Paralelização de um *Solver* Intervalar para a Resolução de Sistemas Lineares Esparsos

Antonio Argeu Moreira de Lima[1], Prof. Dr. Carlos Amaral Hölbig

Curso de Ciência da Computação – Universidade de Passo Fundo (UPF)
Campus 1 – BR 285 – Bairro São José – CEP 91501-970 – Passo Fundo – RS – Brasil
{75956, holbig}@upf.br

Resumo. *O presente artigo descreve o estudo que está sendo realizado sobre o solver BAND. O solver verificado BAND é um módulo disponível na biblioteca C-XSC, e tem por finalidade a resolução de sistemas lineares com matrizes esparsas do tipo banda. Sistemas com matrizes banda representam um tipo especial de sistema esparso, onde os elementos não-nulos estão localizados em torno da diagonal principal. Este estudo objetiva a paralelização deste solver, a fim de se obter uma melhora no desempenho das aplicações relacionadas em relação à versão seqüencial.*

1. Introdução

A resolução de sistemas lineares do tipo $Ax = b$, onde A é uma matriz quadrada, de ordem n, b é o vetor de termos independentes e x o vetor solução, é um dos assuntos mais abordados em Análise Numérica. Tal importância reservada a esse assunto deve-se às suas amplas aplicações em diversas áreas do conhecimento humano, como, por exemplo, em construção de curvas e superfícies por pontos especificados, desenvolvimento de redes elétricas, programação linear geométrica, teoria dos grafos, economia, computação gráfica, tomografia computadorizada, fractais, criptografia e genética [Anton 2001].

Este trabalho tem por meta apresentar um estudo com relação ao desempenho do *solver BAND*, implementado em C-XSC, que atua na resolução de sistemas lineares com matrizes esparsas do tipo banda. Sistemas com matrizes banda são um caso particular de sistemas esparsos, onde os elementos não nulos encontram-se em torno da diagonal principal [Hölbig 2005]. O *solver* suporta dados de entrada do tipo real e real intervalar, e dados de saída do tipo real intervalar. Todos os algoritmos foram implementados com técnicas que possibilitam a Computação Verificada.

A Computação Verificada significa o processamento numérico de problemas utilizando uma aritmética de alta exatidão, métodos intervalares de inclusão e a convergência garantida pelo Teorema de Ponto Fixo de Brouwer. Entende-se por aritmética de alta exatidão a aritmética computacional de ponto-flutuante baseada no padrão IEEE-754, acrescida de arredondamentos direcionados, da Matemática Intervalar e do cálculo do produto escalar e somatórios em registradores especiais, que permitem

[1] Bolsista PIBIC-CNPq

que valores parciais sejam armazenados sem arredondamentos, resultando que o valor final dessas operações difira do valor real por apenas um arredondamento, vindo daí a alta exatidão.

Este trabalho é uma extensão da tese de doutorado de Hölbig (2005) e corresponde a um dos trabalhos que fazem parte do projeto em cooperação internacional com as universidades alemãs Wuppertal e Karlsruhe para a otimização da biblioteca C-XSC.

2. Biblioteca C-XSC

O C-XSC é uma biblioteca científica baseada na linguagem C++ criada para fins de desenvolvimento de algoritmos numéricos, com a geração de resultados com alta exatidão e verificados automaticamente. Algumas características importantes da biblioteca são [Hölbig 2005]:

- Aritmética intervalar para números reais, complexos, intervalares e intervalares complexos com propriedades definidas matematicamente;
- Tipos de dados de alta exatidão;
- Operadores aritméticos predefinidos com alta exatidão;
- Controle de arredondamento para os dados de entrada e saída;
- Aritmética de múltipla precisão dinâmica e funções padrão.

3. *Solver BAND*

As matrizes do tipo banda são matrizes onde os elementos não-nulos são alocados em faixas (*bandwidths*). Para esse tipo de matriz, o algoritmo utilizado para resolver sistemas densos não é eficiente, uma vez que é necessária uma grande quantidade de memória para alocação dos sistemas e, além disso, há um aumento no tempo de execução.

Os algoritmos utilizados no *solver BAND* foram implementados com base no estreito relacionamento existente entre matrizes do tipo banda e equações diferenciais. Seguindo esse relacionamento, as equações diferenciais podem ser reescritas equivalentemente como um sistema linear triangular com matrizes banda. Similarmente, pode-se reescrever um sistema triangular com matrizes banda como equações diferenciais [Hölbig 2005]. Alguns dos algoritmos que fazem parte do solver são: Decomposição LU, pós-substituição e retro-substituição.

4. Testes de Desempenho

Os testes do *solver* foram realizados em um computador com as seguintes características: processador Intel Pentium IV 3.0 GHz com 2 MB de memória *cache*, 1 GB de memória principal, e com o sistema operacional Linux distribuição Ubuntu 7.04.

O padrão para os testes foi a definição de uma matriz esparsa *A* com 5 bandas contendo os valores 1, 2, 4, 2, 1, e todos os valores de *b* iguais a 1. O tempo de execução do programa foi calculado para sistemas de ordem 50.000, 100.000, 200.000, 400.000,

800.000 e 1.400.000. Dessa forma, foi possível construir o gráfico representado pela Figura 1, que mostra o tempo de execução em função da dimensão.

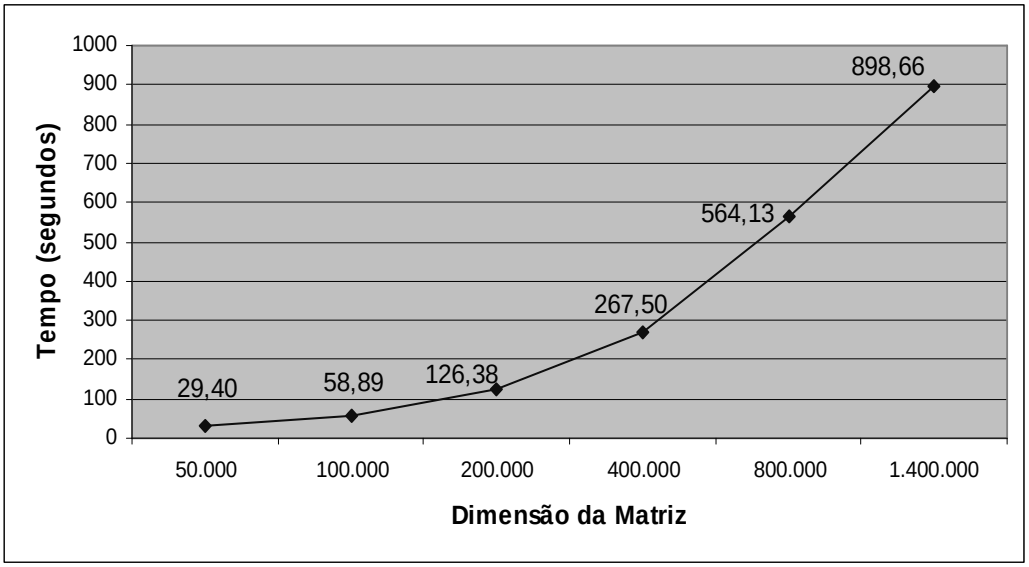

Figura 1. Tempo de execução do *Solver*

Com os resultados apresentados na Figura 1, pôde-se inferir que a ordem da matriz influencia diretamente no tempo de execução do *solver*. Nota-se, além do crescente tempo de processamento, o alto consumo de memória decorrente da alocação das matrizes. Devido a esse fato, o limite da dimensão da matriz testada foi o de 1.4000.000, pois a resolução de sistemas com matrizes de ordem maior não foram completados, mesmo com a utilização de *swap* que também tende a se esgotar. Na ordem de 1.600.000, por exemplo, o programa foi abortado pelo sistema operacional por falta de memória. Para efeito de testes, com ordem igual ou maior à citada anteriormente, será preciso uma quantidade de memória principal consideravelmente maior que à utilizada nos testes deste artigo.

Quanto às funções internas do *solver* BAND, o gráfico da Figura 2 mostra aquelas que exigem mais tempo de processamento. A Decomposição LU é executada uma vez durante o programa, e demonstra um comportamento semelhante ao gráfico da Figura 1. Tanto a pós-substituição, quanto a retro-substituição possuem um tempo de execução semelhante, sendo as duas funções que mais necessitam de tempo de processamento do *solver*. Além disso, devido a estrutura do *solver*, algumas funções são executadas várias vezes, o mesmo ocorre com a pós-substituição e retro-substituição, e como conseqüência disto, elas são responsáveis por cerca de 80% do tempo de execução do *solver*.

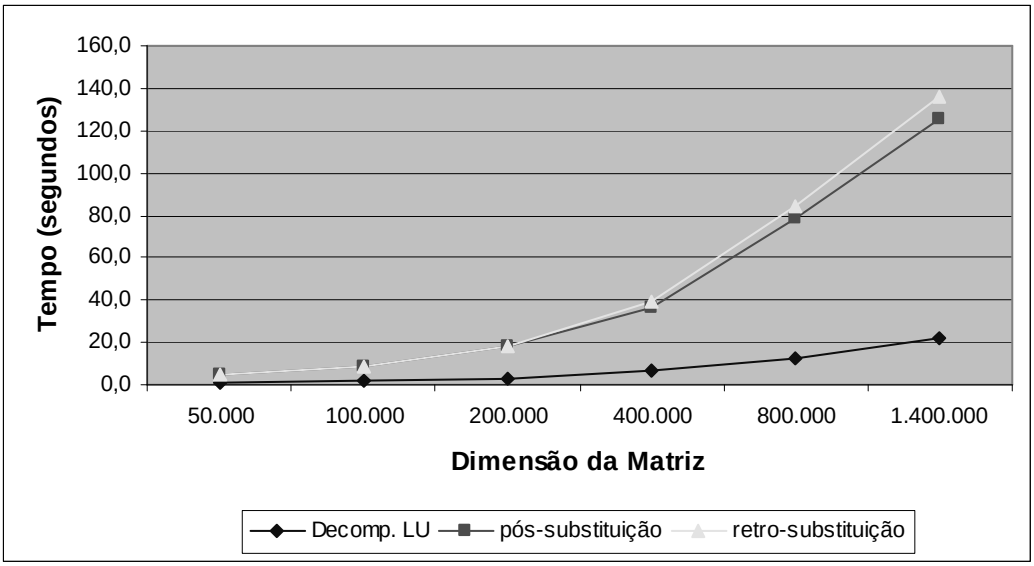

Figura 2. Tempo de execução da Decomposição LU, pós-substituição e retro-substituição

5. Considerações Finais

Com a finalização do estudo do solver BAND e seus algoritmos será possível a escolha de uma alternativa de paralelização. Inicialmente, poderá ser considerada a paralelização do solver com a utilização de *threads* para comunicação intra-nodo, ou da biblioteca MPI para troca de mensagens. Para definir isto, será necessário avaliar se o custo de comunicação pela rede usando MPI será compensando pelo tempo de processamento total gasto. Caso isto não ocorra, o uso de *threads*, ou *threads* com MPI pode ser uma boa alternativa.

Além do tempo de processamento, o uso de memória pelo *solver* é um fator que deve ser considerado, levando-se em consideração o alto consumo de memória para sistemas de ordem superior a 1.600.000, que torna inviável a execução do programa para sistemas de tal ordem em máquinas SISD. Portanto, a distribuição do problema entre os nodos de um *cluster* pode permitir que sistemas de dimensões maiores possam ser processados, pelo fato de que a quantia de memória disponível é normalmente maior em máquinas paralelas.

Referências

Anton, H.; Rorres, C.. *Álgebra Linear com Aplicações*. Oitava Edição, Bookman, Porto Alegre, 2001.

Hölbig, C. A.. *Ambiente de Alto Desempenho com Alta Exatidão para a Resolução de Problemas*. 2005. Tese (Instituto de Informática/Programa de Pós-Graduação em Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2005.

Klatte, R. et al. *C-XSC – A C++ Class Library for Extended Scientific Computing*. Springer-Verlag, Heidelberg, 1993.