INSTITUTO FEDERAL
Paraná
Campus Pinhais

Bacharelado em Ciência da Computação
Análise Numérica — 30 de abril de 2026
Trabalho — v1.2

1 Introdução

Sua tarefa é implementar um programa que resolva sistemas de equações lineares. Você irá implementar um método que não estudamos em sala, mas que é muito conhecido pela comunidade. O objetivo do trabalho é que você possa experimentar com um novo método, exercitar a programação como ferramenta para resolução de problemas numéricos, e também exercitar a capacidade de avaliação de desempenho e otimização de programas. O trabalho deverá ser feito em linguagem C ou C++.

2 Método do Gradiente Conjugado

O método a ser implementado é chamado de Gradiente Conjugado. Apesar de não termos estudado esse método, sua implementação é de dificuldade equivalente aos que estudamos em sala. A ideia do método consiste em tentar minimizar o gradiente de algumas funções relacionadas ao sistema.

Para usar o Método do Gradiente Conjugado, a matriz do sistema a ser resolvida deve ser simétrica e positiva definida ($A^T = A$ e $x^T Ax > 0$ para todo x não-nulo). Esse método tem diversas aplicações no mundo real.

Você deverá escrever o código que gera uma matriz de entrada para o problema aleatoriamente. Lembre-se que você pode usar uma *seed* fixa para iniciar o sistema de números aleatórios (`srand`). Dessa forma seu programa irá gerar sempre a mesma matriz e você pode comparar os resultados, garantindo que sua resposta continua correta. Ao mudar a *seed*, seu programa irá gerar uma matriz diferente e, portanto, um novo caso de teste.

2.1 Matriz esparsa

Em problemas reais, é muito comum que a matriz seja grande, mas esparsa. Isso significa que a maioria dos elementos da matriz é nulo (0). Neste trabalho, você considerará dois casos: matrizes com 7 e 27 bandas. Cada banda é uma das diagonais da matriz. No caso de uma matriz com 7 bandas, teremos uma diagonal principal e, 3 diagonais "para cima" da diagonal principal, e 3 diagonais **simétricas** "para baixo" da diagonal principal.

2.2 Casos de teste

Você deve executar testes para diferentes tamanhos de matriz: 1024, 4096, 16384, 65536, 262144 e 1048576. Note que sua matriz completa não irá caber na memória do seu computador. Por isso, você deverá buscar uma forma de representar apenas as diagonais não-nulas de sua matriz.

3 Análise de desempenho e otimização

Você deve implementar uma primeira versão do trabalho com o menor número de otimizações. Você deve instrumentalizar seu código para medir o tempo gasto pelas principais partes do método implementado, e também o comportamento do hardware, como memória cache. Para monitorar o hardware, sugere-se o uso da aplicação Likwid[1].

Depois, você deverá otimizar seu código com as estratégias discutidas em sala e outras que encontrar. Você deverá repetir os testes para obter os dados de desempenho otimizados e comparar com a versão inicial. As otimizações, bem como a discussão crítica de como funcionam, o porquê, e as consequências devem constar no relatório a ser entregue.

Use sempre **a mesma máquina** para os testes, uma vez que mudanças no hardware **afetarão** o desempenho de seu programa.

3.1 Sugestões de métricas a serem avaliadas

- Tempo de execução das principais partes do método
- Uso da largura de banda da memória e na cache
- Quantidade de acertos/erros na cache
- ...

3.2 Sugestões de otimizações

- Representação da matriz do problema (otimizar a estrutura de dados)
- *Loop unroll and jam* (desenrolar laços)
- Utilizar operações vetoriais do tipo AVX (pesquisar por *intrinsics*[2])
- Alterações no uso e organização dos dados na memória
- ...

4 Entrega

Você deverá entregar pelo SUAP, até 26/maio, um arquivo *.zip* contendo uma pasta, que por sua vez contém:

- Os arquivos `.c` do trabalho, uma versão não otimizada e uma versão otimizada.
- Um arquivo Makefile que compila o seu trabalho e gera o binário `cgSolver`.
- Um arquivo de texto chamado `README.md` contendo o nome completo dos integrantes do grupo e quaisquer outras informações sobre o progresso do trabalho que julgarem relevantes;

[1] `https://github.com/RRZE-HPC/likwid`
[2] `https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html`

- Um arquivo chamada `relatorio.pdf` que contém a descrição detalhada do que foi implementado, os gráficos de desempenho e as análises correspondentes.
- Outros scripts e programas usados para cumprir o trabalho, como scripts de testes.

O arquivo compactado descrito acima deve ser nomeado com as iniciais do nome de cada integrante separados por um hífen. Por exemplo: se o Fulano da Silva Sousa fez o trabalho com o João de Souza, o arquivo deve ser nomeado `fss-js.zip`. A pasta, dentro do arquivo compactado, deve ter o mesmo nome, a não ser pela extensão.

O trabalho pode ser feito individualmente ou em duplas.

Histórico das Revisões:

- 30/abr/2026 - v1.2: data de entrega adiada.
- 08/abr/2026 - v1.1: pontuação; adicionados links externos; definir linguagem como C/C++.
- 07/abr/2026 - v1.0: primeira versão.