Implementação do módulo LSS em C-XSC: Solucionador de Sistemas Lineares com alta exatidão

Paulo Sérgio Morandi Jr. [1],
Bernardo F. Krämer Alcalde [2],
Tiarajú Asmuz Diverio

Instituto de Informática e PPGC – UFRGS
Av. Bento Gonçalves, 9500 – Campus do Vale
Bloco IV – Bairro Agronomia
Porto Alegre – RS – Brasil
CEP 91501-970 Caixa Postal: 15064
{bfkalcalde, sergio, diverio}@inf.ufrgs.br

Carlos Amaral Hölbig

Universidade de Passo Fundo
Curso de Ciência da Computação – ICEG
Passo Fundo – RS – Brasil
&
PPGC – UFRGS
holbig@upf.br

Introdução

A resolução de sistemas lineares do tipo Ax = b, onde A é uma matriz quadrada, de ordem n x 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 [ANT 01] 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.

O presente trabalho tem por meta apresentar algoritmos, implementados em C-XSC, que resolvem não apenas sistemas quadrados, mas, também, sobre-determinados (ou seja, com mais equações do que incógnitas) e sub-determinados (com mais incógnitas do que equações) para valores reais, números complexos, intervalos e intervalos complexos. O cálculo da aproximação da inversa para as matrizes quadradas e da pseudo-inversa para as matrizes sobre e sub-determinadas, é igualmente abordado. Todos esses algoritmos foram implementados com técnicas que possibilitam a Computação Verificada.

Computação Verificada significa o processamento numérico de problemas, utilizando a aritmética de alta exatidão, os 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 de produto escalar e somatórios em registradores especiais que permitem 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 máxima exatidão. Os

[1,2] Alunos do curso de Ciência da Computação da UFRGS (Bolsistas do Projeto LabTeC – DELL – UFRGS - Laboratório de Tecnologia em Clusters.

métodos de inclusão trabalham com intervalos. Cada aproximação é um intervalo que contém a solução do problema. As aproximações sucessivas são resultantes da intersecção de intervalos, o que garante que tenham um diâmetro menor (ou igual) a aproximação anterior. Esses métodos também identificam a não existência de solução, através de uma aproximação vazia (o resultado não é um intervalo).

Os programas desenvolvidos neste trabalho foram baseados nos algoritmos apresentados em *Numerical Toolbox for Verified Computing II – Advanced Numerical Problems*, de W. Krämer, U. Kulisch e R. Lohner, que serão, posteriormente, adaptados e portados para o ambiente de alto desempenho (agregados), visando assim, aliar a alta exatidão com o alto desempenho.

Códigos em PASCAL-XSC

Os algoritmos que serão descritos na próxima seção encontravam-se codificados para Pascal-XSC. Por ser uma linguagem essencialmente seqüencial, o Pascal-XSC [KLA 91] dificulta a paralelização, que é o objetivo da continuação deste trabalho. Não existem no mercado distribuições de bibliotecas que tornam o Pascal paralelo. Já para o C++ existem um grande número de bibliotecas (MPI, DECK, threads POSIX) que o tornam uma linguagem de programação paralela. Isso se deve ao fato do C++ possuir uma implementação mais (implicitamente) paralela, com sua orientação à objetos. Tendo em vista o futuro do trabalho e as condições citadas acima, os algoritmos codificados em Pascal-XSC foram transcritos para a linguagem C-XSC [KLA 93].

LSS – Linear System Solver

O LSS (Linear System Solver) é uma ferramenta que auxiliará na resolução dos seguintes problemas:

1. Solução de Sistemas Lineares Sobre-determinados (m x n, com $m > n$);
2. Solução de Sistemas Lineares Quadrados (n x n);
3. Solução de Sistemas Lineares Sub-determinados (m x n, com $m < n$);
4. Calculo da Inversa A^{-1} de A, no caso do Sistema Quadrado;
5. Calculo da Pseudo-inversa A^{+} de A, no caso do Sistema Sobre-determinado;
6. Calculo da Pseudo-inversa A^{+} de A, no caso do Sistema Sub-determinado.

Abaixo estão descritos os algoritmos que serão usados para resolver esses problemas.

Algoritmos

Cálculo da aproximação da inversa: Calcula a aproximação da inversa de uma matriz A, através do algoritmo de Gauss-Jordan. Além da matriz A, recebe como parâmetro um código controlador de erro, que indica se foi possível fazer o cálculo ou não.

Solução de Sistemas Lineares Quadrados (n x n): Primeiro o algoritmo calcula uma aproximação R1 da inversa de A (Gauss-Jordan) e, após, testa a precisão da solução. Se ela for ruim, ela multiplica a aproximação da inversa pela própria matriz (R1.A) e calcula uma nova aproximação (S1) da inversa de R1.A, e testa novamente a precisão da solução. Caso o resultado continue insatisfatório, o programa avisa que ocorreu um erro (matriz mal-condicionada ou singular).

Solução de Sistemas Lineares Sobre-determinados (m x n, com m > n): Para o cálculo de sistemas sobre-determinados [KRA 00] usamos também o algoritmo 3.1. Será resolvido esse Sistema Linear utilizando-se a aproximação de mínimos quadrados ($A^HAx = A^Hb$). Seguiremos a sugestão do [KRA 00] e evitaremos fazer tantas multiplicações, bastando para isso resolver o seguinte sistema ($n+m$) x ($n+m$) $A_{big}\, Y_{big} = B_{big}$:

$$A_{big} = \begin{pmatrix} A & -I \\ 0 & A^H \end{pmatrix},\ B_{big} = \begin{pmatrix} b \\ 0 \end{pmatrix},\ Y_{big} = \begin{pmatrix} x \\ y \end{pmatrix},$$

onde A^H é a matriz hermitiana (transposta no caso dos números reais), I é matriz identidade (m x m). O bloco x do vetor Y_{big} corresponde à solução desejada.

Solução de Sistemas Lineares Sub-determinados (m x n, com m < n): Para o cálculo de sistemas sub-determinados [KRA 00], estaremos interessados nas soluções de *Norma Euclidiana Mínima* e usaremos o mesmo algoritmo anterior, apenas mudando um pouco o sistema de entrada:

$$A_{big} = \begin{pmatrix} A^H & -I \\ 0 & A \end{pmatrix},\ B_{big} = \begin{pmatrix} 0 \\ b \end{pmatrix},\ Y_{big} = \begin{pmatrix} x \\ y \end{pmatrix},$$

onde A^H é a matriz hermitiana (transposta no caso dos números reais), I é a matriz identidade (n x n). O bloco x do vetor Y_{big} corresponde a solução desejada.

Cálculo da Inversa A^{-1} de A, no caso do Sistema Quadrado: Para calcularmos a Inversa de sistemas lineares quadrados iremos resolver o seguinte sistema: AX = I (onde A é uma matriz n x n e I, uma matriz identidade). Para tanto, passaremos, como parâmetro do algoritmo *3.1,* uma coluna da matriz identidade de cada vez (n vezes, portanto) como um vetor b e um vetor x, que armazenará o resultado desse cálculo. As n respostas desse cálculo corresponderão à solução do sistema desejado (AX = I).

Cálculo da Pseudo-inversa A^+ de A, no caso do Sistema Sobre-determinado: Para calcularmos a Pseudo Inversa de sistemas lineares sobre-determinados utilizaremos a mesma idéia do algoritmo anterior: resolver o sistema AX = I, substituindo A por $A_{big} = \begin{pmatrix} A & -I \\ 0 & A^H \end{pmatrix}$, I ($m$ x m) por $B_{big} = \begin{pmatrix} I \\ 0 \end{pmatrix}$, X por $Y_{big} = \begin{pmatrix} Y \\ X \end{pmatrix}$,

que serão passados como parâmetro para o procedimento anterior, ou seja, será resolvido o sistema $A_{big}\ Y_{big} = B_{big}$. No final, o bloco X da matriz Y é a solução desejada do problema.

Cálculo da Pseudo-inversa A^+ de A, no caso do Sistema Sub-determinado: Para calcularmos a Pseudo Inversa de sistemas lineares sub-determinados utilizaremos a mesma idéia do algoritmo utilizado no cálculo da inversa para sistemas quadrados: resolver o sistema: AX = I, substituindo A por

$$A_{big} = \begin{pmatrix} A^H & -I \\ 0 & A \end{pmatrix},\ I\ (n \times n)\ \text{por}\ B_{big} = \begin{pmatrix} I \\ 0 \end{pmatrix},\ X\ \text{por}\ Y_{big} = \begin{pmatrix} Y^H \\ X^H \end{pmatrix},$$

que serão passados como parâmetro do algoritmo anterior, ou seja, será resolvido o sistema $A_{big}\ Y_{big} = B_{big}$. Novamente, ao final dos cálculos, o bloco X é a solução desejada do problema.

Conclusão

A implementação dos algoritmos do *Numerical Toolbox for Verified Computing II* [KRA 00] em C-XSC é apenas o primeiro passo rumo a um objetivo maior. A meta agora é paralelizar as rotinas do módulo LSS, para que ele possa ser executado em ambientes de alto desempenho, (do tipo agregados). Sendo assim, a escolha do C-XSC em detrimento do Pascal-XSC, o que acarretou o ônus de transcrever os programas de uma linguagem para a outra, não foi meramente arbitrária. O Pascal, por originalmente ser uma linguagem voltada ao processamento seqüencial, mostra-se pouco própria para esse fim. Em termos da linguagem C++/C-XSC, no entanto, graças ao uso da biblioteca MPI, a paralelização das rotinas torna-se viável. Paralelizado, o módulo LSS ficará disponível para o uso nas aplicações desenvolvidas em ambientes de agregados.

Referências

[ANT 01] ANTON, H.; RORRES, C.: *Álgebra Linear com Aplicações*, Oitava Edição, Bookman, Porto Alegre, 2001.

[CLA 94] CLAUDIO, D. M.; MARINS, J. M.: *Calculo Numérico Computacional, Teoria e Pratica*, Segunda Edição, Editora Atlas S.A., 1994.

[KLA 91] KLATTE, R.; KULISCH, U.; NEAGA, M.; RATZ, D.; ULLRICH, CH.: *PASCAL--XSC – Language Reference with examples*, Springer-Verlag, Heidelberg, New York, 1991.

[KLA 93] KLATTE, R.; KULISCH, U.; WIETHOFF, A.; LAWO, C.; RAUCH, M.: *C-XSC – A C++ Class Library for Extended Scientific Computing*, Springer-

[KRA 00] KRAMER, W.; KULISCH, U.; LOHNER, R.: *Numerical Toolbox for Verified Computing II, Advanced Numerical Problems*, http://www.xsc.de.
Verlag, Heidelberg, New York, 1993.

[PRE 90] PRESS, W. H.: *Numerical Recipes in C, The Art of Scientific Computing*, Cambridge University Press, 1990.