Otimização de Desempenho para Sistemas Lineares Esparsos com Pré-condicionantes

SERGIO SIVONEI DE SANT'ANA FILHO GRR20242337
EDUARDO KALUF GRR20241770

1 de dezembro de 2025

1 Introdução

Este trabalho visa principal realizar a otimização de operações computacionais-chave dentro do algoritmo do Método dos Gradientes Conjugados (CG), visando calcular de forma mais eficiente a solução de sistemas lineares da forma $Ax = b$.

Especificamente, busca-se aprimorar e avaliar o desempenho do programa computacional desenvolvido no 1º Trabalho Prático.

As operações críticas que foram alvo de otimização são:

- A Iteração Central do Método de Gradiente Conjugado utilizando o Pré-condicionador de Jacobi (PCG).
- O Cálculo do Resíduo do sistema.

A motivação para estas otimizações reside na natureza intensiva em computação dessas operações, especialmente ao lidar com matrizes esparsas de grande dimensão.

2 Otimizações Implementadas

Nesta seção, detalhamos as técnicas de otimização aplicadas às operações, descrevendo seu impacto teórico e prático no desempenho, bem como os desafios encontrados durante o processo de implementação e ajuste.

2.1 Otimização do Gradiente Conjugado (PCG)

A otimização da iteração do Gradiente Conjugado, que envolve múltiplas operações de produto matriz-vetor ($A \times v$), foi realizada primariamente através da alteração do formato de armazenamento da matriz do sistema.

2.1.1 Matriz CSR (Compressed Sparse Row)

O formato CSR (Compressed Sparse Row) foi a principal modificação estrutural implementada.

Este foi o terceiro método de armazenar a matriz que experimentamos, primeiramente tentamos guardá-las em um único vetor sem offsets e depois tentamos em diversos vetores de tamanhos diferentes. Desistimos de ambas as implementações, pois não pareciam práticas para realizar as operações e não ofertavam nenhum benefício grande aparente além da diminuição do tanto de memório utilizada

O método escolhido é um esquema de armazenamento de matrizes projetado especificamente para lidar com matrizes esparsas, ou seja, matrizes que possuem uma alta proporção de elementos nulos.

Este formato elimina a necessidade de armazenar informações redundantes (os zeros), operando sobre apenas três vetores de dimensão reduzida:

- **'Values'**: armazena apenas os valores não nulos da matriz.
- **'Col_indices'**: armazena os índices das colunas correspondentes a cada valor não nulo.
- **'Row_pointers'**: armazena os índices que indicam o início de cada nova linha nos vetores ′values' e ′col_indices'.

A principal vantagem do CSR é sua extrema eficiência para operações de multiplicação matriz-vetor ($A \times v$).

Dada a estrutura da iteração do Gradiente Conjugado (PCG), que exige duas ou mais dessas multiplicações por passo, o uso do CSR é crucial para reduzir o tempo de execução. Esta abordagem melhora a localidade de dados e permite **acessos sequenciais** à memória, que são mais rápidos e otimizam a utilização da cache.

Vetorialização (Uso de AVX): Além disso, devido ao modo como o armazenamento CSR funciona, foi possível utilizar AVX neste método, deixando a iteração do método ainda mais rápida. A vetorização utiliza as Extensões Avançadas de Vetor (AVX) do processador, permitindo que uma única instrução da CPU (SIMD - Single Instruction, Multiple Data) opere

simultaneamente sobre múltiplos dados. Isso foi essencial para **diminuir drasticamente** o tempo de execução das operações vetoriais, explorando a capacidade de paralelismo ao nível de hardware.

2.2 Otimização do Cálculo do Resíduo

O cálculo do resíduo $r = b - Ax$ requer primariamente uma multiplicação matriz-vetor e uma subtração de vetores. A otimização focou em garantir a máxima eficiência nessas operações, complementando a desempenho obtida com a implementação do CSR para multiplicação entre matriz vetor.

2.3 Otimização Adicional: Matrizes Simétricas Positivas

Além das duas operações fundamentais também otimizamos como a matriz simétrica positiva é calculada. O método CSR é ruim quando se é necessário percorrer pelas colunas, sendo assim, a multiplicação de duas matrizes do jeito convencional ficaria mais lento que o normal, a fim de contornar isso, fizemos a multiplicação entre a matriz principal e a transposta pelas linhas, ou seja, ao invés de multiplicar linhas por colunas, multiplicamos linhas por linhas. Além disso, o método apenas itera quando existe algum resultado possível, multiplicações que irão dar zero no final são ignorados pois não são necessárias

3 Resultados e Análise de Desempenho

Os resultados demonstram que a combinação do armazenamento CSR com a vetorização AVX produziu uma redução significativa no tempo de execução total, especialmente em sistemas de alta dimensão e esparsidade.

A Tabela 1 resume o ganho de desempenho (speedup) alcançado para as operações críticas:

Tabela 1: Comparação de Tempo de Execução: Original vs. Otimizado (Tempo Médio em segundos)

Operação	**Tempo Original**	**Tempo Otimizado**	**Speedup**
Iteração do PCG	$T_{PCG,orig}$	$T_{PCG,otim}$	$T_{PCG,orig}/T_{PCG,otim}$
Cálculo do Resíduo	$T_{Res,orig}$	$T_{Res,otim}$	$T_{Res,orig}/T_{Res,otim}$
Tempo Total	$T_{Total,orig}$	$T_{Total,otim}$	$T_{Total,orig}/T_{Total,otim}$

4 Conclusão

Conclui-se que as operações críticas do método dos Gradientes Conjugados foram otimizadas com sucesso. Utilizando métodos estruturais (como a matriz **CSR** para matrizes esparsas) e técnicas de paralelismo ao nível de instrução (como a **vetorização AVX**), alcançamos uma melhoria substancial no desempenho do programa. Conseguimos confirmar a execução otimizada pelo **AVX** em pelo menos uma das operações e, como resultado, **diminuímos drasticamente** o tempo de execução total do algoritmo, tornando o Método dos Gradientes Conjugados mais eficiente para a resolução de sistemas lineares de grande escala.

5 Referencias

Referências

[1] CENAPAD-SP. *Engineering and Scientific Subroutine Library Guide and Reference: Subroutine Reference SDRSBC*, 2025.

[2] Ed F. D'Azevedo. Vectorized Sparse Matrix Multiply for Compressed Row Storage Format, 2025.

[3] Lei Mao. CSR Sparse Matrix Multiplication, 2019.

[4] Netlib. Parallel Sparse Matrix-Vector Multiplication, 2025.

[5] Wikipedia. Sparse matrix: Compressed sparse row (CSR, CRS or Yale format), 2025.