Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia

João Paulo Kely Zanardi

Precondicionadores Baseados na Aproximação da Inversa

Rio de Janeiro

2014

João Paulo Kely Zanardi

Precondicionadores Baseados na Aproximação da Inversa

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Ciências, ao Programa de Pós-Graduação em Engenharia Mecânica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Fenômenos de Transporte.

Orientador: Prof. Dr. Luiz Mariano Paes de Carvalho Filho

Rio de Janeiro

2014

João Paulo Kely Zanardi

Precondicionadores Baseados na Aproximação da Inversa

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Ciências, ao Programa de Pós-Graduação em Engenharia Mecânica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Fenômenos de Transporte.

Aprovado em: 07/01/2014

Banca Examinadora:

Prof. Dr. Luiz Mariano Paes de Carvalho Filho
Instituto de Matemática e Estatística da UERJ

Prof. Dr. Norberto Mangiavacchi
Faculdade de Engenharia da UERJ

Prof. Dr. Alexandre Loureiro Madureira
Coordenação de Matemática Aplicada e Computacional do LNCC

Prof. Dr. Paulo Goldfeld
Departamento de Matemática Aplicada da UFRJ

Dr. José Roberto Rodrigues
PETROBRAS/CENPES/Tecnologia de Reservatórios

Rio de Janeiro

2014

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

Z27 Zanardi, João Paulo Kely.
Precondicionadores baseados na aproximação da inversa / João Paulo Kely Zanardi. - 2014.
49 f.

Orientador: Luiz Mariano Paes de Carvalho Filho.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia Mecânica. 2. Sistemas lineares – Dissertações. 3. Metodos interativos – Dissertações. 4. Solucionadores de Krylov – Dissertações. I. Carvalho Filho, Luiz Mariano Paes de. II. Universidade do Estado do Rio de Janeiro. III. Título.

CDU 627.03

Assinatura Data

DEDICATÓRIA

À minha mãe, à minha irmã e aos meus amigos.

AGRADECIMENTO

Agradeço, em primeiro lugar, a Goku, pois sem ele não estaria aqui.

Agradeço aos meus familiares por todo suporte durantes esses anos, em especial à minha mãe Vera Lúcia de Oliveira Kely.

Agradeço aos meus professores, em especial ao professor Luiz Mariano de Carvalho, pois graças a ele concluí mais esta etapa.

Agradeço aos meus amigos em especial à Stephanie Rabello de Melo, Fellipe de Oliveira Pinto, Fernanda de Oliveira Souza, Isabela Paixão Josiane Nonato, Luis Philipe Barrozo, Mario Maia, Tatiana Clessia e Thiago José Figueira, por me aturarem.

E agradeço à CAPES pelo fomento fornecido.

RESUMO

ZANARDI, João Paulo K. *Precondicionadores Baseados na Aproximação da Inversa*, 2014. 51 f. Dissertação (Mestrado em Engenharia Mecânica) - Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro.

Neste trabalho de dissertação apresentaremos uma classe de precondicionadores baseados na aproximação esparsa da inversa da matriz de coeficientes, para a resolução de sistemas lineares esparsos de grandes portes através de métodos iterativos, mais especificamente métodos de Krylov. Para que um método de Krylov seja efeciente é extremamente necessário o uso de precondicionadores. No contexto atual, onde computadores de arquitetura híbrida são cada vez mais comuns temos uma demanda cada vez maior por precondicionadores paralelizáveis. Os métodos de inversa aproximada que serão descritos possuem aplicação paralela, pois só dependem de uma operação de produto matriz-vetor, que é altamente paralelizável. Além disso, alguns dos métodos também podem ser construídos em paralelo. A ideia principal é apresentar uma alternativa aos tradicionais precondicionadores que utilizam aproximações dos fatores LU, que apesar de robustos são de difícil paralelização.

Palavras-chave: Sistemas lineares. Métodos Iterativos. Precondicionadores. Inversa Aproximada.

ABSTRACT

ZANARDI, João Paulo K. *Preconditioners Based on Sparse Approximate Inverse*, 2014. 51 f. Dissertação (Mestrado em Engenharia Mecânica) - Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro.

In this thesis we will present a class of preconditioners based on the sparse approximation of the inverse of the coefficient matrix, for the solution of large sparse linear systems by iterative methods, Krylov methods in special. In order to make an efficient Krylovś method solver it is extremely necessary the use of a good preconditioner. In the actual scenario, where hybrid architectures machines are more and more common we have an increasing on the demand of highly parallel preconditioners. The sparse approximate inverse methods that will be decribed have parallel application, since its application needs only sparse matrix-vector multiplication operations, which is a highly parallelizable kernel. Futhermore, some methods can also be constructed in parallel. The main ideia of this thesis is to present an alternative to the traditionals preconditioners based on sparse approximation of the LU factors, which are very robust but their parallelization are nontrivial.

Keywords: Linear Systems. Iterative Methods. Preconditioners. Approximate Inverse.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

Um dos mais importantes problemas em computação científica é resolver, de maneira eficiente, sistemas lineares da forma

$$Ax = b,$$

onde A é uma matriz não singular esparsa em $\mathbb{R}^{n\times n}$, com n grande, x é o vetor das incógnitas e b é um vetor em $\mathbb{R}^{n\times 1}$. Em nosso caso, A é proveniente da discretização de equações diferenciais parciais (EDPs).
Devido ao tamanho de A, fica inviável resolver este sistema linear utilizando métodos diretos. Então são usados métodos iterativos, isto é, procedimentos que geram soluções aproximadas do sistema a cada iteração. Para que um método iterativo seja eficiente, é necessário o uso de precondicionadores. Um precondicionador é uma matriz (no nosso caso, esparsa) P tal que o número de condicionamento da matriz precondicionada AP^{-1} ou $P^{-1}A$ é menor que o número de condicionamento de A ou tal que os autovalores da matriz preconcionada tenham uma distribuição mais adequada, por exemplo agrupados em torno de 1.

Observação 1 *O número de condicionamento de uma matriz A é definido como*

$$\kappa(A)_p = \|A\|_p \|A^{-1}\|_p,$$

onde $\|.\|_p$ denota a norma p matricial. A taxa de convergência de métodos iterativos está ligada a este número.

O precondicionamento pode ser feito multiplicando-se a inversa P^{-1} de P, pela esquerda

$$P^{-1}Ax = P^{-1}b$$

se obtendo a mesma solução do sistema original ou pela direita fazendo

$$AP^{-1}y = b,\ Px = y.$$

Existem diversos métodos que calculam a matriz P. Os mais utilizados em nossa classe de problemas são os baseados na fatoração LU incompleta da matriz A [21]. Infeliz-

mente este tipo de precondicionadores são estritamente sequenciais, contrastando com o atual cenário de paralelismo e máquinas híbridas [20]. Neste contexto surgem os precondicionadores baseados na aproximação da inversa, isto é, calcula-se uma matriz esparsa M, tal que $M \approx A^{-1}$ em algum sentido. Estas técnicas de cálculo do precondicionador foram bastante estudadas no fim da década de 90 e início da década de 2000 [2], [3], [11] e [7] mas se mostraram ineficientes se comparados com os baseados na fatoração LU incompleta. No entanto, os precondicionadores de inversa aproximada voltaram a ser estudados recentemente por terem aplicação paralelizável, pois estes utilizam operações de produto matriz-vetor em sua aplicação, diferentemente dos precondicionadores de fatoração LU incompleta que precisam da resolução de sistemas triangulares.

Neste trabalho vamos apresentar uma visão geral sobre os precondicionadores que utilizam técnicas de aproximação da inversa. No Cápitulo 1 será feito um apanhado dos artigos lidos, explicitando os artigos e apontando a importância de cada um na concepção deste trabalho. No Cápitulo 2 serão apresentadas as diferentes técnicas de aproximação da inversa e seus principais métodos serão detalhados. No Cápitulo 3 serão discutidos aspectos computacionais, também serão apresentados pseudo-códigos dos métodos e uma discussão sobre a implementação dos algoritmos. Por último, no Capítulo 4 serão apresentados resultados de experimentos numéricos utilizando os precondicionadores de inversa aproximada, com intuito de comparar a inversa aproximada com outros tipos de precondicionador, além de avaliar sua eficácia em melhorar a convergência de métodos iterativos.

1 INVERSA APROXIMADA

Vamos descrever brevemente alguns artigos relevantes na literatura que foram estudados durante a preparação desta dissertação. Em [11] os autores propuseram o Sparse Approximate Inverse (**SPAI**). Após introduzirem o conceito de precondicionador eles descreveram cuidadosamente como calcular um precondicionador baseado na aproximação da inversa através de um padrão de esparsidade fixo ou adaptativo. As colunas do precondicionador são calculadas separadamente como solução exata de um problema de minimização de norma de Frobenius restrito ao padrão de esparsidade. Além disso, foram dadas algumas sugestões para a implementação do algoritmo e muito dos valores padrão que utilizamos em nossos testes foram baseados neste artigo. Um modelo de atualização da fatoração QR, utilizado no modelo adaptativo, também foi apresentado. Os autores também apresentaram algums resultados numéricos com o **SPAI**.

Edmond Chow e Yousef Saad propuseram em [7] uma alternativa ao **SPAI** que calcula iterativamente as colunas um precondicionador de inversa aproximada usando um algoritmo de decrescimento do resíduo, no caso o Minimal Residual (**MR**). Neste artigo são discutidos em detalhe como calcular o precondicionador, como manter a esparsidade do mesmo, além de serem apresentadas algumas propriedades. Uma versão do **MR** com auto precondicionamento também é apresentada e uma discussão sobre estratégias de *dropping* também é feita. Por fim, são apresentados experimentos numéricos com os precondicionadores obtidos.

Em [2], é descrito o Approximate Inverse (**AINV**). O artigo compara técnicas de precondicionamento implícito e explícito antes de introduzir o precondicionador. Um processo de biconjugação, que é usado no calculo do precondicionador, é descrito detalhadamente, incluindo algumas características de preenchimento que estão ligadas ao grafo da matriz. Uma versão do **AINV** para blocos também é descrita. Nas últimas seções, aspectos de implementação e experimentos numéricos são apresentados e a estrutura dos dados é detalhada. Nas últimas seções são apresentados resultados de experimentos numéricos. Os resultados, de acordo com os autores, sugerem que o **AINV** é extremamente robusto e comparável a alguns precondicionadores baseados na fatoração LU incompleta.

O artigo [3] resume e compara as contribuições para inversa aproximada a partir de 1980 até o final da década de 90. Na primeira parte os autores revisam algumas técnicas de precondicionadores baseados na aproximação da inversa, classificando estas técnicas em três categorias: métodos baseados na minimização da norma de Frobenius, inversas aproximadas fatorizadas e inversa aproximada dos fatores LU incompletos. Eles apresentam alguns métodos (**SPAI**, **MR**, **AINV**, entre outros), explicam suas ideias básicas e apresentam pseudo códigos. Nas últimas seções, os autores discutem a implementação dos métodos e apresentam alguns experimentos númericos com matrizes do Matrix Market [18], usando métodos de Krylov, como por exemplo o **GMRES** [21]. Além disso, é feita uma comparção entre os métodos apresentados.

Em [6] os autores apresentam uma versão não simétrica do Balanced Incomplete Factorization (**BIF**) [5], que é um algoritmo que computa uma aproximação do fator de Cholesky de uma matriz simétrica positiva definida. O artigo mostra como estender o algoritmo para o caso geral. Além disso, são apresentados vários teoremas que justificam algumas propriedades básicas do método e critérios para escolher alguns fatores escalares. Uma boa estratégia para se preservar a esparsidade do precondicionador também é proposta. Resultados de experimentos numéricos comparando o algoritmo apresentado com um precondicionador de fatoração LU incompleta também são apresentados.

Nos últimos anos, precondicionadores baseados na aproximação da inversa têm ganho destaque devido ao já comentado paralelismo. Como consequência, muitas contribuições podem ser encontradas na literatura, alguns exemplos recentes são [8], [10], [15], [16], [22] e [23]. Usando a ferramenta de busca Google Acadêmico, se encontram centenas de artigos sobre inversa aproximada em métodos de Krylov, entre janeiro e setembro de 2013.

2 MÉTODOS DE APROXIMAÇÃO DA INVERSA

Neste capítulo vamos introduzir alguns métodos de aproximação da inversa. É conveniente utilizar a abordagem proposta em [3] e dividir os métodos de aproximação da inversa três categorias: métodos baseados na minimização da norma de Frobenius, inversa aproximada fatorada e inversa aproximada dos fatores LU incompletos. Na Seção 2.1 descreveremos a ideia de cálculo do precondicionador a partir da minimização da norma de Frobenius e apresentaremos os métodos **SPAI** e **MR**. Na seção 2.2 descreveremos a inversa aproximada fatorada e os métodos **AINV** e **NBIF**. Na seção 2.3 descreveremos como calcular a inversa aproximada dos fatores LU incompletos, mas nenhum método será apresentado.

2.1 Métodos baseados na minimização da norma de Frobenius

Os métodos baseados na minimização da norma de Frobenius foram os primeiros a serem propostos [1], e tem como ideia básica calcular o precondicionador $M \approx A^{-1}$ como solução do seguinte problema de minimização:

$$\min_{M \in S} \|AM - I\|_F, \tag{2.1}$$

onde S é um conjunto de matrizes esparsas, I é a matriz identidade $n \times n$ e $\|.\|_F$ denota a norma de Frobenius matricial. Das propriedades da norma de Frobenius, temos que

$$\|AM - I\|_F^2 = \sum_{k=1}^{n} \|Am_k - e_k\|_2^2. \tag{2.2}$$

Aqui m_k e e_k denotam a k-ésima coluna da matrizes M e I, respectivamente. Então, calcular M se resume a resolver n problemas de mínimos quadrados independentes. Note que essa formulação evidencia que os métodos de minimização da norma de Frobenius, além da aplicação, têm também construção paralelizável. Os principais métodos utilizados para computar M utilizando esta abordagem são o **SPAI** [11] e o **MR** [7], que serão detalhados a seguir.

2.1.1 SPAI

O **SPAI**, **SP**arse **A**pproximate **I**nverse [11], resolve o problema de minimização de norma de Frobenius de maneira exata restrito a um determinado padrão de esparsidade. O padrão de esparsidade de uma matriz qualquer M é definido como um conjunto de índices $G = \{(i, j) \mid m_{ij} \neq 0\}$.
Para um k fixo, vamos mostrar como o **SPAI** resolve o problema:

$$\min_{m_k} \in J_k \|Am_k - e_k\|. \tag{2.3}$$

Seja J_k o conjunto de índices j tal que $m_k(j) \neq 0$, e denotemos $m_k(J_k)$ por $\hat{m_k}$. Claramente vemos que as únicas colunas de A que entram na definição de $\hat{m_k}$ são aquelas cujos índices estão em J_k. Seja $A(:, J_k)$ a submatriz formada por essas colunas e I_k o conjunto de índices i das linhas não nulas de $A(:, J_k)$. Podemos nos restringir (2.3) a matriz $\hat{A} = A(I_k, J_k)$, ao vetor de incógnitas $\hat{m_k}$ e ao lado direito $\hat{e_k} = e_k(I_k)$. Então calcular as entradas não nulas de m_k é o mesmo que resolver

$$\min_{\hat{m_k}} \|\hat{A}\hat{m_k} - \hat{e_k}\|. \tag{2.4}$$

Como A é não singular $\hat{A}$ deve ter posto completo [14] e então podemos fazer

$$\hat{A} = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{2.5}$$

onde R é uma matriz triangular superior não singular [9] $n_2 \times n_2$, com $n_2 = |J_k|$. Definindo $\hat{c} = Q^T \hat{e_k}$ temos que a solução de (2.4) é dada por $\hat{m_k} = R^{-1}\hat{c}(1 : n_2)$ e assim temos calculados as entradas não nulas de m_k. Fazendo o procedimento para $k = 1, \ldots, n$ temos o precondicionador M.
Como achar um bom padrão de esparsidade para a inversa de A é difícil, foi desenvolvido um modelo de padrão de esparsidade adaptativo. Este modelo começa com um padrão de esparsidade simples, geralmente diagonal, e aumenta sucessivamente este padrão para se obter um melhor precondicionador. Para fazer esta estratégia adaptativa suponha que tenhamos calculado $\hat{m_k}$ para um determinado padrão de esparsidade inicial e considere o

resíduo

$$r_k = A(:, J_k)\hat{m}_k - e_k. \tag{2.6}$$

Se $r = 0$, m_k é exatamente a k-ésima coluna de A^{-1} e nada precisa se feito. Caso $r \neq 0$ considere o conjunto L_k dos índices l tais que $r(l) \neq 0$ e para cada l defina o conjunto Γ_l que consiste dos índices dos elemento não nulos de $A(l, :)$ que não estão em J_k. Os candidatos a novos índices de J_k estão contidos no conjunto

$$\tilde{J}_k = \bigcup_{l \in L_k} \Gamma_l. \tag{2.7}$$

Como queremos um precondicionador esparso, vamos reduzir $\tilde{J}_k$ aos índices que resultam nas maiores reduções em $\|r\|_2$. Para isto, considere para cada $j \in \tilde{J}_k$ o problema de minimização unidimensional

$$\min_{\mu_j} \|r + \mu_j A e_j\|_2, \tag{2.8}$$

para resolver (2.8), considere a seguinte função de μ_j:

$$F(\mu_j) = \|r + \mu_j A e_j\|_2^2 = \|r\|_2^2 + 2\mu_j \langle r, Ae_j \rangle + \mu_j^2 \|Ae_j\|_2^2. \tag{2.9}$$

Derivando (2.9) em relação a μ_j e igualando a zero temos

$$2\mu j \|Ae_j\|_2^2 + 2\langle r, Ae_j \rangle = 0, \tag{2.10}$$

então temos como solução

$$\mu_j = -\frac{r^T A e_j}{\|Ae_j\|_2^2}. \tag{2.11}$$

Computamos para cada j a norma euclidiana ρ_j do novo resíduo $r + \mu_j Ae_j$ com μ_j dado por (2.11):

$$\rho_j^2 = \|r\|_2^2 - \frac{(r^T A e_j)^2}{\|Ae_j\|_2^2}, \tag{2.12}$$

e deixamos em $\tilde{J}_k$ os índices que resultam nos menores valores em (2.12).

Agora definimos o conjunto de índices $\tilde{I}_k$ das linhas não nulas de $A(:, J_k \cup \tilde{J}_k)$ que não estão em I_k, e então resolvemos o problema para $A(I_k \cup \tilde{I}_k, J_k \cup \tilde{J}_k)$, tendo assim uma melhor aproximação m_k da k-ésima coluna de A^{-1}. Este processo é repetido até que a norma do resíduo seja menor que uma tolerância prescrita ou que um máximo de entradas

não nulas em m_k seja alcançado.

A cada vez que o padrão de esparsidade é aumentado é necessário calcular a fatoração QR da matriz $A(I_k \cup \tilde{I}_k, J_k \cup \tilde{J}_k)$. Este problema pode ser minimizado observando que a fatoração pode ser atualizada usando o fato de que

$$A(I_k \cup \tilde{I}_k, J_k \cup \tilde{J}_k) = \begin{pmatrix} \hat{A} & A(I_k, \tilde{J}_k) \\ 0 & A(\tilde{I}_k, \tilde{J}_k) \end{pmatrix} \tag{2.13}$$

$$= \begin{pmatrix} Q & \\ & I_{|\tilde{I}_k|} \end{pmatrix} \begin{pmatrix} R & Q_1^T A(I_k, \tilde{J}_k) \\ 0 & Q_2^T A(I_k, \tilde{J}_k) \\ 0 & A(\tilde{I}_k, \tilde{J}_k) \end{pmatrix} \begin{pmatrix} Q & \\ & I_{|\tilde{I}_k|} \end{pmatrix} \begin{pmatrix} R & B_1 \\ 0 & B_2 \end{pmatrix}, \tag{2.14}$$

com

$$B_1 = Q_1^T A(I_k, \tilde{J}_k) \quad equadB_2 = \begin{pmatrix} Q_2^T A(I_k, \tilde{J}_k) \\ Q_2^T A(I_k, \tilde{J}_k) \end{pmatrix}. \tag{2.15}$$

Então só precisamos computar a fatoração QR reduzida [17] de B_2. Seja

$$B_2 = \hat{Q} \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}, \tag{2.16}$$

então podemos escrever (2.13) como

$$A(I_k \cup \tilde{I}_k, J_k \cup \tilde{J}_k) = \begin{pmatrix} Q & \\ & I_{|\tilde{I}_k|} \end{pmatrix} \begin{pmatrix} I_{|J_k|} & \\ & \hat{Q} \end{pmatrix} \begin{pmatrix} R & B_1 \\ 0 & \hat{R} \\ 0 & 0 \end{pmatrix}. \tag{2.17}$$

Este modelo adaptativo será chamado de **SPAI-A**, enquanto o que usa um padrão de esparsidade fixo será chamado de **SPAI- S**.

2.1.2 MR

O **MR**, **M**inimal **R**esidual [7], é uma alternativa que calcula o precondicionador M resolvendo o problema

$$Am_k = e_k, \ k = 1, \ldots, n, \tag{2.18}$$

a partir de um chute incial M_0, utilizando algumas iterações de um método iterativo de decrescimento de resíduo. Neste caso, utilizando o Minimal Residual, porém é possível utilizar qualquer método de decrescimento de resíduo. Para este método ser eficiente, o método iterativo deve trabalhar no modo esparso, isto é, as colunas de M são armazenadas e operadas como vetores esparsos.

Os chutes iniciais mais comuns são $M_0 = \alpha I$, onde I é a matriz identidade $n \times n$, e $M_0 = \alpha A^T$. O fator α é escolhido como

$$\min_{\alpha} \|I - \alpha A M_0\|_F^2. \tag{2.19}$$

Das propriedades de norma de Frobenius temos

$$\|I - \alpha A M_0\|_F^2 = \mathrm{tr}[(I - \alpha A M_0)^T (I - \alpha A M_0)]. \tag{2.20}$$

Derivando (2.20) em relação a α e igualando a 0 temos:

$$\mathrm{tr}(AM_0)^T - 2\alpha(\mathrm{tr}(AM_0(AM_0)^T)) + \mathrm{tr}(AM_0) = 0, \tag{2.21}$$

como $\mathrm{tr}(AM_0)^T = \mathrm{tr}(AM_0)$ chegamos em

$$\alpha = \frac{\mathrm{tr}(AM_0)}{\mathrm{tr}(AM_0(AM_0)^T)}. \tag{2.22}$$

O chute inicial $M_0 = \alpha A^T$ é recomendado caso A seja muito indefinida. Mesmo com o chute inicial $M_0 = \alpha I$, que é bem esparso, as colunas de M tendem a ficarem densas, para remediar isto é aplicado o procedimento de *numerical dropping*, que consiste em retirar a cada iteração os menores elementos em módulo.

Caso a matriz A seja muito mal condicionada, resolver (2.18) usando um método iterativo terá os mesmos problemas de resolver o sistema original. Entretando é possível fazer um auto precondicionamento no **MR** utilizando iterações internas e externas, isto é, inicia-se com um *loop* (externo) e a cada iteração deste *loop* é feito por completo outro *loop* (interno). Calcula-se um precondicionador com algumas iterações internas e precondiciona-se o sistema para a próxima iteração externa. Vamos denotar por **MR-D** o modelo tradicional e **MR-DP** o modelo com auto precondicionamento.

Também foram implementados modelos do **MR** com padrões de esparsidade fixo e adaptativo. Nestes modelos a esparsidade do precondicionador é baseada na posição do elemento. No padrão fixo, que denotaremos por **MR-S**, são mantidos os elementos que estão nas posições não nulas do padrão predefinido. Já no padrão adaptativo, ao qual denotaremos por **MR-A**, as melhores posições a serem preenchidas são escolhidas similarmente ao **SPAI-A**.

2.2 Inversa aproximada fatorada

Considere que A admite uma fatoração

$$A = LDU, \tag{2.23}$$

onde L é triangular inferior unitária, D é diagonal e U é triangular superior unitária. Então a inversa de A pode ser calculada como

$$A^{-1} = U^{-1}D^{-1}L^{-1} = \bar{Z}\bar{D^{-1}}\bar{W^T}, \tag{2.24}$$

onde $\bar{Z} = U^{-1}$ e $\bar{W} = L^{-T}$ são matrizes triangulares superiores unitárias. Os métodos de inversa aproximada fatorada calculam matrizes esparsas $Z \approx \bar{Z}$ e $W \approx \bar{W}$ e assim constroem o precondicionador

$$M = Z\bar{D}^{-1}W^T \approx A^{-1}. \tag{2.25}$$

Os métodos de inversa aproximada geralmente apresentam maiores reduções no número de iterações dos métodos iterativos, porém diferentemente dos métodos de minimização de norma de Frobenius, têm construção sequencial. Vale ressaltar que, mesmo com Z e W esparsos, o produto (2.25) pode ter um número considerável de elementos não nulos, e em geral se tem precondicionadores mais densos que os calculados pela minimização da norma de Frobenius. Essa maior densidade contribui para a geração de melhores precondicionadores.

O mais conhecido método de inversa aproximada fatorada é o **AINV** [2]. Outro método

estudado foi o **NBIF** [6]. Ambos os métodos serão descritos a seguir.

2.2.1 AINV

O **AINV**, **A**pproximate **INV**erse [2] , calcula as aproximações Z e W através de um algoritmo de biconjugação. Sejam $\{z_i\}_{i=1}^n$ e $\{w_i\}_{i=1}^n$ dois conjuntos de vetores biconjugados a A, isto é

$$w_i^T A z_j = 0 \Leftrightarrow i \neq j. \tag{2.26}$$

Note que de (2.23) temos que $L^{-1}AU^{-1} = D$ e consequentemente L^{-1} e U^{-1} são biconjugados à A.
Definindo agora, as matrizes $Z = [z_1, z_2, \ldots, z_n]$ e $W = [w_1, w_2, \ldots, w_n]$, então temos

$$W^T A Z = D, \tag{2.27}$$

onde $D = diag(p_1, p_2, \ldots, p_n)$, $p_i = w_i^T A z_i \neq 0$. É importante observar que a definição $p_i = w_i^T A z_i$ guarda três importantes informações:

1. w_i não é A-conjugado a z_i.
2. w_i não pertence ao núcleo de A^T, pois caso w_i estivesse no núcleo de A^T teríamos que $e_i = w_i^T A z_i = 0$ o que é absurdo.
3. z_i não pertence ao núcleo de A, pois caso contrário $p_i = w_i^T A z_i = 0$.

Como A é uma matriz não singular segue que as matrizes W e Z são necessariamente não singulares e

$$A^{-1} = Z D^{-1} W^T = \sum_{i=1}^{n} \frac{z_i w_i^T}{p_i}. \tag{2.28}$$

Assim A^{-1} é conhecida se temos calculados dois conjuntos de vetores biconjugados a A. As matrizes Z e W podem ser calculadas explicitamente através de um processo de biconjugação aplicado a quaisquer duas matrizes não singulares $Z^{(0)}, W^{(0)}$, usualmente escolhe-se $Z^{(0)} = W^{(0)} = I$. O processo de biconjugação pode ser visto como um processo de Gram-Schmidt conjugado, isto é, ortogonalização dos vetores unitários com respeito a forma bilinear $\langle x, y \rangle := x^T A y$. Como nosso objetivo é encontrar um precondicionador esparso, o processo de biconjugação é feito de maneira incompleta retirando-se das colunas

z_i e w_i os menores elementos em módulo (*numerical dropping*). Caso o processo seja efetuado sem *numerical dropping* chegaremos exatamente nos fatores $Z = U^{-1}$ e $W = L^T$.

2.2.2 NBIF

O **NBIF**, **N**onsymmetric **B**alanced **F**actorization [6] é uma extensão do **B**alanced **I**mproved **F**actorization, denotado por **BIF**, que aproxima o fator de Cholesky de uma matriz positiva definida através da fórmula de Sherman-Morrison-Woodbury (ISMW) [9] [13].

Considere que A pode ser escrita como

$$A = A_0 + \sum_{k=1}^{n} x_k y_k^T \tag{2.29}$$

onde A_0 é não singular e $\{x_k\}_{k=1}^n$ e $\{y_k\}_{k=1}^n$ são dois conjuntos de vetores em $\mathbb{R}^n$. Usando a ISMW temos que a inversa de A é dada por

$$A^{-1} = A_0^{-1} - A_0^{-1} U_{A_0} D_{A_0}^{-1} V_{A_0}^T A_0^{-1}, \tag{2.30}$$

onde U_{A_0} e V_{A_0} tem colunas u_k e v_k dadas por

$$u_k = x_k - \sum_{i=1}^{k-1} \frac{v_i^T A_0^{-1} x_k}{r_i} u_i \text{ e } v_k = y_k - \sum_{i=1}^{k-1} \frac{y_k^T A_0^{-1} u_i}{r_i} v_i, \tag{2.31}$$

respectivamente, e $D_{A_0} = diag(r_1, r_2, \ldots, r_n)$, $r_i = 1 + y_k^T A_0^{-1} u_k = 1 + v_k^T A_0^{-1} x_k$ para $k = 1, \ldots, n$.
Escolhendo por simplicidade

$$A_0 = sI,\ s > 0,\ x_k = e_k,\ y_k = (a^k - a_0^k)^T,$$

onde a^k e a_0^k denotam a k-ésima linha das matrizes A e A_0, respectivamente. Substituindo os valores de x_k e y_k em (2.31) temos

$$u_k = e_k - \sum_{i=1}^{k-1} \frac{v_i^T e_k}{s r_i} u_i \text{ e } v_k = y_k - \sum_{i=1}^{k-1} \frac{y_k^T z_i}{s r_i} v_i, \tag{2.32}$$

e então obtemos

$$A^{-1} = s^{-1}I - s^{-2}U_s D_s^{-1} V_s^T. \tag{2.33}$$

Os próximos resultados vão nos permitir associar (2.33) a uma fatoração LDU de A.

Lema 2.2.1 *Sejam u_k, v_k e r_k os vetores computados pela ISMW para algum $s > 0$. Sejam u_k^*, v_k^* e r_k^* os vetores computados pela ISMW para $s = 1$. Então,*

$$u_k = u_k^*,\ v_k = v_k^* - (s-1)w_k \text{ e } r_k = s^{-1}r_k^*, \tag{2.34}$$

onde

$$w_k = x_k - \sum_{i=1}^{k-1} \frac{y_k^T u_i}{r_i^1} w_i. \tag{2.35}$$

Prova:Por indução.

Para $k = 1$, pela definição 2.32 temos que $u_1 = x_1 = u_1^*$.

De 2.32 e 2.30 temos que

$$v_1 = y_1 = (a^1 - se^1)^T = (a^1 - e^1)^T - (s-1)e_1 = y_1^* - (s-1)e_1 = v_1^* - (s-1)w_1.$$

Além disso,

$$\begin{aligned} r_1 &= 1 + s^{-1}y_1^T u_1 = 1 + s^{-1}(y_1^* - (s-1)e_1)^T u_1^* \\ &= 1 + s^{-1}((y_1^*)^T u_1^* - (s-1)) \\ &= s^{-1}(1 + (y_1^*)^T u_1^*) = s^{-1}r_1^*. \end{aligned} \tag{2.36}$$

Vamos supor agora, que as sentenças são verdadeiras para $i = 2, \ldots, k-1$. De (2.35) temos $(w_k)_i = 0$ se $i < k$, e $(v_k)_i = (v_k^*)_i$ para $i < k$. Então, de (2.32) temos $u_k = u_k^*$.

Observe agora, que $y_k^T u_i = (y_k^*)^T u_i^*$, para $i < k$. De (2.32)

$$\begin{aligned} v_k &= (y_k^* - (s-1)e_k) - \sum_{i=1}^{k-1} \frac{(y_k^*)^T u_i^*}{r_i^*}(v_i^* - (s-1)w_i) \\ &= \left(y_k^* - \sum_{i=1}^{k-1} \frac{(y_k^*)^T u_i^*}{r_i^*} v_i^*\right) - (s-1)\left(e_k \sum_{i=1}^{k-1} \frac{(y_k^*)^T u_i^*}{r_i^*} w_i\right) \\ &= v_k^* - (s-1)w_k. \end{aligned} \tag{2.37}$$

Por último, $r_k = s^{-1} r_k^*$ segue como em 2.36.

O Lema 2.2.1, pode ser visto matricialmente como

$$\begin{aligned} U_s &= U_1 \\ V_s &= V_1 - (s-1)W \\ D_s &= s^{-1} D_1, \end{aligned}$$

onde U_s, V_s e D_s são as matrizes calculadas pela ISMW para algum $s > 0$, U_1, V_1 e D_1 são as matrizes calculadas pela ISMW para $s = 1$ e W é a matriz cujo as colunas são os vetores $w_k,\ k = 1 \ldots, n.$ definidos por (2.35).

O próximo teorema relaciona a estrutura da matriz V_s com os fatores da U_1 e D_1 e a matriz W.

Teorema 2.2.1 *Seja $A^{-1} = s^{-1}I - s^{-2}U_s D_s^{-1} V_s^T$ para algum $s > 0$. Então*

$$V_s^T = DU^{-1} - sW^T. \tag{2.38}$$

Prova:Do Lema 2.2.1 e de (2.33) temos

$$\begin{aligned} s^{-1}I - A^{-1} &= s^{-2}U_s D_s^{-1} V_s^T = U_1(s^{-1}D_s^{-1})(s^{-1}V_s^T) \\ &= U_1 D_1(s^{-1}V_1^T - (1 - s^{-1})W^T) \end{aligned} \tag{2.39}$$

Fazendo o limite com $s \to \infty$, obtemos

$$A^{-1} = U_1 D_1^{-1} W^T. \tag{2.40}$$

De (2.33) e (2.40) temos que

$$U_1 D_1^{-1} W^T = s^{-1} I - U_1 D_1^{-1} V_s^T.$$

Isto é,

$$U_1 D_1^{-1} V_s^T = I - s U_1 D_1^{-1} W^T,$$

e consequentemente

$$V_s^T = D_1 U_1^{-1} - s W^T. \tag{2.41}$$

O coralário que vamos enunciar abaixo, relaciona a ISMW com uma fatoração LDU de A.

Corolário 2.2.1 *Supondo que existe a ISMW exata (2.30) para algum s e que pode ser fatorado como $A = LDU$. Além disso, seja W definido como em (2.35), então*

$$L = W^{-T} \; DU = D_1 U_1^{-1}, \tag{2.42}$$

e ainda

$$D = D_1 \; e \; U = U_1^{-1}.$$

Prova:Note que de (2.40) temos

$$A = W^{-T} D U^{-1}.$$

Então o resultado segue imediatamente da unicidade das fatorações incompletas de A e do Teorema (2.2.1).

Pelo Corolário 2.2.1 e o Teorema 2.2.1 temos que

$$V_s = U^T D - s L^T$$

e caso A seja simétrica positiva definida temos

$$V_s = LD - s L^{-T}.$$

Então, ao calcularmos a ISMW de uma matriz SPD calculamos ao mesmo tempo os fatores, D, L e L^{-1}, e esta é a grande vantagem do **BIF**. O **BIF** é um algoritmo que calcula aproximações $\hat{L}^{-1}$ e $\hat{D}$, dos fatores L^{-1} e D de uma fatorção LDL^T de uma matriz SPD A. As aproximações são calculadas utilizando incompletamente (2.31) para $k = 1, \ldots, n$. Ambos os fatores $\hat{L}^{-1}$ e $\hat{D}$ são armazenados na matriz $\hat{V}_s$ que é uma aproximação de V_s. A esparsidade é controlada por uma estratégia de *numerical dropping* acoplado, esta estratégia proposta em [4], mostra que é justificável retirar as entradas $\hat{l}_{ik}$ de uma coluna k de $\hat{L}^{-1}$ se

$$|\hat{l}_{ik}|\|e_k^T \hat{L}\| \leq \epsilon, \tag{2.43}$$

onde $\hat{L}$ é uma aproximação de L e ϵ é uma tolerância prescrita. O *numerical dropping* acoplado se mostra robusto em retirar as entradas que contribuem pouco para a qualidade do precondicionador. Infelizmente se A for não simétrica esta estratégia não pode ser aplicada, pois não se tem simultaneamente os fatores U, U^{-1}, L e L^{-1}. O *improved balanced factorization* busca uma alternativa onde se tenha os quatro fatores e assim possamos realizar o *numerical dropping* aclopado. Para se ter os quatro fatores se calcula a ISMW de A^T, e assim na estrutura da matriz $\bar{V}_s = LD - sU^{-1}$ temos os fatores que faltavam. Denotaremos este método por **NBIF**, onde **N** quer dizer **nonsymmetric** (não simétrico).

2.3 Inversa aproximada dos fatores LU incompletos

Ainda existe uma terceira classe de métodos que constroem inversas aproximadas fatoradas a partir de uma fatoração incompleta LU de A. Primeiramente calcula-se uma fatoração incompleta $A \approx \bar{L}\bar{U}$ usando as técnicas usuais e depois estes fatores são aproximadamente invertidos.

Assumindo que os fatores incompletos $\bar{L}$ e $\bar{U}$ foram calculados, os fatores inversos aproximados podem ser calculados resolvendo inexatamente os problemas triangulares

$$\bar{L}x_i = e_i, \ \bar{U}y_i = e_i \ i = 1, \ldots, n. \tag{2.44}$$

Este problema pode ser resolvido pelo **SPAI** a partir de um dado padrão de esparsidade ou de maneira mais efetiva por substituição para frente e para trás. Este tipo de método não são utilizados pois não apresentam nenhuma vantagem, tendo em vista que esses métodos assumem que uma fatoração LU incompleta tenha sido calculada. Isto implica

que estes método sequer é aplicável se essa fatoração LU incompleta não existir ou for instável. Além disso, esta dependência limita o paralelismo já que os métodos de fatoração LU incompletos são sequenciais.

3 ASPECTOS COMPUTACIONAIS

Neste capítulo vamos falar sobre a implementação dos algoritmos apresentados. Serão apresentados pseudo códigos e algumas propriedades importantes, como a complexidade (em serial) de alguns códigos, também serão abordados.

Antes de falarmos dos algoritmos de cáculo da inversa, vamos introduzir a rotina auxiliar `sparse-sparse`, esta rotina é essencial para que alguns métodos apresentados sejam efetivos, pois reduz o tamanho do problema.

```
function [A,b] = sparse_sparse(cols,A,b)
A = A(:,cols);        % only the columns in cols
[rows,~] = find(A);
rows = unique(rows); % nonzero rows
A = A(rows,:);
b = b(rows);
end
```

3.1 SPAI

O `SPAI-S` é bem simples de ser implementado e tem como parâmetros de entrada apenas a matriz A e o padrão de esparsidade para o precondicionador. Infelizmente, encontrar um bom padrão de esparsidade para a inversa de A é uma tarefa difícil e essencial para eficiência do `SPAI-S`. Pode se usar a série de Neumann para inversa da matriz

$$(I - A)^{-1} = \sum_{k=0}^{\infty} A^k. \tag{3.1}$$

Por (3.1) vemos que existe uma relação entre a inversa e as potências de A, então é justificável utilizar, para algum inteiro positivo l, o padrão de esparsidade de A^l. Quanto maior for l melhor o precondicionador tende a ser, mas note que mesmo A sendo muito esparsa suas potências podem ficar extremamente densas, o que não é desejado. Além disso, calcular tais potências pode ser algo inviável dependendo do tamanho do problema. Por simplicidade utilizamos em nossos experimentos o mesmo padrão de A para o

precondicionador.

```
function M = spai_s(A,S)
n = length(A); I = speye(n);
for j = 1:n
    ej = I(:,j);            % j-th column of identity
    rows = find(S(:,j));    % indices of nonzero rows
    [ss_A,ss_ej]=sparse_sparse(rows,A,ej);
    ss_mj=ss_A\ss_ej;       % solves the system
    M(rows,j) = drop(ss_mj); % update the j-th column of M
end
end
```

O SPAI-A foi implementado começando com um padrão de esparsidade diagonal. Além disso, são necessários os parâmetros nmax, que regula o máximo de elementos não nulos do precondicionador M, innz, que especifica quantos elementos não nulos serão adicionados por iteração e a tolerância residual tol. Os valores utilizados em nossos testes foram innz $= 1$ e tol $= 0.1$ e o valor nmax foi escolhido de tal forma que o precondicionador tenha no máximo duas vezes o número de não zeros que a matriz A.

```
function M = spai_a(A,nmax,innz,tol)
n = length(A); I = speye(n); M = I;
Q = []; R = [];                  % QR factors of ss_A
iloop = floor(nmax/innz);    % inner loop size
for j = 1:n
    ej = I(:,j); rows = j;       % initial sparsity pattern
    for i=1:iloop
        % calculates the residue (full mode)
        rj = ej - A * M(:,j);    % residue
        if(norm(rj) < tol) break; end

        % sparsity pattern update
        rows = row_adaptive(A,M,rows);
        [ss_A,ss_ej] = sparse_sparse(rows,A,ej);

```

```
16          [Q,R] = qr_grote(Q,R,ss_A); % QR update
17          ss_mj = R\(Q'*ss_ej);       % solves the system
18          M(rows,j) = drop(ss_mj);    % M update
19      end
20  end
21  end
```

Todas as variações do **SPAI** são invariantes por permutações, isto é, se calcularmos o precondicionador utilizando qualquer uma dessas variações para P_1AP_2, onde P_1 e P_2 são matrizes de permutação, encontraremos $P_2^T M P_1^T$ como precondicionador.

3.2 MR

Assim como o `SPAI-S`, o `MR-S` tem a implementação bem simples. Além do padrão de esparsidade o código precisa de duas tolerâncias `tol1` e `tol2`, ambos critérios de parada para o algoritmo, e um número máximo de iterações `maxiter`. Os valores utilizados foram `tol1` = `tol2` = 0.1 e `maxiter` = 30.

```
1   function M = mr_s(A,S,tol1,tol2,maxiter)
2   n = length(A); I = speye(n);
3   scale = trace(A)/norm(A,'fro')^2; % min||I-A*(alpha*I)||_F
4   for j = 1:n
5       ej = I(:,j);
6       rows = find(S(:,j));    % indices of nonzero rows
7       mj = scale * ej;        % initial solution
8
9       % adjusts A to the sparse-sparse mode
10      [ss_A,ss_ej] = precond.sparse_sparse(jid,A,ej);
11      ss_mj = mj(rows);
12      for k=1:maxiter
13          ss_rj=ss_ej-ss_A*ss_mj;
14          if(norm(ss_rj) < tol1) break; end
15
16          % steepest descent method
17          gj = ss_A'*ss_rj;
18          Agj = ss_A*gj;
```

```
        nrm_Agj = norm(Agj);
        if(nrm_Agj < tol2) break; end
        alpha = (norm(gj)/nrm_Agj)^2; % step size
        ss_mj = ss_mj + alpha*gj;
    end
    M(rows,j) = ss_mj; % update M
end
end
```

Na implementação do `MR-A`, o padrão de esparsidade utilizado como inicial foi o padrão diagonal, além disso são necessários o critério de parada `restol` e o número máximo de iterações `maxiter`. Foram utilizados os valores, `restol` $= 0,1$ e `maxiter` $=$ 30.

```
function M = mr_a(A,restol,maxiter)
n = length(A); I = eye(n);
scale = trace(A)/norm(A,'fro')^2; % min ||I-A*(alpha*I)||_F
for j = 1:n
    ej = I(:,j);
    mj = scale * ej;              % initial guess
    rows=j;                       % set sparsity pattern
    for i=1:iloop
        % sparsity pattern update
        rows = row_adaptive(A,M,rows);
        [ss_A,ss_ej] = sparse_sparse(rows,A,ej);
        ss_mj = mj;
        for k=1:maxiter
            ss_rj = ss_ej - ss_A * ss_mj;
            if(norm(ss_rj) < restol) break; end
            % using steepest descent method
            gj  = ss_A' * ss_rj;
            Agj = ss_A * gj;
            nrm_Agj=norm(Agj);
            if(nrm_Agj < 1E-8)
                break;
            end
            alpha = (norm(gj)/nrm_Agj)^2;
```

```
24                ss_mj = ss_mj + alpha * gj;
25            end
26            M(rows,j)=ss_mj;
27        end
28    end
29    end
```

O MR-D, que utiliza *numerical dropping* ao invés de padrão de esparsidade, utiliza quatro parâmetros de entrada: a tolerância do processo de *numerical dropping* `droptol`, a tolerância da norma do resíduo `restol`, o valor `nmax` que regula o máximo de elementos não nulos do precondicionador M e o número máximo de iterações `n_inner`. Em nossos experimentos a matriz foi dividida pelo seu maior elemento em módulo e $0,01$ foi usado como `droptol` $0,1$ foi utilzado como valor de `restol`. O valor de `nmax_col` foi escolhido de tal forma que o precondicionador tivesse no máximo 2 vezes o número de não zeros da matriz A e 30 foi o valor escolhido para `n_inner`.

```
1   function M = mr_d(A,droptol,tol,nmax,n_inner)
2   n = length(A); I = eye(n);
3   scale = trace(A)/norm(A,'fro')^2; % min||I-A*(alpha*I)||_F
4   for j=1:n
5       ej = I(:,j);
6       mj = scale * ej; % initial guess
7       for k = 1:n_inner
8           r = ej - A * mj;
9           if(norm(r)<restol) break; end
10          Ar = A*r;
11          alpha = (r'*Ar)/(Ar'*Ar);
12          mj = mj + alpha * r;
13          mj = drop(mj,nmax,droptol);
14      end
15      M(:,j) = mj;
16  end
17  end
```

A versão auto precondicionada, MR-DP utliza os parâmetros do MR-D e mais um parâmetro `n_outer` que regula o número de iterações externas. Nos experimentos esco-

lhemos `n_outer` = 5 e os outros valores utilizados foram os mesmos do `MR-D`.

```
1  function M = mr_dp(A,droptol,tol,nmax,n_outer,n_inner)
2  n = length(A); I = speye(n);
3  scale = trace(A)/norm(A,'fro')^2; % min||I-A*(alpha*I)||_F
4  for i=1:n_outer
5      for j=1:n
6          ej = I(:,j);
7          mj = scale * ej;
8          for k = 1:n_inner
9              r = ej - A * mj;
10             if(norm(r) < tol) break; end
11             z = M*r; % self-preconditioning step
12             q = A*z;
13             alpha = (r'*q)/(q'*q);
14             mj = mj+alpha*z;
15         end
16         mj = drop(mj,nmax_col,droptol);
17         M(:,j) = mj;
18     end
19 end
20 end
```

3.3 AINV

Em aritmética exata o processo de biconjugação não tem divisão por zero [2] se e somente se A admite uma fatoração LU, ou no nosso caso LDU, mas devido ao algoritmo de biconjugação no `AINV` ser feito de maneira incompleta é possível haver divisão por zero se A não for uma H-matriz. Uma matriz $A = [a_{ij}]$ é uma H-matriz se a matriz $\hat{A} = [\hat{a}_{ij}]$, onde

$$\hat{a}_{ij} = \begin{cases} -|a_{ij}| \text{ se } i \neq j \\ a_{ij} \text{ se } i = j \end{cases},$$

tem autovalores com parte real positiva.

Como estamos tratando de matrizes genéricas, é feita uma análise nas linhas 9 e 10 do

código que substitui os números menores que 10^{-8} por 1. Como parâmetros de entrada o `AINV` tem o `droptol` e `nmax`, ambos ligados à esparsidade da matriz. O valor `droptol` é a tolerância para a retirada de elementos das colunas de Z e W, seguindo as recomendações de [2], todos os elementos de A foram divididos pelo maior valor em módulo da mesma. Assim temos $|a_{ij}| \leq 1$ e `droptol` $= 0,1$ é um bom valor. O parâmetro `nmax` regula o máximo de elementos não nulos do precondicionador M. Em nossos testes escolhemos `nmax` de tal forma que M tivesse no máximo o dobro de entradas não nulas de A.

```
function [Z,D,W] = ainv(A,droptol,nmax)
n = length(C);
W = speye(n); Z = W;
for i = 1:n
    p(i:n)=A(:,i)' * Z(:,i:n);
    q(i:n)=C(:,i)' * W(:,i:n);

    % eliminates the possibility of division by zero
    if(abs(p(i))<1E-8) p(i)=1; end
    if(abs(q(i))<1E-8) q(i)=1; end
    if(i == n) break; end

    % update Z and W
    for j = i+1:n
       Z(:,j) = Z(:,j) - (p(j)/p(i))*Z(:,i);
       W(:,j) = W(:,j) - (q(j)/q(i))*W(:,i);
    end

    % drop
    for j = i+1:n
        Z(:,j) = drop(Z(:,j),nmax,droptol,j);
        W(:,j) = drop(W(:,j),nmax,droptol,j);
    end
end
D = sparse(1:n,1:n,1./p);
end
```

3.4 NBIF

O `NBIF` foi implementado praticamente igual ao proposto em [6], a única diferença é que lá as saídas são os fatores L, D e U e no nosso os fatores aproximados L^{-1} e U^{-1} também são argumentos de saída. Relembrando que o `NBIF` calcula os 4 fatores L, U, L^{-1} e U^{-1}. Como entradas o `NBIF` precisa da matriz A, as tolerâncias `dropv` e `dropz` utilizadas no *dropping*, e um vetor S, cujas entradas são maiores que zero. Este vetor S é necessário pois o `NBIF` utiliza uma matriz diagonal ao invés de apenas um escalar s. Em nosso experimentos usamos S como um vetor com todas entradas iguais a $0,5$ e `dropv` $=$ `dropz` $= 0,1$.

```
function [L,D,U,W,Z] = nbif(A, S, dropv, dropz)
n = size(A, 1); Z  = speye(n); Zt = speye(n);
V  = tril(A') - spdiags(S,0,n,n); % Matrix V
Vt = tril(A) - spdiags(S,0,n,n);  % Matrix Vt
nrm_L = zeros(n,1);       % Norms of rows of L
nrm_U = zeros(n,1);       % Norms of rows of L of A'
nrm_invL = zeros(n,1);    % Norms of rows of inv(L)
nrm_invU = zeros(n,1);    % Norms of rows of inv(L) of A'
D  = zeros(n,1);  Dt = zeros(n,1); % Diagonal

for k = 1:n
    for i = 1:(k-1) % update
        mult_invU = (A(k,:)*Z(:,i))/(D(i)*S(i));
        mult_invL = (A(:,k)'*Zt(:,i))/(Dt(i)*S(i));
        mult_L = Vt(k,i)/(Dt(i)*S(i));
        mult_U = V(k,i)/(D(i)*S(i));
        V(1:i-1,k)  = V(1:i-1,k) - mult_L*V(1:i-1,i);
        V(i,k)    = V(i,k) + S(i)*mult_invU;
        V(kid,k)  = V(kid,k) - mult_invU*V(kid,i);
        Vt(1:i-1,k) = Vt(1:i-1,k) - mult_U*Vt(1:i-1,i);
        Vt(i,k)   = Vt(i,k) + S(i)*mult_invL;
        Vt(kid,k) = Vt(kid,k) - mult_invL*Vt(kid,i);
        Z(:,k)  = Z(:,k)-(V(k,i)/(D(i)*S(i)))*Z(:,i);
        Zt(:,k) = Zt(:,k)-(Vt(k,i)/(Dt(i)*S(i)))*Zt(:,i);
    end
```

```
26      D(k) = V(k,k)/S(k) + 1.0;
27      Dt(k) = Vt(k,k)/S(k) + 1.0;
28      nrm_invL(k) = sqrt(1.0+norm(V(1:k-1,k)+S(k)^2))/S(k);
29      nrm_invU(k) = sqrt(1.0+norm(Vt(1:k-1,k)+S(k)^2))/S(k);
30      temp = 1.0/((D(k)*S(k))^2);
31      tempt = 1.0/((Dt(k)*S(k))^2);
32      nrm_L(k+1:n) = nrm_L(k+1:n)+temp*(V(k+1:n,k).^2);
33      nrm_U(k+1:n) = nrm_U(k+1:n)+tempt*(Vt(k+1:n,k).^2);
34      nrm_L(k) = sqrt(nrm_L(k) +1.0);
35      nrm_U(k) = sqrt(nrm_U(k) +1.0);
36
37      % standard dropping in Z and Zt
38      Z(1:k-1,k) = Z(1:k-1,k).* (abs(Z(1:k-1,k)) > dropz);
39      Zt(1:k-1,k) = Zt(1:k-1,k).* (abs(Zt(1:k-1,k)) > dropz);
40
41      % BIF dropping in V and Vt
42      V(1:k-1,k) = V(1:k-1,k) .*((abs(V(1:k-1,k))...
43          > dropv./nrm_U(1:k-1))); % L --> L^-1
44      V(k+1:n,k) = V(k+1:n,k) .*((abs(V(k+1:n,k))...
45          > dropv*D(k)/nrm_invU(k))); % U^-1 --> U
46      Vt(1:k-1,k) = Vt(1:k-1,k).*((abs(Vt(1:k-1,k))...
47          > dropv./nrm_L(1:k-1))); % U --> U^-1
48      Vt(k+1:n,k) = Vt(k+1:n,k).*((abs(Vt(k+1:n,k))...
49          > dropv*D(k)/nrm_invL(k))); % L^-1 --> L
50  end
51  % Set the results
52  D  = diag(V)+S; Dt = diag(Vt)+S;
53  U  = ((tril(V)+spdiags(S,0,n,n))/spdiags(D,0,n,n))';
54  L  = (tril(Vt)+spdiags(S,0,n,n))/spdiags(Dt,0,n,n);
55  W  = (spdiags(S,0,n,n) - triu(V,+1))';
56  end
```

4 EXPERIMENTOS NUMÉRICOS

Neste capítulo vamos apresentar uma série de experimentos numéricos com os algoritmos testados. Os algoritmos `SPAI-S`, `SPAI-A`, `MR-S MR-A`, `MR-D` e `MR-DP` foram implementados em MATLAB. Já o `AINV` e `NBIF` forma implementados em linguagem C. O algoritmo `SPAI` foi implementado em C por T. Huckle e pode ser encontrado em [12]. Em todos os testes o solver utilizado foi o GMRES [21]. Para efeito de comparação foram realizados testes sem precondicionamento e com precondicionadores baseados na fatoração LU incompleta que são bastantes utilizados hoje, no caso o `ILU(level)` e o `ILUT` com estratégia de *dropping*. Em nossos experimentos foram usados `ILU(0)` e `ILUT` com *threshold* igual a 0.01. O tempo de construção e aplicação do precondicionador não foi levado em consideração, pois os códigos estão implemnetados sequencialmente e a grande vantagem da inversa aproximada é o paralelismo. Além disso, alguns códigos estão em MATLAB, que não é a melhor linguagem em relação a tempos de CPU.

4.1 Testes Preliminares

Os primeiros testes realizados visavam avaliar a eficácia da inversa aproximada em reduzir o número de iterações do GMRES. Os testes foram feitos utilizando-se matrizes do Matrix Market [18], todas oriundas de simulações de reservatório de petróleo.

Nos testes foi utilizado precondicionamento à direita, o lado direito foi tal que o vetor formado apenas por 1 fosse solução exata do sistema. O critério de parada foi que o resíduo relativo fosse menor que 10^{-6} ou que um máximo de 1000 iterações do GMRES fosse atingido. A Tabela 1 mostra as matrizes testadas, o tamanho n e o número nnz de elementos não nulos na mesma.

A Tabela 2 mostra o resultado dos testes realizados sem precondicionamento. Os resultados deixam bem claro a necessidade de precondicionamento para que o GMRES seja eficiente. A Tabela 3 e a Tabela 4 mostram os resultados utilizando os precondicionadores padrão de fatoração LU incompleta e utilizando os precondicionadores baseados na aproximação da inversa. O valor $nnz(M)/nnz(A)$ denota a razão entre o número de não zeros do precondicionador e o número de não zeros da matriz utilizada. Os precondicionadores baseados na inversa aproximada mostraram-se eficientes em reduzir o número de iterações

Tabela 1 Tamanho e número de elementos não nulos das matrizes testadas.

MATRIZ	n	$nnz(A)$
ORSIRR_1	1030	6858
ORSIRR_2	886	5970
ORSREG_1	2205	14133
SHERMAN1	1000	3750
SAYLR_3	1000	3750
PORES_1	30	180

Tabela 2 Testes sem precondicionamento.

MATRIZ	Iterações
ORSIRR_1	—
ORSIRR_2	—
ORSREG_1	259
SHERMAN1	—
SAYLR_3	—
PORES_1	27

do GMRES mas mostraram-se, no geral, inferiores ao `ILU(0)` e `ILUT`. Vale ressaltar que os precondicionadores de inversa aproximada foram capazes de resolver o sistema para as matrizes `SAYLR_3` e `PORES_1`, matrizes as quais o `ILU(0)` e `ILUT` não conseguiram ser eficientes.

4.2 Testes com Problemas de Discretização

Nesta seção vamos analisar o desempenho dos precondicionadores de inversa aproximada em três problemas de discretização. Os dois primeiros baseados na modelagem de um fluído imcompressível em um meio poroso tridimensional, com discretização de diferenças finitas do operador diferencial $\nabla \cdot K\nabla$, impondo condições de contorno de Dirilichet. O primeiro modelo assume um campo K homogêneo e isotrópico, já o segundo assume um campo heterogêneo. Para o modelo heterogêneo, foi gerado um "ladrilho" de tamanho 12×12 de valores para K, esse ladrilho é a realização de uma distribuição estacionária normal logarítmica com uma função de covariancia Gaussiana anisotrópica ([19], Seção 5.4). Cada camada de K foi criada por um ladrilhamento utilizando os ladrilhos 12×12, com algumas reflexões para manter a suavidade. Todas as camadas foram feitas identicamente, resultando num campo heterogêneo e isotrópico.

Tabela 3 Resultados com as matrizes do Matrix Market.

	Iterações			**nnz(M)/nnz(A)**		
MÉTODO	ORSIRR_1	ORSIRR_2	ORSREG_1	ORSIRR_1	ORSIRR_2	ORSREG_1
ILU(0)	48	46	49	1	1	1
ILUT	44	43	45	0.58	0.61	0.61
SPAI-S	135	138	117	1	1	1
SPAI-A	53	53	48	1.25	1.21	1.30
MR-S	479	381	185	1	1	1
MR-A	—	—	238	1.65	1.63	1.72
MR-D	179	244	170	0.30	0.30	0.31
MR-DP	—	—	—	1.91	1.93	1.87
AINV	47	46	45	2.22	2.20	2.38
NBIF	46	46	44	0.97	0.99	1.01

Tabela 4 Resultados com as matrizes do Matrix Market.

	Iterações			**nnz(M)/nnz(A)**		
MÉTODO	SHERMAN1	SAYLR_3	PORES_1	SHERMAN1	SAYLR_3	PORES_1
ILU(0)	—	—	—	1	1	1
ILUT	12	—	—	1.58	1.58	0.78
SPAI-S	82	—	18	1	1	1
SPAI-A	59	59	16	1.17	1.17	1.06
MR-S	221	221	17	1	1	1
MR-A	366	363	21	1.17	1.17	1.80
MR-D	243	—	7	0.45	0.45	0.33
MR-DP	134	—	7	1.36	1.36	0.73
AINV	44	490	18	2.23	1.09	2.69
NBIF	33	—	15	1.96	1.96	1.78

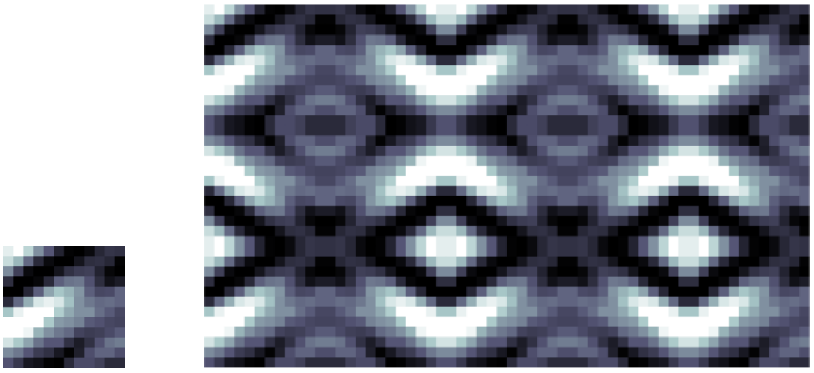

Figura 1 Na esquerda, o ladrilho básico 12×12 usado nos experimentos com campo heterogêneo. Na direita a composição de 3×5 ladrilhos refletidos.

Tabela 5 Tamanho da matriz A e o número de elementos não nulos para todos os problemas.

TODOS OS PROBLEMAS	n	$nnz(A)$
$12 \times 2 \times 12 = 12^3$	1728	11232
$36 \times 36 \times 36 = 36^3$	46656	318816
$48 \times 48 \times 48 = 48^3$	110592	760320

O ladrilho e o ladrilhamento 3×5 resultantes são mostrados na Figura 1.

O terceiro experimento é baseado na discretização do operador $L(u) = \nabla.(u\vec{v}) - \epsilon\Delta u$, onde $\vec{v}$ é um campo de velocidade incompressível, relacionado com a equação de convecção e difusão.

A Tabela 5 mostra o tamanho das matrizes n e o número de elementos não nulos nnz, para três tamanhos diferentes de malhas. A Tabela 6 e a Tabela 7 mostram os resultados para os testes com o problema homogêneo e heterogêneo, respectivamente. Os testes foram realizados utilizando o $GMRES$, o lado direito foi escolhido de tal forma que a solução exata x^* seja o vetor formado só por 1. O critério de parada foi que o resíduo relativo $\|b - Ax\|/\|b\|$ fosse inferior a 10^{-7} ou que um máximo de 1000 iterações fosse atingido. O valor `iGMRES` representa a razão entre o número de iterações do $GMRES$ sem precondicionamento e o número de iterações do $GMRES$ precondicionado. O valor $nnz(M)/nnz(A)$ representa a razão entre o número de elementos não nulos de A e M.

A Figura 2 e a Figura 3 mostram a melhoria nas iterações do GMRES para o problema homegêneo e heterogêneo, respectivamente. A Tabela 8 mostra os resultados para o problema de convecção e difusão para $\epsilon = 10^{-2}$. Os parâmetros dos testes foram os mesmos utilizados no problema homogêneo e heterogêneo. Já a Figura 4 mostra a

Tabela 6 Resultados do problema Homogêneo.

	iGMRES			**nnz(M)/nnz(A)**		
MÉTODO	12^3	36^3	48^3	12^3	36^3	48^3
GMRES	1.00	1.00	1.00	0.00	0.00	0.00
ILU(0)	3.58	5.37	5.66	1.00	1.00	1.00
ILUT	6.14	9.47	10.55	1.52	1.55	1.55
SPAI	2.53	3.35	3.41	2.83	2.86	2.86
SPAI-S	1.79	1.92	2.04	1.00	1.00	1.00
SPAI-A	2.15	2.33	2.39	0.92	0.88	0.87
MR-S	1.79	1.92	2.00	1.00	1.00	1.00
MR-A	1.65	1.81	2.00	1.69	1.61	1.60
MR-D	2.26	—	—	2.00	—	—
MR-DP	1.79	—	—	2.00	—	—
AINV	2.53	2.73	2.83	2.15	2.39	2.42
NBIF	2.53	2.73	2.83	1.08	1.12	1.13

Tabela 7 Resultados do problema Heterogêneo.

	iGMRES			**nnz(M)/nnz(A)**		
MÉTODO	12^3	36^3	48^3	12^3	36^3	48^3
GMRES	1.00	1.00	1.00	0.00	0.00	0.00
ILU(0)	30.54	25.64	19.61	1.00	1.00	1.00
ILUT	44.11	43.48	34.48	1.45	1.49	1.50
SPAI	19.85	17.86	13.70	2.81	2.85	2.85
SPAI-S	12.81	10.75	8.33	1.00	1.00	1.00
SPAI-A	15.27	13.33	9.71	0.92	0.88	0.87
MR-S	11.03	9.71	7.75	1.00	1.00	1.00
MR-A	6.73	3.45	1.28	1.69	1.61	1.60
MR-D	18.90	—	—	2.00	—	—
MR-DP	16.54	—	—	2.00	—	—
AINV	17.26	14.08	11.24	2.15	2.41	2.45
NBIF	17.26	15.15	11.63	1.13	1.21	1.22

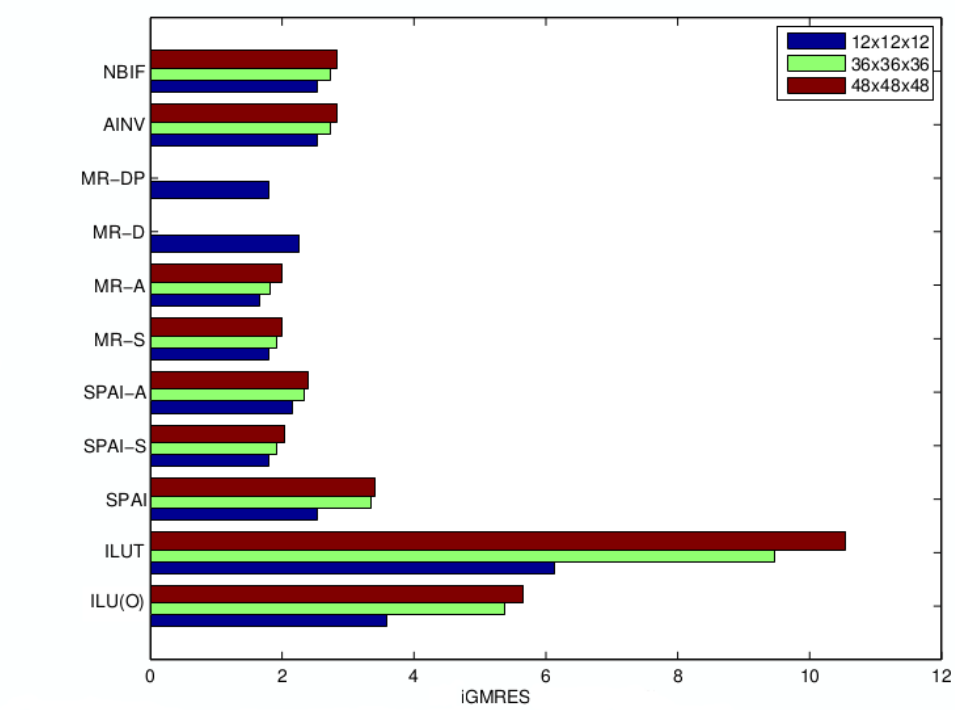

Figura 2 Melhoria no número de iterações do GMRES para o problema Homogêneo.

melhoria nas iterações do GMRES para o problema de Convecção e Difusão.

Na Figura 5, Figura 6, Figura 7 e Figura 8 são mostrados a distribuição de autovalores da matriz gerada no problema Laplaciano Homogêneo para a malha $12 \times 12 \times 12$ sem precondicionamento, precondicionada com o `ILUT`, `SPAI-S` e `AINV`, respectivamente. Por último a Figura 9 mostra a melhora no número de condicionamento do mesmo problema. Note que apesar do `MR-D` e `MR-DP` terem reduzido mais o número de condicionamento que outros métodos estes métodos não foram efetivos na resolução do sistema. Isto acontece porque outros fatores, como por exemplo a distribuição de autovalores, também estão ligados a convergência dos métodos iterativos.

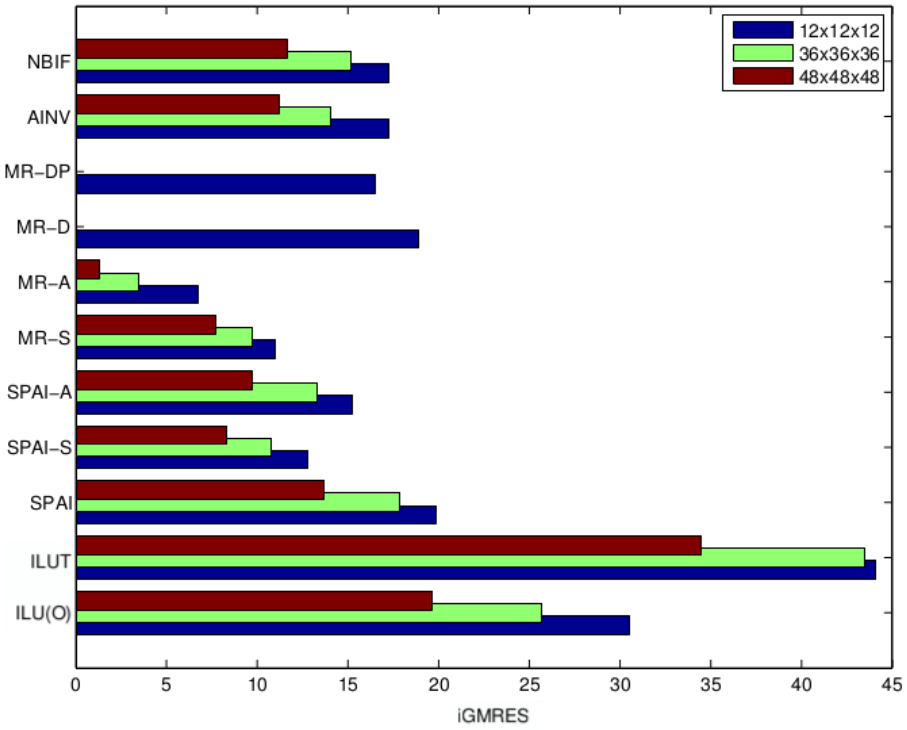

Figura 3 Melhoria no número de iterações do GMRES para o problema Heterogêneo.

Tabela 8 Resultados do problema de Convecção e Difusão com $\epsilon = 0$.

	iGMRES			**nnz(M)/nnz(A)**		
MÉTODO	12^3	36^3	48^3	12^3	36^3	48^3
GMRES	1.00	1.00	1.00	0.00	0.00	0.00
ILU(0)	10.91	13.88	13.48	1.00	1.00	1.00
ILUT	40.00	57.50	59.71	1.47	1.63	1.65
SPAI	8.57	9.58	7.08	2.76	2.84	2.84
SPAI-S	5.22	2.78	2.26	1.00	1.00	1.00
SPAI-A	6.00	5.07	3.32	1.45	1.53	1.54
MR-S	4.62	2.80	2.26	1.00	1.00	1.00
MR-A	4.29	2.21	1.78	1.69	1.61	1.60
MR-D	4.44	2.24	—	0.31	0.29	—
MR-DP	10.00	11.50	9.72	1.88	1.87	1.87
AINV	9.23	4.73	7.60	2.31	2.41	2.40
NBIF	10.00	11.13	8.20	1.48	1.54	1.55

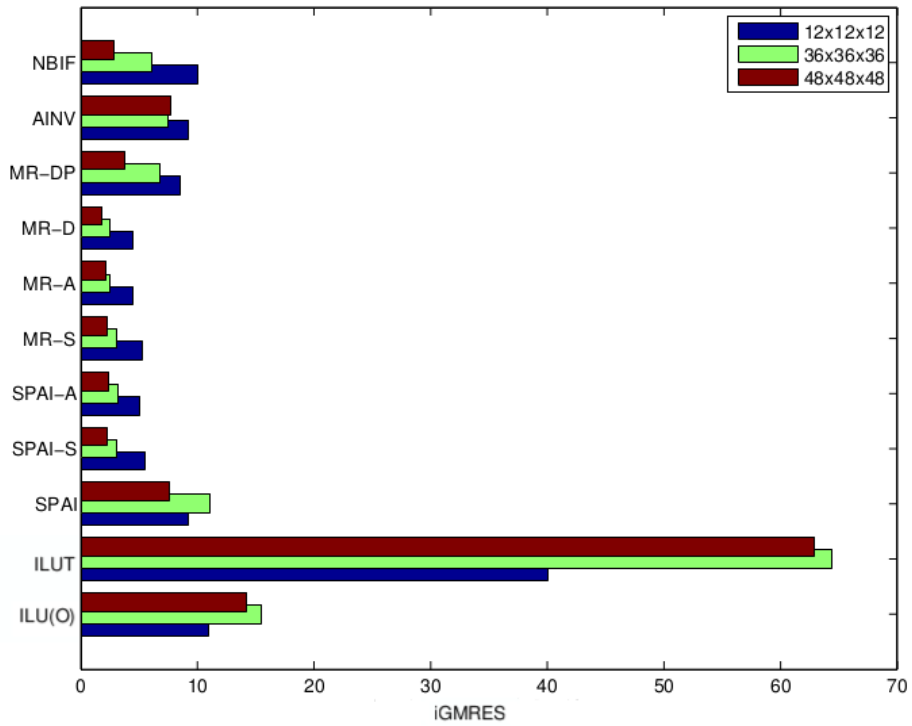

Figura 4 Melhoria no número de iterações do GMRES para o problema de Convecção e Difusão.

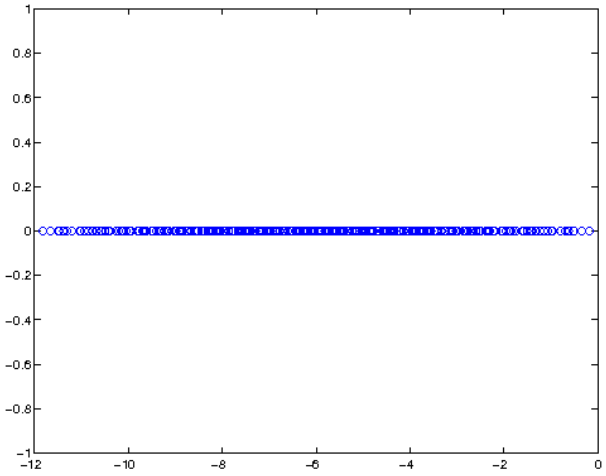

Figura 5 Distribuição de autovalores da matriz Laplaciana Homogênea com malha $12 \times 12 \times 12$.

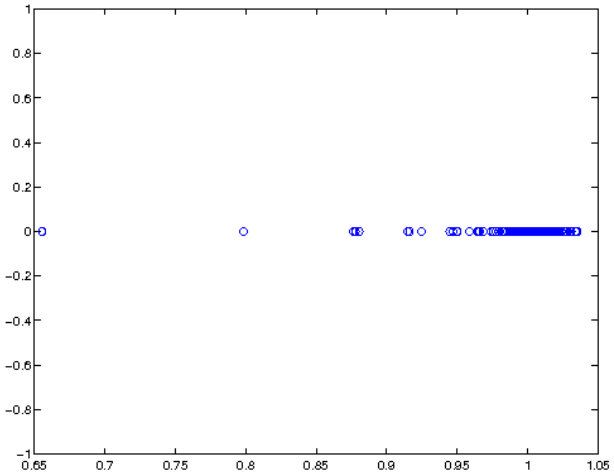

Figura 6 Distribuição de autovalores da matriz Laplaciana Homogênea com malha $12 \times 12 \times 12$ precondicionada com ILUT.

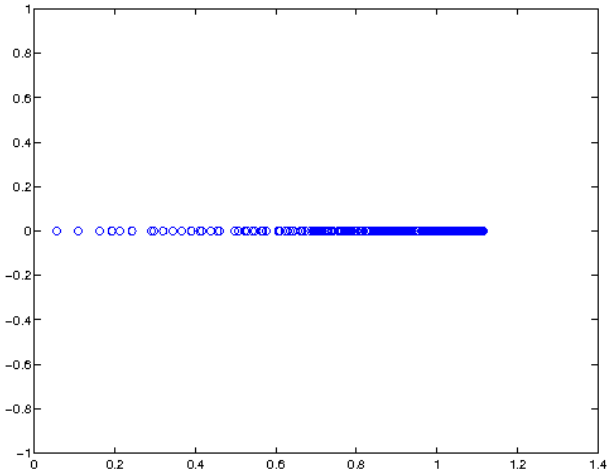

Figura 7 Distribuição de autovalores da matriz Laplaciana Homogênea com malha $12 \times 12 \times 12$ precondicionada com SPAI-S.

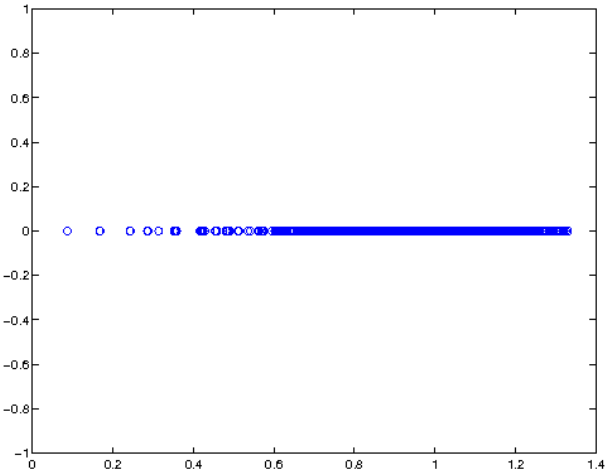

Figura 8 Distribuição de autovalores da matriz Laplaciana Homogênea com malha $12 \times 12 \times 12$ precondicionada com AINV.

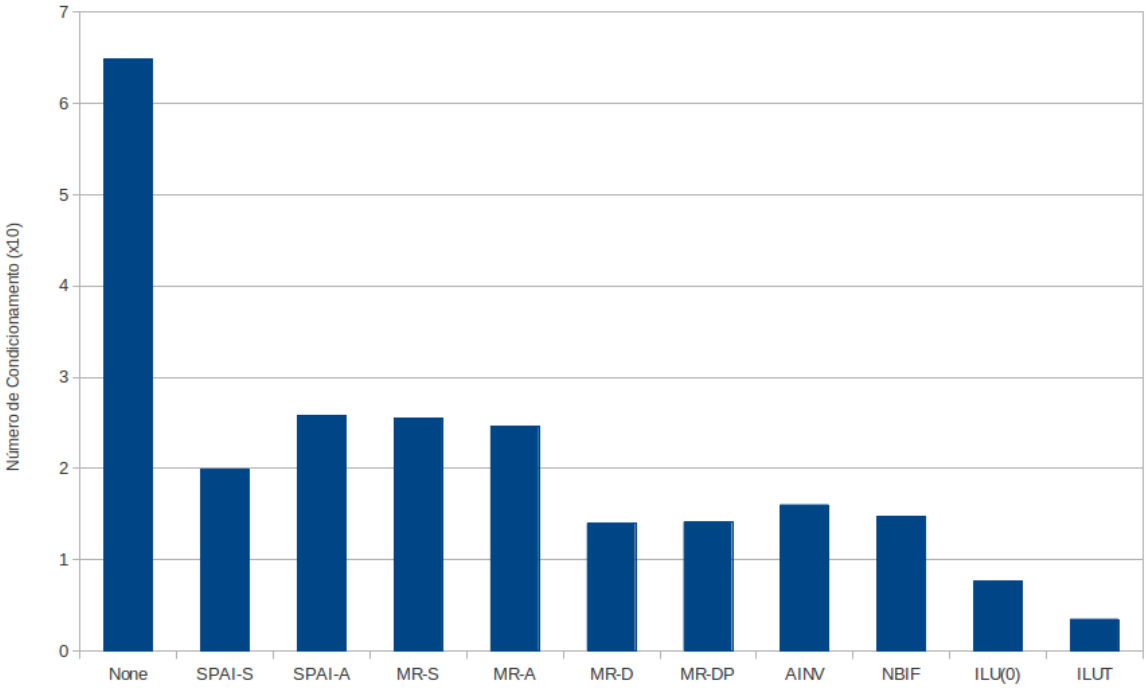

Figura 9 Melhoria no número de condicionamento da matriz Laplaciana Homogênea com malha $12 \times 12 \times 12$

CONCLUSÃO

Os precondicionadores baseados na aproximação da inversa mostraram-se aptos em tornar o GMRES mais eficiente porém, como era esperado, se mostraram menos eficientes em serial que os tradicionais **ILU(0)** e **ILUT**.

Dentre os métodos de minimização de norma de Frobenius o **SPAI** se destaca, com uma boa taxa de convergência e total paralelismo (construção e aplicação). Entretanto deve-se ressaltar que o **SPAI-S** é simples de se paralelizar mas altamente dependente de um bom padrão de esparsidade, que em geral é algo difícil de se encontrar. Já o modelo adaptativo **SPAI-A**, é capaz de capturar bem as entradas importantes da inversa mas requer acesso a matriz toda e isso torna a paralelização do algoritmo não trivial. O **MR** se mostrou muito inferior ao **SPAI** e não tão mais barato como era esperado. O **MR** também foi o método que mais falhou na resolução de alguns sistemas propostos, sendo assim o pior precondicionador de todos os testados.

Os métodos de inversa aproximada fatorada apresentaram taxas de convergência praticamente idênticas, com o **NBIF** sendo um pouco superior e mais esparso, algo justificável pela excelente estratégia de *numerical dropping* do mesmo. Por outro lado, o **NBIF** cálcula quatro fatores que torna seu algoritmo mais complexo e caro que o do **AINV**. Estes métodos se mostraram os mais robustos de inversa aproximada, e mesmo tendo construção sequencial podem ser bastante aproveitados no futuro devido às arquiteturas híbridas. Vale ressaltar que em [2], os autores afirmam que é possível se conseguir paralelismo no **AINV** através de propriedades do grafo da matriz.
Existem muitos estudos a serem feitos na inversa aproximada e novos métodos ainda podem surgir. Mesmo com os métodos de inversa aproximada não consolidados, não é nenhum absurdo projetar que devido às arquiteturas híbridas em alguns anos os métodos de aproximação da inversa serão os precondicionadores mais utilizados .

REFERÊNCIAS

[1] BENSON, M. W. *Iterative solution of large scale linear systems*. Tese (Doutorado) — Master thesis, Lakehead University, Thunder Bay, Canadá, 1973.

[2] BENZI, M.; TUMA, M. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, SIAM, v. 19, n. 3, p. 968–994, 1998.

[3] BENZI, M.; TUMA, M. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, Amsterdam [Netherlands]: North-Holland, c1985-, v. 30, n. 2, p. 305–340, 1999.

[4] BOLLHOFER, M.; SAAD, Y. On the relations between ilus and factored approximate inverses. *SIAM Journal on Matrix Anal. Appl.*, SIAM, v. 24, n. 1, p. 219–237, 2002.

[5] BRU, R. et al. Balanced incomplete factorization. *SIAM Journal on Scientific Computing*, SIAM, v. 30, n. 5, p. 2302–2318, 2008.

[6] BRU, R. et al. Improved balanced incomplete factorization. *SIAM Journal on Matrix Analysis and Applications*, SIAM, v. 31, n. 5, p. 2431–2452, 2010.

[7] CHOW, E.; SAAD, Y. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM Journal on Scientific Computing*, SIAM, v. 19, n. 3, p. 995–1023, 1998.

[8] FASANO, G.; ROMA, M. Preconditioning NewtonKrylov methods in nonconvex large scale optimization. *Computational Optimization and Applications*, Springer US, p. 1–38, 2013. ISSN 0926-6003. Disponível em: <http://dx.doi.org/10.1007/s10589-013-9563-6>.

[9] GOLUB, G. H.; LOAN, C. F. van. *Matrix Computations*. 3rd. ed. [S.l.]: Johns Hopkins University Press, 1996.

[10] GONZáLEZ, L.; SUáREZ, A. Improving approximate inverses based on Frobenius norm minimization. *Applied Mathematics and Computation*, v. 219, n. 17, p. 9363 – 9371, 2013. ISSN 0096-3003. Disponível em: <http://www.sciencedirect.com/science/article/pii/S009630031300310X>.

[11] GROTE, M. J.; HUCKLE, T. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, SIAM, v. 18, n. 3, p. 838–853, 1997.

[12] GROTE, M. *SPAI 3.0 - Parallel Preconditioning with Sparse Approximate Inverses*. maio 2005. http://www.sam.math.ethz.ch/ grote/spai.

[13] HAGER, W. Updating the inverse of a matrix. *SIAM Review*, v. 31, n. 2, p. 221–239, 1989. Disponível em: <http://epubs.siam.org/doi/abs/10.1137/1031049>.

[14] HORN, R. A.; JOHNSON, C. R. M. A. *Matrix analisys*. [S.l.]: Cambridge University Press, 1985.

[15] JIA, Z.; ZHANG, Q. An approach to making SPAI and PSAI preconditioning effective for large irregular sparse linear systems. *SIAM Journal on Scientific Computing*, v. 35, n. 4, p. A1903–A1927, 2013. Disponível em: <http://epubs.siam.org/doi/abs/10.1137/120900800>.

[16] LI, R.; SAAD, Y. Divide and conquer low-rank preconditioners for symmetric matrices. *SIAM Journal on Scientific Computing*, v. 35, n. 4, p. A2069–A2095, 2013. Disponível em: <http://epubs.siam.org/doi/abs/10.1137/120872735>.

[17] MEYER, C. D. *Matrix analisys and applied linear algebra*. [S.l.]: Siam, 2000.

[18] NIST. *Matrix Market*. [S.l.], 2013. Disponível em: <http://math.nist.gov/MatrixMarket/>.

[19] OLIVER, D.; REYNOLDS, A.; LIU, N. *Inverse Theory for Petroleum Reservoir Characterization and History Matching*. Cambridge University Press, 2008. ISBN 9780521881517. Disponível em: <http://books.google.com.br/books?id=oPuy3OfbtfIC>.

[20] PATTERSON, D.; GANNON, D.; WRINN, M. *The Berkeley Par Lab: Progress in the Parallel Computing Landsacpe*. [S.l.]: Berkeley, 2013.

[21] SAAD, Y. *Iterative Methods for Sparse Linear Systems*. 2nd. ed. [S.l.]: SIAM, 2003.

[22] SOLEYMANI, F. On a fast iterative method for approximate inverse of matrices. *Commun. Korean Math. Soc*, v. 28, n. 2, p. 407–418, 2013.

[23] STRASSBURG, J.; ALEXANDROV, V. A Monte Carlo approach to sparse approximate inverse matrix computations. *Procedia Computer Science*, v. 18, n. 0, p. 2307 – 2316, 2013. ISSN 1877-0509. 2013 International Conference on Computational Science. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050913005450>.