SaddleOO um programa para o estudo de soluções de sistemas lineares em problemas de ponto de sela*

ÍTALO C. NIEVINSKI LIMA† LUIZ M. CARVALHO‡

Resumo

Desenvolvimento de uma *toolbox* com interface gráfica para o estudo de soluções numéricas de sistemas lineares com ênfase em problemas de ponto de sela. Software desenvolvido através de orientação a objeto e que terá uma biblioteca de métodos de solução de sistemas lineares e que poderá acomodar novos métodos desenvolvidos pelo usuário.

1 Problema de ponto de sela

Grandes sistemas lineares do tipo ponto de sela surgem em uma grande variedade de aplicações em toda a ciência computacional e engenharia. Devido à sua indefinição e muitas vezes propriedades espectrais pobres, tais sistemas lineares representam uma desafio significativo para os desenvolvedores de solvers [1]. Uma das aplicações está na área de dinâmica dos fluidos computacional onde uma matriz de ponto de sela é obtida através da discretização das equações de Navier-Stokes que modelam o escoamento de um fluido newtoniano utilizadas para simular enchimento de reservatórios de hidrelétricas. São impotantes, então, técnicas específicas para garantir uma convergência rápida de forma a tornar viável grandes simulações. Em [1], temos que o problema de ponto de sela para matrizes reais é caracterizado por sua matriz de coeficientes em blocos 2×2 apresentada abaixo:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix} \quad \text{ou} \quad \mathcal{A}\boldsymbol{x} = \boldsymbol{b}, \tag{1}$$

$$\mathbf{A} \in \mathbb{R}^{n\times n}, \quad \mathbf{B}^T, \mathbf{C} \in \mathbb{R}^{m\times n}, \quad \mathbf{D} \in \mathbb{R}^{m\times m} \quad \text{com} \quad n \geq m. \tag{2}$$

* *Palavras chave:* Sistemas Linares, Problemas de Ponto de Sela, MATLAB, Orientação a Objeto

†Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brasil. E-mail: italonievinski@gmail.com

‡Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brasil. E-mail: luizmc@gmail.com

É verdade que muitos sistemas lineares podem ser postos na forma (1)-(2). Excluímos explicitamente os casos onde A ou um ou ambos B e C são nulos. Quando o sistema linear descreve um problema de ponto de sela (generalizado), os blocos constituintes A, B, C e D satisfazem uma ou mais das seguintes condições:

C1 $\mathbf{A}$ é simétrica: $\mathbf{A} = \mathbf{A}^T$

C2 a *parte simétrica* de $\mathbf{A}$, $\mathbf{H} = \frac{1}{2}(\mathbf{A} + \mathbf{A}^T)$, é positiva semidefinida

C3 $\mathbf{B} = \mathbf{C}$

C4 $\mathbf{D}$ é simétrica ($\mathbf{D} = \mathbf{D}^T$) e positiva semidefinida

C5 $\mathbf{D} = \mathbf{0}$ (a matriz nula)

O estudo de métodos de solução deste tipo de sistema demanda esforço e investimento de tempo em testes e comparações de resultados para diferentes estruturas de matrizes, métodos, reordenamentos, precondicionadores, aproximações entre outros fatores necessários para resolver problemas do tipo apresentado.

2 SaddleOO - A toolbox

O objetivo desse projeto é criar um programa que será uma *toolbox* para desenvolver e testar métodos de soluções numéricas, com ênfase em problemas de ponto de sela, fornecendo entre suas ferramentas: bibliotecas de métodos de soluções de sistemas lineares, formas de operar a multiplicação matriz $\times$ vetor, precondicionadores, decomposições e ordenamentos, entre outras, desenvolvido dentro do paradigma da orientação a objeto e com interface gráfica. A ideia principal é facilitar, o quanto for possível, o estudo de métodos para resolver sistemas lineares fornecendo ao usuário diversas opções para auxiliá-lo no desenvolvimento de seus próprios testes sem impor restrições. Um dos pontos importantes do programa será uma ferramenta que gera relatórios com os dados obtidos da resolução dos problemas para uma análise detalhada do método. A estrutura de classes no paradigma da orientação a objeto permite oferecer liberdade para implementar ideias em novas classes sem modificar a estrutura como um todo. Com uma documentação apropriada, esta característica diz respeito tanto ao desenvolvedor quanto ao usuário, permitindo ao mesmo a inserção de novas ferramentas com tranquilidade. Esta é uma das propostas relevantes deste projeto, uma vez que para desenvolver novos métodos surgem novas ideias que

não podem ser inteiramente contempladas no programa, mas serão incorporadas a ele com um esforço computacional baixo.

Para realização do projeto, o desenvolvimento do programa foi dividido em duas fases principais, sendo a primeira o desenvolvimento de um protótipo totalmente operacional utilizando o MATLAB, com a estrutura de classes e interface bem definidas. A segunda fase será, já com um protótipo em funcionamento, reescrever de forma otimizada toda a *toolbox* em C++, onde a orientação a objeto é muito mais eficiente e os testes podem ser rodados em problemas mais difíceis, com matrizes de maior ordem, com maior velocidade, utilizando pacotes de reconhecida eficácia na comunidade científica como o BLAS [2], Lapack [11] e Metis [12]. Pensa-se também na sua integração com softwares que resolvem sistemas lineares oriundos de diversas aplicaçoes científicas e industriais, como o IFISS em MATLAB [6], e outros conhecidos pacotes para computação científica como Trillinos em C++ [17] e PETSc em C [14].

3 A primeira versão

Uma primeira versão do programa já está em funcionamento em MATLAB, inclusive com interface gráfica, e contempla mais de 200 formas de se resolver um sistema linear com matrizes de ponto de sela através da combinação de diferentes precondicionadores, reordenamentos e solvers. Tabelas com resultados e dados dos testes já podem ser gerados em planilhas em formato XLS ou TEX para compilação em PDF.

O desenvolvimento deste primeira versão foi crucial para entender as dificuldades de se construir uma plataforma genérica o bastante para auxiliar no estudo de problemas de ponto de serviu de inspiração para se dar inicio a uma versão muito mais abrangente e eficaz para se atinfir o objetivo proposto. Como poder ser visto na Figura 1, a interface também apresenta algumas informações sobre a matrizes, como a visualização dos elementos não nulos, que pode ser muito importante em problemas envolvendo matrizes esparsas.

Além disso a versão pode ser perfeitamente utilizada para resolver problemas pelo método de blocos com precondicionador contruído através do método de projeção, abaixo descrito.

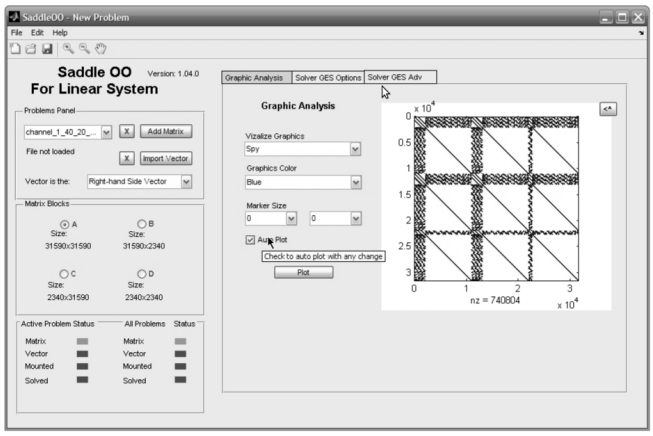

Figura 1: SaddleOO v1.04 com uma matriz carregada

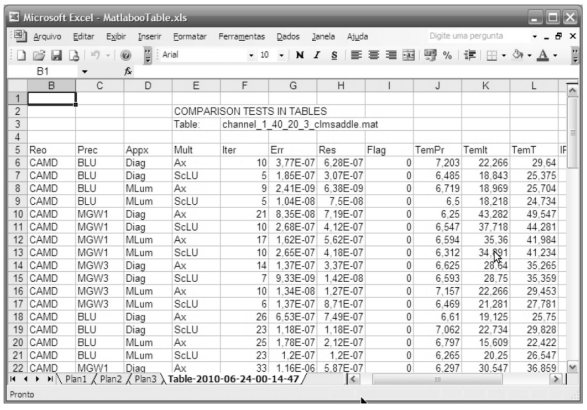

Figura 2: Planilha de relatório gerada a partir de um problema pelo SaddleOO v1.04

3.1 Método de bloco com precondicionador construído através de método de projeção

Seja $\mathcal{A}$ a matriz de coeficientes do sistema $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$ tal que:

$$\mathcal{A} = \begin{bmatrix} \mathbf{A} & \mathbf{G} \\ \mathbf{D} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix} \quad \text{ou} \quad \mathcal{A}\boldsymbol{x} = \boldsymbol{b}, \tag{3}$$

$$\mathbf{A} \in \mathbb{R}^{n\times n}, \quad \mathbf{G}^T, \mathbf{D} \in \mathbb{R}^{m\times n}, \quad \mathbf{C} \in \mathbb{R}^{m\times m} \quad \text{com} \quad m \geq n. \tag{4}$$

Para resolver o sistema $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$ pelo método de blocos utilizamos a seguinte metodologia:

1. **Reordenamento de $\boldsymbol{A}$**
2. **Multiplicação Matriz $\times$ Vetor**
3. **Precondicionador**

O sistema definido por estas características será resolvido como uma única matriz através de um método iterativo de Krylov, que será referido aqui como método externo.

1. Um reordenamento aplicado em $\mathbf{A}$, de modo a preservar sua simetria, em seguida as trocas de linhas são aplicadas em $\mathbf{G}$ e a troca de colunas aplicadas em $\mathbf{D}$;

2. A forma que será realizada a operação $\mathcal{A}\boldsymbol{x}$, ou seja, a operação matrix vetor que é feita a cada iteração do método externo. No nosso caso, utilizamos a multiplicação em blocos, já que estamos resolvendo o problema através dos blocos $\mathbf{A}$, $\mathbf{G}$, $\mathbf{D}$, $\mathbf{0}$.

3. Para precondicionar o sistema constrói-se $\mathbf{M} = \tilde{\mathcal{A}}$ ($\tilde{\mathcal{A}}$ é uma aproximação de $\mathcal{A}$) tal que $\mathbf{M}^{-1}\mathcal{A}\boldsymbol{x} = \mathbf{M}^{-1}\boldsymbol{b}$ onde $M^{-1}\mathcal{A}$ possui melhores propriedades numéricas do que $\mathbf{A}$. Uma das alternativas implantadas é a utilização de um método de projeção (ou seja, aproxima-se $\mathbf{S}$, a matriz do complemento de Schur) e o precondicionador utilizado usa uma fatoração LU incompleta: segue a fórmula (8)

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{G} \\ \mathbf{D} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{D} & \tilde{\mathbf{S}} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{A}}^{-1}\mathbf{G} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{5}$$

Onde $\tilde{\mathbf{A}}$ é uma aproximação de $\mathbf{A}$ e $\tilde{\mathbf{S}} = -\mathbf{D}\tilde{\mathbf{A}}^{-1}\mathbf{G}$

Para aplicar $\mathbf{M}^{-1}\mathbf{A}\boldsymbol{x} = \mathbf{M}^{-1}\boldsymbol{b}$, como calcular $\mathbf{M}^{-1}$ é caro, pois é o custo de uma eliminação gaussiana, a cada iteração do método fazemos a multiplicação $\mathcal{A}\boldsymbol{x}_0 = \boldsymbol{y}$ e em seguida $M\backslash\boldsymbol{y}$ utilizando um método iterativo, no nosso caso, métodos de Krylov.
Para resolver $\boldsymbol{M}\backslash\boldsymbol{y}$, seguimos:

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{D} & \tilde{\mathbf{S}} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{A}}^{-1}\mathbf{G} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} \tag{6}$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{D} & \tilde{\mathbf{S}} \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} \tag{7}$$

$$\mathbf{A}\boldsymbol{z}_1 = \boldsymbol{y}_1 \tag{8}$$

$$\tilde{\mathbf{S}}\boldsymbol{z}_2 = \boldsymbol{y}2 - \mathbf{D}\boldsymbol{z}_1 \tag{9}$$

Para resolver (11) utilizamos a simetria de A e fazemos PCG, com precondicionador Cholesky incompleto. Com z_1 obtido em (11) conhecemos $-\mathbf{D}\boldsymbol{z}_1$
Para resolver (12) utiliza-se uma metodologia similar a que estamos estruturando para o problema principal.

3.1. Reordenamento de $\tilde{\mathbf{S}}$
É feito o reordenamento da matriz $\tilde{\mathbf{S}}$ o que obriga uma permutação de colunas em $\mathbf{G}$ e de linhas em $\mathbf{D}$.

3.2. Multiplicação Matriz × Vetor
Escolhemos como será feita a operação Matriz × Vetor a cada iteração do método

3.3. Precondicionador
Utilizamos uma fatoração aproximada de $\tilde{\mathbf{S}}$ como precondicionador (Ex.: LU incompleto)

Seguindo o formato estabelecidos resolve-se (12). Voltando ao sistema temos agora:

$$\mathbf{M} = \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{A}}^{-1}\mathbf{G} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{bmatrix} \tag{10}$$

$$\boldsymbol{x}_2 = \boldsymbol{z}_2 \tag{11}$$

$$\boldsymbol{z}_1 + \tilde{\mathbf{A}}^{-1}\mathbf{G}\boldsymbol{z}_2 = \boldsymbol{z}_1 \tag{12}$$

$$\boldsymbol{x}_1 = \boldsymbol{z}_1 - \tilde{\mathbf{A}}^{-1}\mathbf{G}\boldsymbol{z}_2 \tag{13}$$

A aproximação $\tilde{\mathbf{A}}$ de $\mathbf{A}$ é tal que o cálculo de $\mathbf{A}^{-1}$ seja viável, assim resolve-se (16) com com uma simples multiplicação. A etapa 2 da metodologia apresentada se repete a cada iteração do método externo, até sua convergência.

4 Desenvolvimento da segunda versão

Com o aprendizado obtido da primeira versão, uma estrutura considerada muito superior já está sendo implementada, com novas técnicas para a construção da interface gráfica e classes mais eficientes dentro do objetivo proposto. Com a implementação da nova versão haverá maior flexibilidade para o usuário implantar seus próprios métodos e para utilização de matrizes de diversas origens e formatos, as interfaces serão configuráveis pelo usuário de forma simples, haverá facilidade para incorporar solvers e precondicionadores escritos em Matlab ou em outras linguagens, geração de códigos para os programas montados a partir da interface, entre outras modificações e flexibilizações.

Esta versão pretende não somento executar problemas, mas será capaz de gerar o script para a resolução do problema utilizando a biblioteca orientada a objeto, desta forma o usuário terá o código fonte da resolução do problema e estará livre para fazer suas próprias modificações particulares. Assim a interface gráfica terá seu papel de facilitador no uso das classes sem impor restrições ao usuário. As imagens a seguir mostram duas abas da segunda versão do programa. A Figura 3 mostra a aba que será usada para organizar os solvers individualmente que serão armazenados e podem então ser utilizados na aba Problem Maker para resolver problemas lá organizados, como mostra a Figura 4.

5 Resultados e implementações

A segunda versão está ainda em fase de idealização, tanto em relação a interface quanto às classes da orientação a objeto, que já está com uma estrutura que parece interessante e capaz de cumprir o objetivo. As funções presentes na primeira versão deverão ser todas implantadas na segunda, além disso dados gráficos importantes como os de velocidade de convergencia e autovalores tambéms deverão ser implementados.

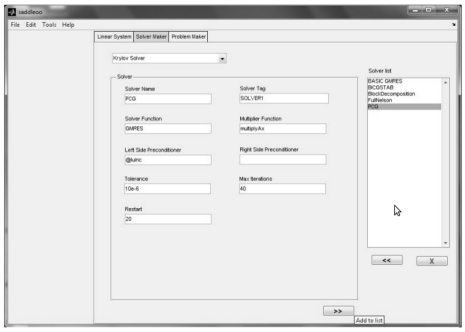

Figura 3: Aba Solver Maker do Saddleoo v.2.0

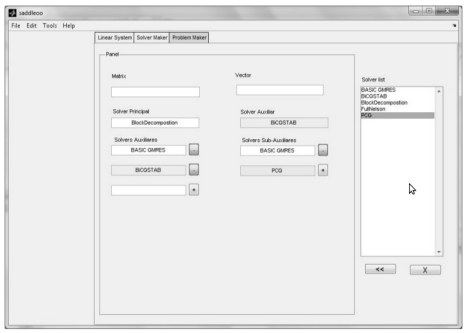

Figura 4: Aba Problem Maker do Saddleoo v.2.0

Referências

[1] BENZI, M.; GOLUB, G. H.; LIESEN, J. Numerical solution of saddle point problems. Acta Numerica, v. 14, p. 1-137, May 2005.

[2] BLAS, <http://www.netlib.org/blas>, acessado em 28/05/2011.

[3] CARVALHO, L. M.; GRATON, S. Avanços em métodos de Krylov para solução de sistemas lineares de grande porte. SBMAC, 2009.

[4] CUTHILL, E.; MCKEE, J. Reducing the bandwidth of sparse symmetric matrices. In: Proc. 24th Nat. Conf. ACM. [S.l.: s.n.], 1969. p. 157-172.

[5] DONGARRA, J. J. Performance of various computers using standard linear equation solvers. n. CS - 89 - 85, March 2008.

[6] ELMAN, H.; SILVESTER, D.; WATHEN, A. Finite Elements and Fast Iterative Solvers with application in incompressible fluid dynamics. Oxford University Press 2005

[7] FORTES, W. R. Precondicionadores e solucionadores para resolução de sistemas lineares obtidos de simulação de enchimento de reservatórios, Dissertação de Mestrado, PPG-Eng.Mec. UERJ , 2008.

[8] FREUND, R. W.; GOLUB, G. H.; NACHTIGAL, N. M. Iterative solution of linear systems. Acta Numerica, Cambridge University Press, p. 57-100, 1992.

[9] GREENBAUM, A. Iterative methods for solving linear systems. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997. ISBN 0-89871-396-X.

[10] LANCZOS, C. Solution of systems of linear equations by minimized iterations. Journal of Research of the National Bureau of Standards, v. 49, n. 1, p. 33-53, July 1952.

[11] Lapack, <http://www.netlib.org/lapack>, acessado em 28/05/2011.

[12] METIS, <http://glaros.dtc.umn.edu/gkhome/metis/metis/overview>, acessado em 28/05/2011.

[13] MEYER, C. D. Matrix analysis and applied liner algebra. [S.l.]: SIAM, 2000.

[14] PETSc, <http://www.mcs.anl.gov/petsc/>, acessado em 28/05/2011.

[15] REGISTER. A. H., A Guide to MATLAB Object-Oriented Programming., Chapman & Hall/CRC, Georgia, 2007.

[16] SAAD, Y.; SCHULTZ, M. H. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput., v. 7, n. 3, p. 856-869, 1986.

[17] Trilinos, <http://trilinos.sandia.gov/>, acessado em 28/05/2011.