PUC Minas
A B A K Ó S
Instituto de Ciências Exatas e Informática



# Relatório Técnico sobre Métodos de Normalização*

Technical Report on Normalization Methods

Bernardo Alvim, [1]
Marcos Alberto, [2]
Elektra El Awar, [3]
Orientador: Prof. Ricardo Carlini Sperandio, [4]

**Resumo**

Este relatório técnico tem como objetivo analisar o desempenho de diferentes métodos de normalização de raízes quadradas, incluindo *Lookup Table*, *Quake*, *SSE* e o método *Default*. A comparação é feita em termos de tempo de usuário, tempo de sistema e uso de memória (tamanho máximo do conjunto residente), com diferentes tamanhos de datasets variando de 50.000 a 250.000 linhas de dados. A análise é apresentada através de gráficos explicativos e uma discussão detalhada de cada método, permitindo a escolha do método mais adequado para diferentes contextos de uso.

**Palavras-chave:** Normalização. Benchmarking. Desempenho. Eficiência.

**Abstract**

This technical report aims to analyze the performance of different square root inversion normalization methods, including *Lookup Table*, *Quake*, *SSE*, and the *Default* method. The comparison is made in terms of user time, system time, and memory usage (maximum resident set size), with different dataset sizes ranging from 50,000 to 250,000 lines of data. The analysis is presented through explanatory graphs and a detailed discussion of each method, allowing for the selection of the most suitable method for different use contexts.

**Keywords:** Normalization. Benchmarking. Performance. Efficiency.

*Relatório apresentado como parte do curso de Engenharia de Software da PUC Minas

[1]Estudante de Engenharia de Software, Programa de Engenharia de Software da PUC Minas, Belo Horizonte - MG– 1528669@sga.pucminas.br
[2]Estudante de Engenharia de Software, Programa de Engenharia de Software da PUC Minas, Belo Horizonte - MG– 1523749@sga.pucminas.br
[3]Estudante de Engenharia de Software, Programa de Engenharia de Software da PUC Minas, Belo Horizonte - MG– 1523578@sga.pucminas.br
[4]Programa de Engenharia de Software da PUC Minas, Belo Horizonte - MG– 754128@sga.pucminas.br

# 1 INTRODUÇÃO

O benchmarking é uma prática essencial para avaliar o desempenho de algoritmos e métodos em ciência da computação. Neste relatório, comparamos o desempenho de diferentes métodos de benchmarking em termos de tempo de execução e uso de memória, utilizando dados de diferentes tamanhos de datasets. Os métodos analisados incluem:

- **Lookup Table:** Uma tabela de pesquisa pré-calculada que visa otimizar o tempo de execução de funções repetidas.
- **Quake III:** Método baseado no algoritmo Quake, comumente usado para otimizar o uso de memória e o tempo de execução.
- **SSE (Streaming SIMD Extensions):** Instruções que melhoram o desempenho computacional, utilizando paralelismo a nível de hardware.
- **Default:** O método padrão sem otimizações específicas.

A análise se baseia em gráficos com escala logarítmica para melhor visualização das diferenças de desempenho.

# 2 METODOLOGIA

Os testes foram conduzidos utilizando datasets de diferentes tamanhos, coletados por meio de uma série de benchmarks. As medições realizadas incluem o tempo de usuário, o tempo de sistema e o uso de memória (tamanho máximo do conjunto residente).

Para executar a bateria de testes, o código base fornecido pelo professor foi modificado para registrar os dados de cada execução em arquivos CSV. Posteriormente, foi desenvolvido um *shell script* que automatiza a execução de cada método de benchmark 10 vezes para cada tamanho de dataset, garantindo maior precisão nos resultados.

Os gráficos apresentados a seguir foram gerados utilizando a plataforma *Jupyter Notebook*. Esta ferramenta foi empregada para calcular a média dos resultados de cada método, presentes nos arquivos CSV, e para criar visualizações gráficas que facilitam a interpretação dos dados. Esses gráficos apresentam os resultados médios obtidos para cada método de *benchmark*.

# 3 RESULTADOS E DISCUSSÃO

## 3.1 Tempo de Usuário

O gráfico da Figura 1 mostra o tempo de usuário médio para cada método em diferentes tamanhos de dataset.

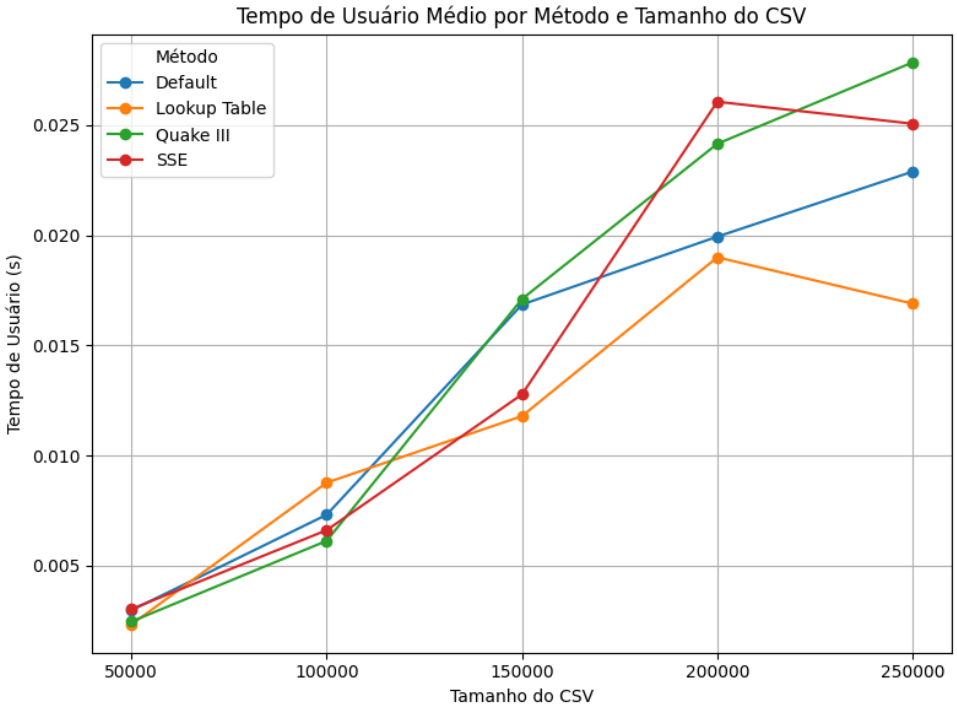


**Figura 1 – Tempo de Usuário Médio por Método e Tamanho do CSV**

Um método *Lookup Table*, que opera procurando os valores pré-computados, é perceptivelmente rápido com volumes menores, mas seu desempenho diminui à medida que os volumes aumentam, embora tente se recuperar com tempos de processamento menores conforme o volume continua a crescer. O outro método de otimização pelo *Quake III* é rápido com volumes menores, mas ficando mais lento com volumes maiores através da iteração extra para maior precisão e lento após 150.000 linhas. Enquanto isso, o *SSE*, embora muito lento no início, superou os outros métodos em volume entre 100.000-150.000 linhas, permitindo tirar vantagem da capacidade de o processador processar grandes quantidades de dados, porém, em volumes maiores, esse método ocorre o aumento de tempo do usuário. Infelizmente, ele é hardware-dependente, o que pode limitar a portabilidade do código. O método *Default* é lento no início comparado com os outros métodos, com ligeiras variações ao longo dos intervalos até o final e apresenta um dos maiores em tempo de usuário. Portanto, a escolha correta é determinada diretamente pelo tamanho do conjunto de dados e recursos disponíveis. Assim, *Lookup Table* é o mais eficaz em grandes volumes, e o *SSE* e *Quake III* são mais rápidos em comparação com o default em volumes menores.

## 3.2 Tempo de Sistema

O gráfico da Figura 2 mostra o tempo médio de sistema para cada método em diferentes tamanhos de dataset.

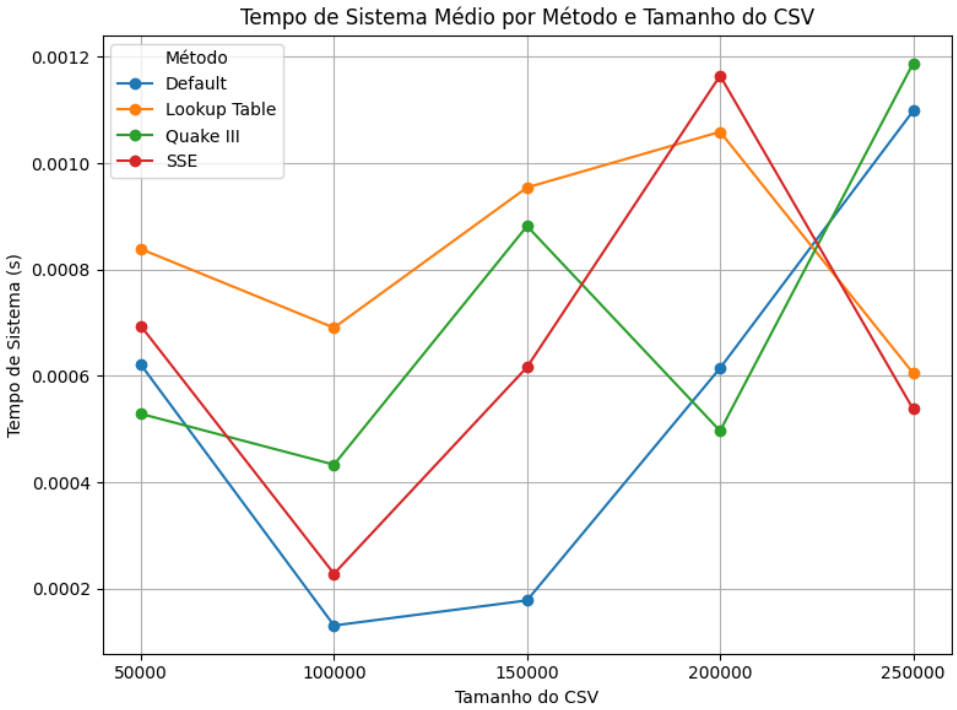


**Figura 2 – Tempo de Sistema Médio por Método e Tamanho do CSV**

O método *Lookup Table* oferece respostas em tempo recorde para cada consulta, mas ocupa grande parte da memória e tem desempenho mais lento quando se trata de pequenos datasets, melhorando apenas acima de 200.000 linhas. O método *Quake III* fornece o equilíbrio preciso da velocidade, fornecendo a estimativa inicial eficiente, exigindo menos iterações de refinamento e mantendo-se razoavelmente estável neste momento, entre 50.000 e 100.000 linhas, mas mostra ampla variação acima desse limite. A abordagem *SSE* apresenta velocidade inicial rápida e tempo do sistema reduzido até 100.000 linhas, mas mostra variações elevadas com o crescimento do dataset, o que pode ser causado por sistemas de cache. O método *Default* é simples, eficiente e mais rápido inicialmente, com uma redução drástica no tempo até 100.000, mas perde a eficiência após esse limite. Observa-se, assim, que cada método terá diferentes comportamentos em função trade-off em memória, velocidade e precisão, sendo mais adequado em função do volume de dados e dos requisitos de desempenho.

### 3.3 Uso de Memória

O gráfico da Figura 3 mostra o tamanho de recursos usados para cada método em diferentes tamanhos de dataset.

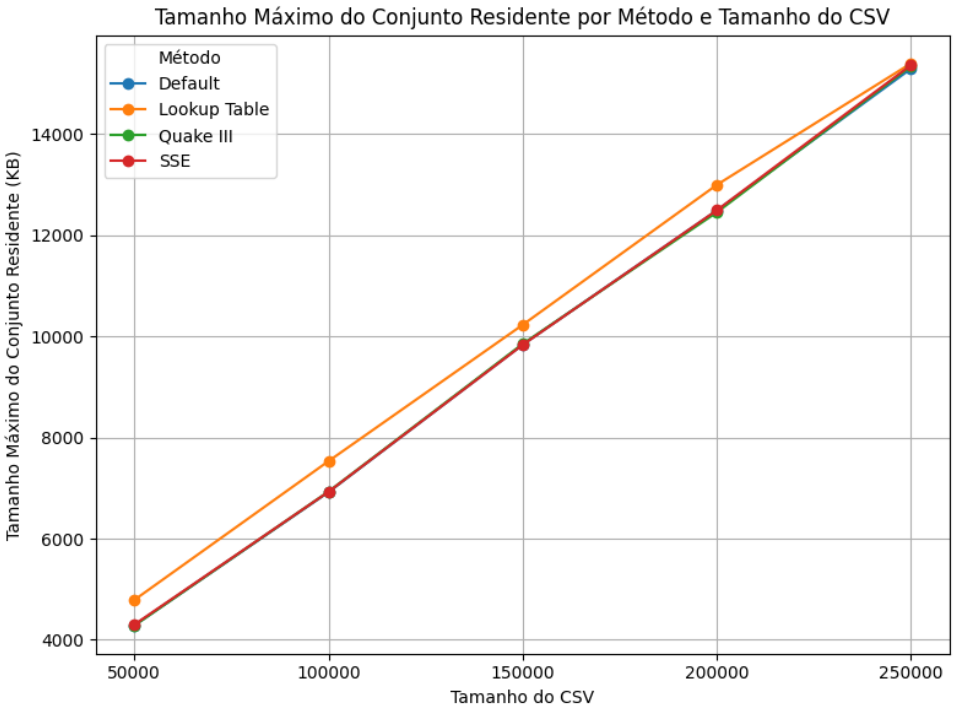


**Figura 3 – Uso de Memória por Método e Tamanho do CSV**

No gráfico todos os métodos mostram um crescimento linear com pequenas flutuações em torno de 250000 linhas. A diferença entre os métodos *Quake III*, *SSE* e *Default* em questão de uso de memória é quase inexistente, enquanto o *Lookup Table* definitivamente consome mais memória, uma vez que armazena valores.

## 4 CONCLUSÃO

Neste relatório, analisamos o desempenho de quatro métodos de benchmarking — *Lookup Table*, *Quake III*, *SSE* e *Default*. Os resultados mostram que cada método possui características distintas, adequadas a diferentes contextos.

O *Lookup Table* se destacou pela rapidez em acessar valores pré-calculados, mas com alto custo de memória, ideal para aplicações que priorizam velocidade. O *Quake III* apresentou um bom equilíbrio entre desempenho e precisão, especialmente em volumes intermediários, enquanto o SSE aproveitou o paralelismo de hardware, embora seu desempenho varie conforme o tamanho do dataset.

O método *Default* foi fácil de implementar e apresentou desempenho aceitável em volumes menores, mas se mostrou menos eficiente em datasets maiores.

A escolha do método de benchmarking deve considerar o tamanho do dataset, os recursos disponíveis e as necessidades de desempenho. Este relatório fornece uma visão clara das vantagens e desvantagens de cada técnica, ajudando desenvolvedores e pesquisadores a tomar decisões informadas.

## Referências

**Fast Inverse Square Root — A Quake III Algorithm** (2021). YouTube. Disponível em: <https://youtu.be/p8u_k2LIZyo?si=WxIC1ubLg5jKu0Wq>. Acesso em: 2024-10-05.

**Fast Inverse Square Root Algorithm** (2024). YouTube. Disponível em: <https://youtu.be/5_pBTSuxdhY?si=igoZehXyyXbqLfPn>. Acesso em: 2024-10-05.

**Fast Inverse Square Root**. Wikipedia. Disponível em: <https://en.wikipedia.org/wiki/Fast_inverse_square_root>. Acesso em: 2024-10-05.

**Streaming SIMD Extensions**. Wikipedia. Disponível em: <https://en.wikipedia.org/wiki/Streaming_SIMD_Extensions>. Acesso em: 2024-10-05.

**Lookup Table**. Wikipedia. Disponível em: <https://en.wikipedia.org/wiki/Lookup_table>. Acesso em: 2024-10-05.

**SSE Intrinsics**. University of Alaska Fairbanks. Disponível em: <https://www.cs.uaf.edu/2009/fall/cs301/lecture/11_13_sse_intrinsics.html>. Acesso em: 2024-10-05.

# A APÊNDICE A: TUTORIAL SOBRE BENCHMARKING

Este apêndice apresenta um tutorial sobre como executar e analisar benchmarks utilizando os métodos apresentados no relatório. O processo envolve três etapas principais: 1) Preparação do ambiente, 2) Execução dos benchmarks, e 3) Interpretação dos resultados.

## A.1 Etapa 1: Preparação do ambiente

Antes de executar os benchmarks, é necessário configurar o ambiente de desenvolvimento. Para este tutorial, usaremos um sistema Linux. Certifique-se de ter as seguintes ferramentas instaladas:

- Python 3.x
- Jupyter Notebook
- Bibliotecas necessárias: `matplotlib`, `numpy`, `pandas`

  Para instalar as bibliotecas, execute o seguinte comando no terminal:

  ```
  pip install matplotlib numpy pandas
  ```

## A.2 Etapa 2: Execução dos benchmarks

O código de benchmarking foi desenvolvido em C e é executado diretamente no terminal. O código registra os tempos de execução e o uso de memória em um arquivo CSV. Para cada método de benchmark, siga os seguintes passos:

1. Compile o código:

   ```
   gcc -o benchmark benchmark.c -lm -msse -O2
   ```

2. Execute o benchmark para cada método e tamanho de dataset:

   ```
   ./benchmark <método> <arquivo csv>
   ```

3. Ou rode o *shell script* que faz uma bateria de execuções automaticamente:

```
chmod +x benchmark_run.sh
./benchmark_run.sh
```

Os resultados serão salvos em arquivos separados. Consulte os seguintes arquivos para mais detalhes:

- `benchmark.c` - Código-fonte do benchmark.
- `benchmark_run.sh` - Script para executar múltiplos benchmarks.
- `resultados.csv` - Arquivo contendo os resultados das execuções.

### A.3 Etapa 3: Interpretação dos resultados

Agora que os resultados foram gerados, podemos usar o `Jupyter Notebook` para criar gráficos e interpretar os dados a partir do código a seguir.

`benchmark_plots.ipynb` - Arquivo que faz a montagem dos gráficos.