarros@ci.ufpb.br, camila.barros@ci.ufpb.br)*

Abstract: The real-time digital simulation (RTDS) of electrical networks is an important analysis tool, with applications in the operation, design, planning and expansion of electrical systems. To make RTDS viable, it is necessary to perform a computational modeling of the electrical network components and their characteristics, from energy generation and distribution and transmission circuits, to loads. This modeling represents a major obstacle for RTDS, since it consists of solving systems of large linear equations. In this work, five array storage methods are compared: *Compressed Sparse Row* (CSR), *Compressed Sparse Column* (CSC), *Compressed Sparse Vector* (CSV), *Skyline* and *DFA2*, which were used in association with the iterative methods of Jacobi and Gauss-Seidel to obtain the solutions of the system of equations that describe the behavior of the electrical network. The results obtained showed that the use of CSR storage techniques with the iterative methods will contribute to improve the performance of RTDS.

Resumo: A simulação digital em tempo real (SDTR) de redes elétricas consiste numa importante ferramenta de análise, com aplicações na operação, projeto, planejamento e ampliação dos sistemas elétricos. Para viabilizar a SDTR, faz-se necessária a modelagem computacional dos componentes das redes elétricas e suas características, desde a geração de energia e circuitos de distribuição e transmissão, até as cargas. Essa modelagem representa grande obstáculo para SDTR, visto que ela consiste na solução de sistemas de equações lineares de grandes dimensões. Neste trabalho, são comparados cinco métodos de armazenamento de matrizes: *Compressed Sparse Row* (CSR), *Compressed Sparse Column* (CSC), *Compressed Sparse Vector* (CSV), *Skyline* e *DFA2*, que foram utilizados em associação aos métodos iterativos de Jacobi e Gauss-Seidel para obter as soluções do sistema de equações que descreve o comportamento da rede elétrica. Os resultados obtidos constataram que o uso da técnica de armazenamento CSR com os métodos iterativos contribuirão para melhorar o desempenho da SDTR.

Keywords: RTDS; Electrical Network; Modeling; Storage Methods; Equation System; Sparse Matrices; Iterative Methods; Real Time.

Palavras-chaves: SDTR; Rede Elétrica; Modelagem; Métodos de Armazenamento; Sistema de Equações; Matrizes Esparsas; Métodos Iterativos; Tempo Real.

1. INTRODUÇÃO

A constante evolução dos equipamentos e das técnicas de planejamento e operação das redes elétricas demandam o uso de ferramentas que possibilitem análises precisas e seguras do comportamento desses sistemas. Nesse sentido, a simulação digital em tempo real (SDTR) é necessária, pois, por meio dos dados obtidos através do seu emprego, é possível prever possíveis falhas e dimensionar o correto funcionamento das partes que compõem as redes.

A modelagem computacional baseia-se na solução numérica de um sistema de equações lineares que tem as tensões de nó da rede elétrica como variáveis e cujos coeficientes consistem na matriz admitância dos componentes da rede passiva, como impedâncias dos circuitos de transmissão e distribuição, e também das cargas, em que os elementos do vetor de entradas correspondem às correntes injetadas pelos geradores. Para a SDTR, a solução do sistema de equações é realizada em um intervalo de tempo, que caracteriza o tempo real, sendo repetida a cada passo de cálculo.

Devido à complexidade das redes elétricas, existem alguns obstáculos que dificultam a viabilização da SDTR; dentre

estes, destaca-se a dimensão da matriz de admitância da rede. Por esta ser grande e ter seus elementos, em maioria, nulos, utilizá-la em sua forma densa causaria aumento do esforço computacional, compromentendo o processo de simulação.

A metodologia adotada neste trabalho para a montagem dessa matriz é a mesma utilizada pelo software OpenDSS. Este, explora a esparsidade da matriz e utiliza os métodos *Compressed Sparse Row* (CSR) e *Compressed Sparse Column* (CSC) associados a um método de ponto fixo iterativo para obter a solução do sistema. A partir disso, foram selecionados cinco métodos de armazenamento de matrizes: CSR, CSC, *Compressed Sparse Vector* (CSV), *Skyline* e *DFA2*, que guardam os termos diferentes de zero e seus respectivos índices, definidos pelas regras de cada uma das técnicas. Para implementar o método de solução do sistema de equações, os métodos iterativos de Jacobi e Gauss-Seidel foram associados aos métodos de armazenamento nos algoritmos. Com o uso desse conjunto de técnicas, propôs-se compará-las, aferindo sua eficiência computacional através do tempo de execução e uso da memória.

A montagem e execução dos algoritmos foi feita em linguagem de programação Python, através do *software* Spyder (versão 5.1.5, python 3.8.8). Seu uso deve-se às funcionalidades e bibliotecas disponíveis, como a *Numpy*, que foi utilizada nos códigos listados na Seção 4, além da possibilidade de adaptação dos códigos construídos a outras linguagens de programação, como o Verilog, que é utilizado em alguns modelos de dispositivos lógico programáveis.

O restante do trabalho está dividido da seguinte forma: a Seção 2 aborda os métodos de armazenamento de matrizes e os métodos escolhidos para resolução do sistema de equações da rede elétrica. A Seção 3 descreve a rede elétrica escolhida e sua matriz de admitância. Na Seção 4, encontram-se os resultados obtidos, com suas respectivas comparações e discussões. Por fim, as conclusões são apresentadas.

2. MÉTODOS DE ARMAZENAMENTO DE MATRIZES E MÉTODOS DE RESOLUÇÃO DO SISTEMA DE EQUAÇÕES

O trabalho propõe o uso de cinco técnicas de armazenamento de matrizes para auxiliar e otimizar a resolução do sistema: CSR, CSC, CSV, Skyline e o método proposto em Jiang et al. (2013), chamado de DFA2. Em associação a estas técnicas, são empregados os métodos iterativos de Jacobi e Gauss-Seidel para a resolução do sistema de equações.

2.1 Métodos de Armazenamento de Matrizes

Lidar com matrizes de grande dimensão exige um uso dispendioso da memória e demanda mais tempo de processamento por parte dos *softwares* que executam a SDTR. Quando estas matrizes são esparsas, várias operações desnecessárias são realizadas, pois há muitos elementos nulos, prejudicando a eficiência computacional. Dessa forma, faz-se necessário utilizar técnicas de armazenamento com o intuito de otimizar os processos de simulação e viabilizar a análise em tempo real.

Os métodos de armazenamento a seguir possuem características comuns: possuem uma matriz que armazena os termos não nulos da matriz da rede (denominada de AN, AA ou VAL), uma matriz que guarda a quantidade desses termos (IA, Rowptr ou #NZ) e uma ou mais matrizes que reúnem índices dos elementos diferentes de zero; estes índices podem ser o número da linha ou coluna, por exemplo.

CSR (Compressed Sparse Row): O método CSR é o mais difundido e utilizado. Descrito em Brayton et al. (1970) e exposto em Rose (1972), ele armazena a matriz esparsa em três matrizes linha, denominadas AN, JA e IA, desprezando os elementos nulos. A formação de IA pode ser descrita pela equação:

$$ia_j = \sum_{j=1}^{j} ia_{j-1}, \tag{1}$$

onde ia_j representa o número de elementos não nulos da coluna j e $ia_1 = 0$.

CSC (Compressed Sparse Column): Trata-se de uma adaptação do método CSR. Difere-se deste na formação das matrizes de armazenamento, pois, a matriz esparsa é percorrida por colunas, ao invés de linhas.

O OpenDSS utiliza os método CSR e CSC, através do *software* KLUSolve (Reynolds (2020)). Ele é uma biblioteca de funções relacionadas a matrizes complexas esparsas adaptado a sistemas elétricos de potência (Radatz (2018)) e é responsável por montar a matriz de admitância do sistema, armazenar e efetuar operações.

CSV (Compressed Sparse Vector): Proposto em Farzaneh et al. (2009), armazena a matriz esparsa em duas matrizes linha: em AA, onde estão os elementos não nulos da matriz e, em IA, em que se encontram os índices referentes aos elementos de AA, percorridos por linha, sendo estes referentes a contagem dos números.

Método Skyline: Apresentado em Coelho (2014), consiste no armazenamento da matriz esparsa em três matrizes linha, as quais são denominadas VAL, Rowptr, cujo preenchimento segue (1), e Fstcol.

Método DFA2: O método apresentado em Jiang et al. (2013) propõe a criação de cinco matrizes para armazenar a matriz esparsa: VAL, ROWID, INDEX, OFFSET e #NZ, sendo estas quatro últimas matrizes linha.

2.2 Métodos de Resolução do Sistema

A matriz de admitância da rede elétrica, que será descrita na Seção 3, foi alvo das técnicas de armazenamento, o que originou as respectivas matrizes para cada técnica. Por meio dos métodos iterativos de Jacobi e Gauss-Seidel, junto à matriz de correntes injetadas I e as correntes de compensação, os algoritmos construídos obtiveram a matriz de tensões nodais V. O uso desses métodos iterativos se deve ao fato de serem indicados para resolução de sistemas de equações lineares grandes e esparsos; além disso, sua resolução por etapas permite o devido acompanhamento dos valores das tensões para cada passo de tempo estipulado.

As equações provenientes dos métodos de Jacobi e Gauss-Seidel para o cálculo do vetor de tensões nodais estão dispostas em (2) e (3), respectivamente:

$$V_i^{k+1} = \frac{1}{Y_{i,i}} \left(I_i^k - \sum_{j=1, j\neq i}^{n} Y_{i,j} V_j^k \right) \tag{2}$$

$$V_i^{k+1} = \frac{1}{Y_{i,i}} \left(I_i^k - \sum_{j=1}^{i-1} Y_{i,j} V_j^{k+1} - \sum_{j=i+1}^{n} Y_{i,j} V_j^k \right) \tag{3}$$

onde V_j^k representa o vetor de tensões nodais atual, V_i^{k+1}, o vetor de tensões nodais a ser calculado, I_i^k, o vetor de correntes e $Y_{i,j}$, a matriz de admitâncias nodais do sistema.

Visto que os métodos de armazenamento já foram executados, a matriz Y presente nessas equações será substituída pelas matrizes que guardaram os elementos não nulos. Devido às características específicas de cada técnica, essa substituição é diferente entre elas, necessitando de estruturas auxiliares.

As correntes de compensação são calculadas a cada passo de iteração para as cargas presentes na rede. Elas são obtidas a partir da seguinte equação:

$$\begin{bmatrix} I_{C1} \\ I_{C2} \\ I_{C3} \end{bmatrix} = \begin{bmatrix} I_{i1} \\ I_{i2} \\ I_{i3} \end{bmatrix} - \begin{bmatrix} Y_1 & Y_2 & Y_3 \\ Y_4 & Y_5 & Y_6 \\ Y_7 & Y_8 & Y_9 \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}, \tag{4}$$

onde

$$\begin{bmatrix} I_{i1} \\ I_{i2} \\ I_{i3} \end{bmatrix} = - \begin{bmatrix} \frac{S_1^*}{3V_1^*} \\ \frac{S_2^*}{3V_2^*} \\ \frac{S_3^*}{3V_3^*} \end{bmatrix} \tag{5}$$

Em (4), temos que o cálculo das correntes de compensação envolve os vetores de correntes injetadas e tensão nodal da carga, e a matriz de admitância da carga. Estas correntes injetadas são obtidas em (5), por meio da potência e da tensão nodal. Após obter os valores, estas correntes de compensação são adicionadas ao vetor I.

Subsequente a esses processos, o algoritmo verifica se os valores do vetor de tensões nodais obedecem à tolerância estabelecida. Caso isso não ocorra, todo o processo se repete até que se tenha êxito nessa comparação ou se atinja o limite de iterações.

3. DESCRIÇÃO DA REDE ELÉTRICA

Nesta seção, descreve-se a rede elétrica escolhida, sua matriz de admitâncias e são apresentados os resultados alcançados.

A rede elétrica da Figura 1 é formada por dezoito barras, oito linhas de transmissão, nove transformadores, sete cargas e duas unidades de geração distribuída.

As barras foram designadas por números que representam múltiplos de dez. Na Figura 1, elas estão identificadas pela letra B acompanhada pelo número correspondente abaixo. Seguindo os parâmetros adotados pelo OpenDSS, não há diferenças entre as barras utilizadas nesta rede. Com relação as tensões, as barras podem ser divididas conforme a Tabela 1.

Tabela 1. Barras.

Barras	Tensão (kV)
10. 20	230
30, 40, 50, 70, 90, 110, 130, 150, 170	69
60, 80, 100, 120, 140, 160, 180	13,8

As linhas de transmissão foram numeradas de um a oito e encontram-se, no diagrama unifilar, acompanhadas pelas letras LT e o número correlato. A Tabela 2 lista estas linhas de acordo com o cabo que possuem.

Tabela 2. Linhas de Transmissão.

Linhas de Transmissão	Cabo Utilizado
1	336,4 CAA - 26x7f
2, 3, 4	336,4 CA - 19f
5, 6, 7, 8	266,8 CA - 7f

Os transformadores foram numerados de um a nove. Identificados, na figura que ilustra a rede, pelas letras TR, com o respectivo número abaixo, possuem o mesmo tipo de ligação: delta - estrela aterrado. Os transformadores 1 e 2 (230:69kV) estão localizados após a linhas de transmissão um (LT 1). Os demais transformadores são 69:13,8kV e se encontram após as demais linhas de transmissão.

As cargas foram numeradas de um a sete e portam o mesmo tipo de ligação: estrela aterrado. Elas estão conectadas às barras onde são ligados os secundários dos transformadores de 69:13,8kV e tem identificação formada pelas letras CG com o número correspondente.

As unidades de geração distribuída possuem ligação em estrela e são conectadas à rede por meio de duas chaves seccionadoras: S2 e S3, conectadas às barras 60 e 100, respectivamente. A unidade geradora 1 é uma Pequena Central Hidrelétrica (PCH) e tem potência nominal de 10 MW. A unidade geradora 2 é uma Usina Termelétrica (UTE) e tem potência nominal de 20 MW.

Para a rede elétrica escolhida, a matriz de admitância possui dimensão 54 x 54. Dos 2916 termos, 468 são diferentes de zero e 2448, nulos. Entre as 54 linhas que compõem a matriz, considerando os valores diferentes de zero, existem três linhas com 18 termos, com 15 e com 12 termos, vinte e uma com 9 e vinte e quatro com 6 elementos.

Além disso, calculando os valores do grau e índice de esparsidade, obteve-se 78% e 84%, respectivamente, de acordo com Monticelli (1983) e Maliska (2017).

4. RESULTADOS OBTIDOS

A partir dos conceitos abordados na Seção 2, esta subseção abordará o uso e a comparação entre os algoritmos construídos, os parâmetros utilizados e os resultados alcançados. Serão considerados dez métodos:

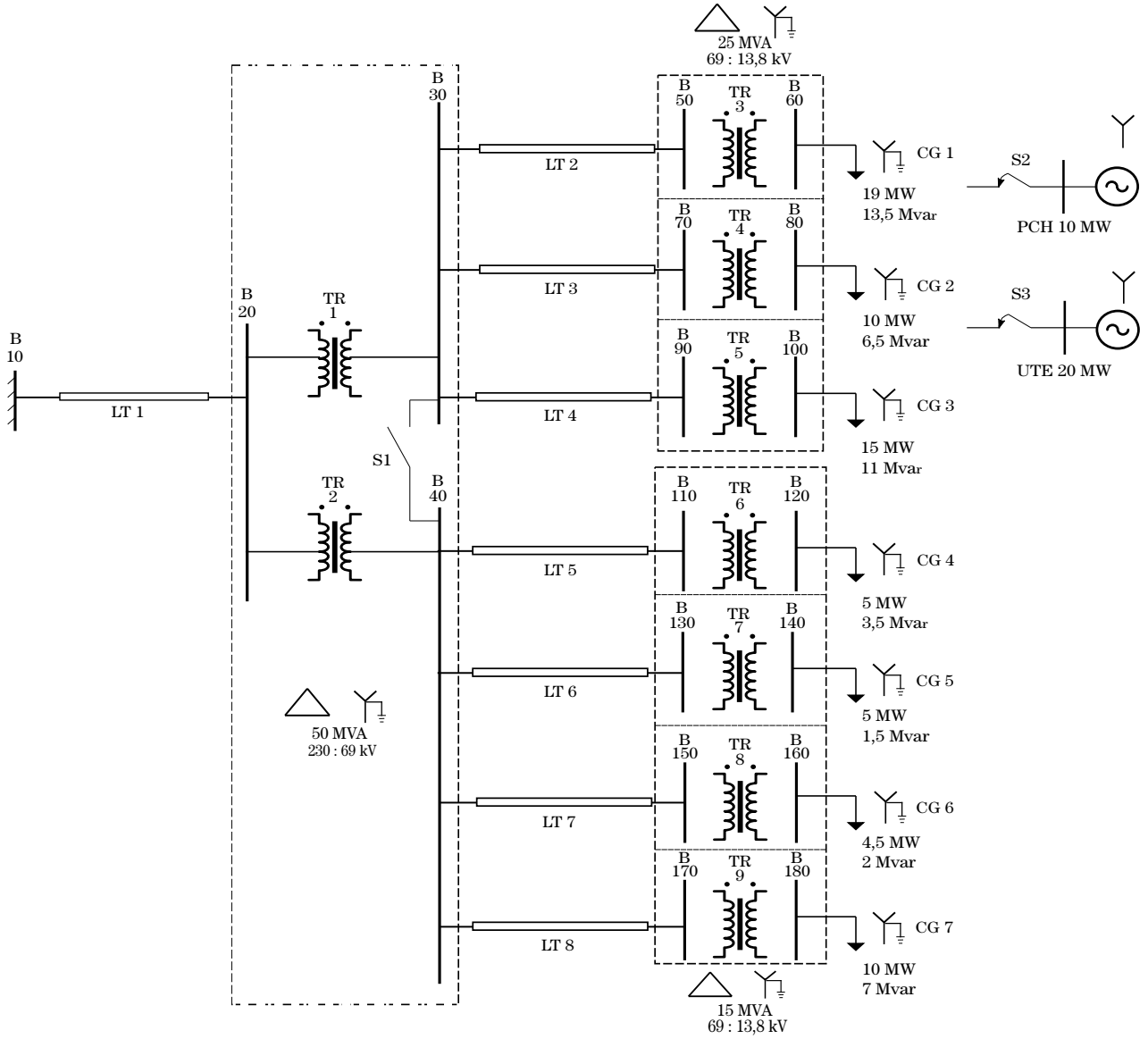

Figura 1. Diagrama Unifilar da Rede Elétrica.

- Método 1: CSR com Método de Jacobi;
- Método 2: CSR com Método de Gauss-Seidel;
- Método 3: CSC com Método de Jacobi;
- Método 4: CSC com Método de Gauss-Seidel;
- Método 5: CSV com Método de Jacobi;
- Método 6: CSV com Método de Gauss-Seidel;
- Método 7: Skyline com Método de Jacobi;
- Método 8: Skyline com Método de Gauss-Seidel;
- Método 9: DFA2 com Método de Jacobi;
- Método 10: DFA2 com Método de Gauss-Seidel;

Como referência, utilizou-se o algoritmo que emprega o método CSR para armazenar a matriz de admitâncias da rede e um método de ponto fixo iterativo para obter a solução do sistema de equações. Por meio dos seus resultados, estabelece-se um parâmetro de tempo de processamento e uso da memória.

A resolução do sistema de equações foi dividida em duas configurações: sistema equilibrado e desequilibrado, cujo desequilíbrio está localizado nas cargas 2 e 5. Estas, possuem as seguintes impedâncias (em *ohms*):

- Carga 2:
  $Z_a = 39.27960777 + 27.41733488j$
  $Z_b = 40.23764698 + 25.99095234j$
  $Z_c = 41.19568620 + 24.44408471j$

- Carga 5:
  $Z_a = 107.25616149 + 21.77929250j$
  $Z_b = 102.87835898 + 37.33985823j$
  $Z_c = 105.06726024 + 30.64461757j$

Para estes arranjos, serão consideradas duas composições da rede: numa delas, as unidades de geração distribuída estarão conectadas à rede. A outra configuração não contará com essa conexão.

A utilização dos métodos iterativos necessita da adoção de alguns parâmetros para o devido funcionamento. Além da condição inicial, foram estabelecidas a tolerância de 10^{-1} e o número máximo de 50 iterações para todos os métodos. A escolha desta tolerância advém do fato desse estudo não estipular uma tensão de base (Radatz (2018)).

A exibição dos valores obtidos através da execução dos códigos construídos será dividida em duas partes: configuração 1, cujos resultados referem-se ao sistema equilibrado, e configuração 2, com resultados relativos ao sistema desequilibrado.

Todos os valores apresentados foram obtidos através do uso de um notebook com as seguintes especificações:

- Modelo: Acer Aspire A515-54G;
- Sistema Operacional: Windows 10, 64 bits;
- Memória RAM DDR4 8 GB;
- Processador Intel(R) Core(TM) i5-10210U 1.60GHz, memória cache de 6 MB, com 4 núcleos físicos e 8 threads;
- Armazenamento de 256 GB, SSD.

Configuração 1: Considerando o sistema equilibrado, foram obtidos, através da utilização do método de Jacobi em associação aos métodos de armazenamento, o tempo de processamento em segundos, presente na Figura 2, e o uso do HD em Megabytes, exposto na Figura 3. Para todas as figuras desta subseção, são apresentados resultados sem e com unidades de geração distribuída (GD) conectadas à rede.

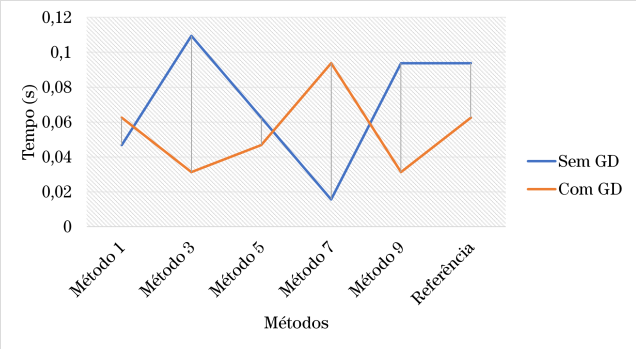

Figura 2. Tempo de processamento para o sistema equilibrado - Método de Jacobi.

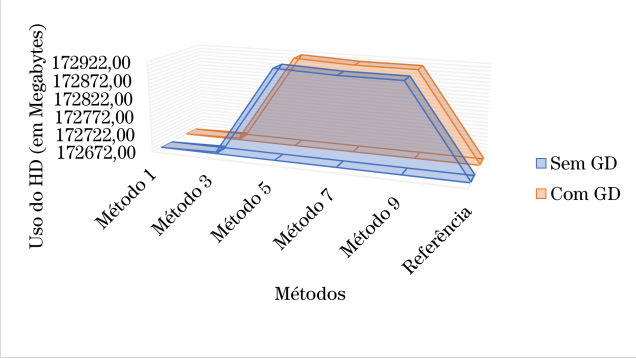

Figura 3. Uso do HD em Megabytes para o sistema equilibrado - Método de Jacobi.

A partir da análise da Figura 2, observa-se que o Método 1 apresenta a menor amplitude entre os tempos registrados para os dois arranjos da rede, sendo o valor gasto para processamento sem a inclusão das unidades de geração distribuída correspondente a $50,0\%$ do tempo gasto pela referência. Percebe-se também que, com a presença das unidades de geração, os tempos aferidos com o uso do algoritmo de referência e o Método 1 são os mesmos, o que indica que essa composição pode ter rendimento próximo ao parâmetro.

Na Figura 3, constata-se que os Métodos 1 e 3 apresentam taxa de uso do HD muito próxima ao algoritmo de referência. Isto se deve ao fato dos demais métodos de armazenamento necessitarem de mais laços de repetição ou possuírem mais estruturas auxiliares para a obtenção dos resultados. Essas estruturas, em sua maioria, caracterizam-se por serem matrizes que dão suporte aos produtos do emprego das técnicas de armazenamento, guardando índices necessários ao funcionamento dos códigos construídos.

Para o método de Gauss-Seidel em associação aos métodos de armazenamento, o tempo de processamento em segundos e o uso do HD em Megabytes são exibidos nas Figuras 4 e 5, respectivamente.

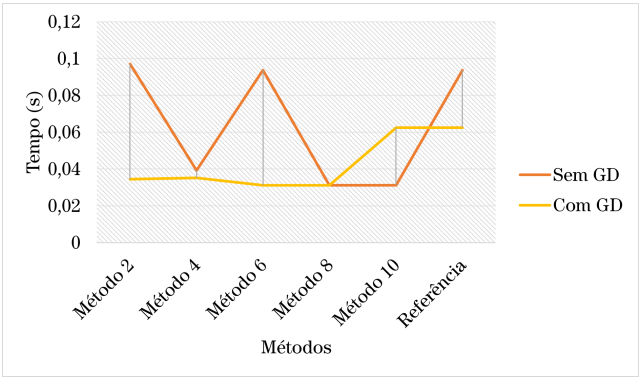

Figura 4. Tempo de processamento para o sistema equilibrado - Método de Gauss-Seidel.

Com a aplicação do método de Gauss-Seidel, foi possível notar na Figura 4 que os valores obtidos com a inclusão das unidades de geração mostraram-se melhores que o aferido para o algoritmo de referência, com exceção do Método 10. Do ponto de vista numérico, essa redução no tempo de execução era esperada, visto que, devido a sua constituição, a técnica proposta por Seidel obtém as soluções com melhor eficiência.

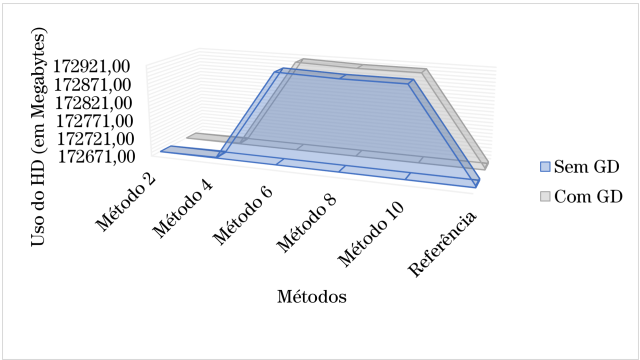

Figura 5. Uso do HD em Megabytes para o sistema equilibrado - Método de Gauss-Seidel.

O gráfico presente na Figura 5 mostra que os valores de uso da memória seguem o mesmo padrão, seja com as unidades de geração distribuída ou não.

Configuração 2: Para o sistema desequilibrado, foram obtidos, através da utilização do método de Jacobi em associação aos métodos de armazenamento, os dados necessários para comparação. A Figura 6 exibe o tempo de processamento, em segundos e a Figura 7, o uso do HD, em Megabytes.

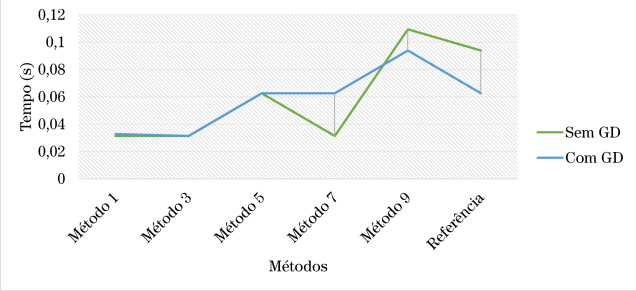

Figura 6. Tempo de processamento para o sistema desequilibrado - Método de Jacobi.

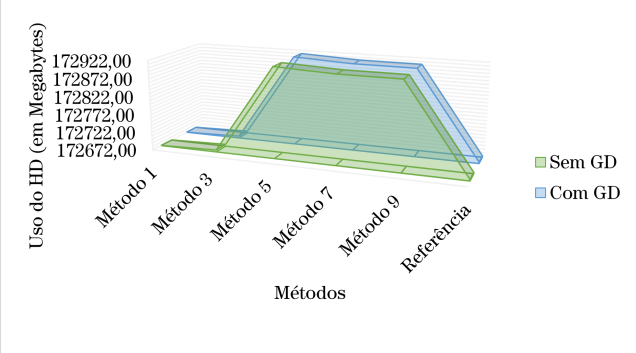

Figura 7. Uso do HD em Megabytes para o sistema desequilibrado - Método de Jacobi.

Assim como foi observado na configuração 1, o Método 1 apresenta, na Figura 6, o menor valor de tempo de processamento para a rede sem a presença das unidades de geração, que corresponde a $33,3\%$ do tempo gasto pelo método de referência. O Método 3 repete o desempenho na configuração anterior com relação ao uso da memória e destaca-se com relação ao tempo de processamento, que pode ser observado na Tabela 4.

As Figuras 8 e 9 exibem o tempo de processamento em segundos, e o uso do HD em Megabytes para o método de Gauss-Seidel em associação aos métodos de armazenamento.

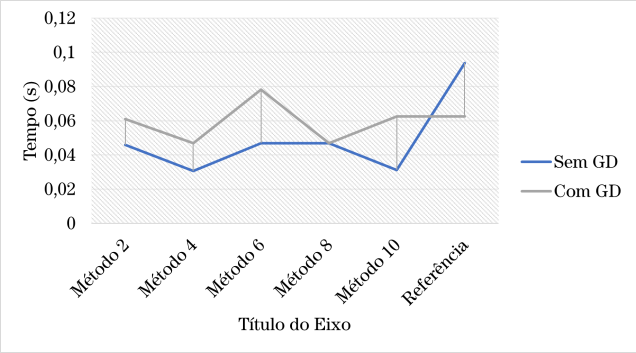

Figura 8. Tempo de processamento para o sistema desequilibrado - Método de Gauss-Seidel.

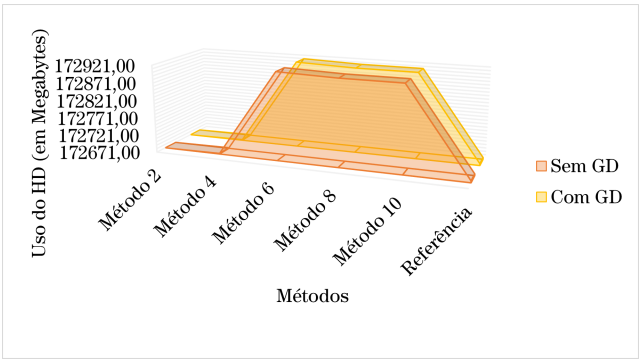

Figura 9. Uso do HD em Megabytes para o sistema desequilibrado - Método de Gauss-Seidel.

Os Métodos 2 e 4, que aparecem nas Figuras 8 e 9, apresentam melhores resultados com relação aos demais, principalmente, em relação ao uso da memória para processamento. Dentre as características que possibilitam esses desempenhos, destacam-se a necessidade de menor utilização de matrizes auxiliares para execução dos métodos iterativos e menos laços de repetição em seu algoritmo do que as demais técnicas. Este atributo da estrutura do algoritmo é um importante componente para o êxito da SDTR.

As Tabelas 3 e 4 congregam os valores de tempo (em segundos) e uso da memória (em Megabytes) para a resolução do sistema de equações conforme as configurações 1 e 2, respectivamente.

Tabela 3. Configuração 1.

Métodos	Tempo (s)		Memória (Megabytes)	
	Sem GD	Com GD	Sem GD	Com GD
Método 1	0,046875	0,062500	172672,88	172673,03
Método 2	0,097068	0,034394	172671,21	172671,22
Método 3	0,109375	0,031250	172677,35	172677,44
Método 4	0,039226	0,035134	172673,07	172673,09
Método 5	0,062500	0,046875	172920,21	172921,06
Método 6	0,093750	0,031250	172918,90	172918,92
Método 7	0,015625	0,093750	172913,48	172913,56
Método 8	0,031250	0,031250	172913,58	172913,60
Método 9	0,093750	0,031250	172913,15	172913,18
Método 10	0,031250	0,062500	172913,30	172913,45
Referência	0,093750	0,062500	172675,71	172675,87

Tabela 4. Configuração 2.

Métodos	Tempo (s)		Memória (Megabytes)	
	Sem GD	Com GD	Sem GD	Com GD
Método 1	0,031250	0,032813	172672,95	172672,97
Método 2	0,045879	0,060939	172671,21	172671,23
Método 3	0,031250	0,031250	172677,43	172677,43
Método 4	0,030555	0,046875	172673,01	172673,12
Método 5	0,062500	0,062500	172921,07	172921,07
Método 6	0,046875	0,078125	172918,91	172919,00
Método 7	0,031250	0,062500	172913,49	172913,57
Método 8	0,046875	0,046875	172913,59	172913,60
Método 9	0,109375	0,093750	172913,16	172913,18
Método 10	0,031250	0,062500	172913,31	172913,40
Referência	0,093750	0,062500	172675,79	172675,90

5. CONCLUSÃO

Este trabalho descreveu os conceitos e a aplicação de métodos de armazenamento de matrizes em conjunto com

os métodos iterativos de Jacobi e Gauss-Seidel com o intuito de viabilizar a implementação destas técnicas para a construção de um simulador de uma rede elétrica em tempo real. A partir da utilização conjunta dos métodos iterativos e de armazenamento, foi possível constatar a complexidade de operações com matrizes esparsas e a necessidade de utilizar técnicas para mitigar a esparsidade e otimizar os processos computacionais.

Sobre os resultados obtidos, notou-se que o método CSR, associado aos métodos iterativos de Jacobi e Gauss-Seidel (Métodos 1 e 2), apresentou o melhor desempenho, principalmente para a configuração 2, onde tratou-se do desequilíbrio do sistema. Isto se deve ao fato das matrizes geradas por esse método apresentarem fácil adaptação às estruturas dos métodos iterativos, necessitando de menor incremento computacional em relação as demais técnicas, como matrizes auxiliares e laços de repetição. Estas características proporcionam maior eficiência computacional, além de facilitar a adaptação a outras linguagens de programação, caso necessário.

Com relação aos valores de uso do HD, observa-se que há pequenas alterações entre os valores obtidos. Isto se deve ao fato de não serem necessárias mudanças nos algoritmos para efetuar os cálculos inerentes ao processo e dos pequenos incrementos nas matrizes das diferentes configurações não exigirem maiores esforços computacionais.

Por fim, observa-se que a associação dos métodos de armazenamento de matrizes aos métodos iterativos propostos apresentou vantagens em relação à referência estabelecida, principalmente com relação ao tempo de processamento, cujos ganhos percentuais foram destacados na Seção 4. Para a simulação em tempo real, estes dados são importantes, pois a rapidez do processamento das informações e da execução dos cálculos necessários são essenciais para o funcionamento correto do simulador, que deve apresentar os resultados necessários, de forma a reproduzir as condições da rede real em estudo. Presume-se que estas técnicas apresentarão ganhos para redes elétricas de maior porte, que, por consequência, terão matrizes de admitâncias nodais esparsas e de maior ordem. Salienta-se que os algoritmos utilizados para este trabalho podem ser aperfeiçoados, proporcionando mais eficiência no emprego em simulações em tempo real.

AGRADECIMENTOS

Os autores querem expressar sua gratidão ao Programa de Pós-Graduação em Modelagem Matemática e Computacional da Universidade Federal da Paraíba pelo suporte e estrutura.

REFERÊNCIAS

Andrade, Vinicius B.; Paixão Jr., Ulisses C.; Moreira, Carlos E.; Soares, Thiago M.; Tabora, Jonathan M.; Tostes, Maria Emília de L.; Bezerra, Ubiratan H.; Albuquerque, Bruno S. and Gouveia, Luciano Da S. Modelagem de um sistema de distribuição real desbalanceado e análise do impacto da geração distribuída utilizando o software OpenDSS. *Anais do Simpósio Brasileiro de Sistemas Elétricos*, 2020.

Arrillaga, J., Watson, N.R. Computer Modelling of Electrical Power Systems. *John Wiley and Sons, LTD*, 2001.

Brayton, R.K., Gustavson, F.G., Willoughby, R.A. Some results on sparse matrices. *Mathematics of Computation*, 24, 937-954, 1970.

Coelho, Marco Antonio de Oliveira. Métodos Iterativos para Resolver Sistemas de Equações Algébricas Lineares em Estruturas Esparsas. *Universidade Federal Fluminense*, 2014.

Cunha, Maria Cristina C. Métodos Numéricos. *Editora da Unicamp*, 2018.

Farzaneh, A., Kheiri, H., Abbaspour, M. An efficient storage format for large sparse matrices. *Communications de la Faculté des Sciences de l'Université d'Ankara. Séries A1: Mathematics and Statistics*, 58, 2009.

Filho, Milton Brown do C., do Coutto, Felipe Azevedo Brown. Métodos Numéricos: fundamentos e implementação computacional. *Elsevier*, 2017.

Jiang, L., Tan, J. and Tang, Q. An efficient sparse matrix format for accelerating regular expression matching on field-programmable gate arrays. *Security and Communication Networks*, 8, 2013.

Kersting, W. Distribution System Modeling and Analysis. *CRC Press*, 2002.

Maliska, Clovis R. Transferência de Calor e Mecânica dos Fluidos Computacional. *LTC*, 2017.

Monticelli, Alcir José Fluxo de Carga em Redes de Energia Elétrica. *Edgar Blucher*, 1983.

Oliveira, Thays Rolim Mendes de. Avaliação de formatos de armazenamento com compressão para resolução de sistemas de equações lineares esparsos. *Universidade Tecnológica Federal do Paraná*, 2019.

Paulilo, Gilson. Qualidade de Energia - Capítulo III: Desequilíbrios de Tensão. *Revista O Setor Elétrico*, 2013.

Python Software Foundation. Time access and conversions *Disponível em: https://docs.python.org/3/library/time.html; Último acesso em 20 de Abril de 2022*

Reynolds, Daniel R.; Gardner, David J.; Woodward, Carol S. and Balos, Cody J. SUNLinSol_KLU Usage. *Disponível em: http://runge.math.smu.edu/arkode_dev/doc/guide/build/html/sunlinsol/SUNLinSol_KLU.html*; Último acesso em 11 de Dezembro de 2021.

Rocha, Celso and Radatz, Paulo Nota Técnica - Algoritmo de fluxo de potência do OpenDSS. *Grupo de Usuários do OpenDSS Brasil*, 2018

Rodola, Giampaolo Psutil Documentation *Disponível em: https://psutil.readthedocs.io/en/latest/; Último acesso em 12 de Dezembro de 2021*

Rose, Donald J. and Willoughby, Ralph A. Sparse Matrices and their Applications. *Springer*, 1972.

Stevenson, William D. Elementos de Análise de Sistemas de Potência. *McGraw-Hill*, 1986.

The Numpy Comunity. Numpy.Ndarray.Nbytes *Disponível em: https://numpy.org/doc/stable/reference/generated/numpy.ndarray.nbytes.html; Último acesso em 12 de Dezembro de 2021*