XXI ENMC
ENCONTRO NACIONAL DE MODELAGEM COMPUTACIONAL
IX ECTM
ENCONTRO DE CIÊNCIA E TECNOLOGIA DE MATERIAIS

08 a 11 de Outubro de 2018
Instituto Federal Fluminense
Búzios – RJ

MÉTODO GMRES REINICIALIZADO COM OS PRÉ-CONDICIONADORES JACOBI E LUMPED

Leonardo Hilário da Silva[1] – leonardohilariodasilva@ymail.com
Ricardo Silveira Sousa[2] – rsousa@infes.uff.br
[1]Universidade Federal Fluminense, INFES - Santo Antônio de Pádua, RJ, Brasil
[2]Universidade Federal Fluminense, INFES - Santo Antônio de Pádua, RJ, Brasil

Resumo. *A resolução de sistemas lineares esparsos é de grande interesse em diversas áreas da ciência. Para resolver sistemas lineares esparsos os métodos iterativos podem ser mais eficientes que os métodos diretos, como é o caso do método GMRES apresentado por Saad e Schultz (1986). Uma maneira eficiente de implementar um método de resolução de sistemas lineares de grande porte e esparsos é armazenar apenas os elementos não nulos utilizando estrutura de dados apropriada, para que possa economizar tempo e memória. Assim, neste trabalho implementamos o método GMRES reinicializado com a estrutura de dados CSR e incorporamos os pré-condicionadores Diagonal e Lumped para melhorar a qualidade da solução e tornar a convergência mais rápida. Os resultados obtidos mostraram que o método convergiu para diversos problemas e que o uso dos pré-condicionadores melhoraram muito o desempenho do método GMRES.*

Palavras Chave: *Sistema Linear, Método GMRES, Pré-condicionadores*

1. INTRODUÇÃO

A resolução de sistemas lineares esparsos é de grande interesse em diversas áreas da ciência. Logo, vários métodos foram desenvolvidos para facilitar a realização dessa tarefa nas últimas décadas. Para resolver sistemas lineares em que a matriz dos coeficientes tenha alguma propriedade, como por exemplo, simétrica definida positiva, os métodos iterativos convergem rapidamente (Saad, 2003).

Nos trabalhos de Sousa (2005) e Sousa *et. al.*, (2006) as matrizes básicas do método dual simplex especializado não tinham propriedades, como simetria ou definida positiva. Apesar disso, o método do Gradiente Bi-Conjugado resolveu com sucesso os sistemas básicos a cada iteração do método dual simplex para problemas grandes de otimização linear.

Sousa e Silva (2016) mostram que o GMRES(*m*) reinicializado (Saad e Schultz, 1986) obteve um ótimo desempenho para problemas de médio porte e bem esparsos em uma implementação que utiliza a forma tradicional de armazenamento de matrizes, chamada de forma coordenada. Este modo é muito conhecido e utilizado para armazenar matrizes de

sistemas lineares de pequeno porte. Foi verificada a convergência na maioria dos problemas testados em matrizes sem propriedade numérica e estrutural. O trabalho mostrou que o valor do parâmetro *m* deve ser em torno 1% a 10% da dimensão do problema para os testes realizados e que o método resolveu os problemas com um número muito baixo de iterações.

Silva *et. al*., (2017) demonstram que ao se alterar a estrutura de dados utilizada nas principais operações do método GMRES(*m*) para uma que se adequa a características e condições da matriz principal do problema (estrutura CSR-*Compressed Sparse Row* para matrizes esparsas), é possível obter uma economia significativa de memória, além de agilizar o processamento evitando operações desnecessárias. Comparado com o trabalho anterior (Sousa e Silva, 2016), a redução de tempo é muito grande.

Outra estratégia para alterar o GMRES(*m*) a fim de obter melhores resultados é fazendo o uso de pré-condicionadores. O uso de técnicas de pré-condicionamento pode tornar a convergência do método bem mais rápida, com um número muito baixo de iterações em função da dimensão do sistema. Pode-se considerar o pré-condicionamento como qualquer modificação realizada no sistema original de modo que o novo sistema seja equivalente ao anterior, tenha a mesma solução, porém que seja mais fácil de resolver.

A ideia do pré-condicionamento é muito simples, no entanto a escolha de um bom pré-condicionador deve ser feita de forma que o sistema pré-condicionado seja fácil de ser resolvido e o pré-condicionador seja também fácil de construir e aplicar. Ou seja, após o sistema ser pré-condicionado, o método deve convergir rapidamente e cada iteração não deve ser muito cara. Assim, resolvemos testar a eficiência de dois pré-condicionadores em alguns problemas de um repositório que serão apresentados mais adiante.

O artigo está organizado da seguinte forma: na seção 2 abordamos brevemente o método GMRES reinicializado, fazemos algumas considerações sobre pré-condicionamento e apresentamos os pré-condicionadores utilizados. A seção 3 apresenta os resultados computacionais obtidos. Na seção 4 estão às conclusões e as considerações finais e, por último temos as referências bibliográficas.

2. MÉTODO GMRES(*m*) COM PRÉ-CONDICIONAMENTO

Temos que o método original GMRES (sem reinicialização), apresentado por Saad e Schultz (1986), converge em no máximo *n* iterações. Então, no pior caso, para uma dimensão muito grande do sistema, como é o caso de aplicações práticas, isso é uma enorme desvantagem e impraticável. Pois, a cada iteração, produtos de uma matriz por um vetor precisam ser armazenados (no pior caso *n* produtos), o que pode acarretar problema de falta de memória quando é necessário um número grande de iterações do método.

Assim, Saad e Schultz (1986) propuseram também a versão chamada reinicializada conhecida como GMRES(*m*) em que *m* é a dimensão da base do subespaço de Krylov. Para este método não há garantia de convergência ou ainda, ela pode ser muito lenta.

O trabalho de Sousa e Silva (2016) revela que a implementação do método GMRES(*m*) obteve êxito na maioria dos problemas testados. Os resultados mostraram que o método convergiu em boa parte dos problemas, cujas matrizes de restrições não tinham propriedades numéricas, e realizou em média um número baixo de iterações em um tempo computacional bem razoável. Cabe destacar que os autores desenvolveram o próprio código em linguagem C, não utilizando assim nenhuma implementação disponível na literatura. Contudo, é interesse investigar e implementar uma estrutura de dados que possa reduzir o tempo computacional como foi feito por Silva *et. al*., (2017) com a estrutura de dados CSR.

No geral, o pré-condicionamento utiliza uma matriz M inversível responsável pela transformação, que é feito multiplicando-se o sistema inicial por M^{-1}. Assim, tem-se

$M^{-1}Ax = M^{-1}b$ (pela esquerda) ou $AM^{-1}x' = b$ (pela direita) onde $x' = Mx$. Pode-se utilizar também a combinação dos dois.

Foram realizadas duas alterações no algoritmo do GMRES(*m*) original apresentado por Sousa e Silva (2016). A primeira se encontra no cálculo do vetor *w* dentro do processo de Arnoldi. Assim, para o cálculo de *w* é utilizado agora o vetor z^j que precisa ser obtido através da igualdade $z^j = M^{-1}v^j$ que envolve a matriz pré-condicionadora. A segunda modificação reside na atualização da solução x^m. Em ambos os casos, o algoritmo utiliza a inversa da matriz de pré-condicionamento. É importante destacar que dependendo do sistema e do valor de *m* o método pode não convergir, pois pode-se gerar uma sequência estacionária de resíduos (Gonçalez, 2005).

Temos que algumas propriedades de uma matriz como: simétrica, definida positiva, número de condicionamento e dominância diagonal, tem grande relevância na convergência e no desempenho do método utilizado para resolver o sistema associado.

Portanto, quando se pré-condiciona a matriz original através de uma matriz de transformação ou de pré-condicionamento, estas propriedades podem surgir no sistema linear transformado conforme discutido por Carvalho *et al*., (2008), Paz (2012), Saad (2003) e Medeiros (2014). Assim, o uso de técnicas de pré-condicionamento pode tornar a convergência do método bem mais rápida, com um número muito baixo de iterações em função da dimensão do sistema linear.

No presente trabalho foram utilizados dois pré-condicionadores diferentes, considerados os mais simples de serem construídos e aplicados: Diagonal e Lumped apresentados a seguir.

2.1 – Pré-Condicionador Diagonal

O pré-condiconador Diagonal tratado por Pini, G. e Gambolati (1990) conhecido também por pré-condiconador de Jacobi, é muito simples, pois é considerado um dos mais fáceis de implementar com um custo muito baixo. A matriz pré-condicionadora *M* é uma matriz diagonal em que cada posição não nula é exatamente o elemento correspondente da diagonal da matriz original. Logo, a matriz *M* é dada por: $m_{ij} = \begin{cases} a_{ij}, & se\ i = j, \\ 0, & se\ i \neq j. \end{cases}$

Para este pré-condicionador é preciso trabalhar com a sua inversa M^{-1}. Para obter os elementos de sua inversa basta fazer $1/m_{ij}$. Porém se a diagonal apresentar algum elemento nulo será impossível obter sua inversa. Logo, para a utilização deste pré-condicionador, é feita a verificação em valor absoluto dos elementos da diagonal. Neste caso utilizamos uma tolerância de 1E-20.

Por fim, depois de todas as verificações, guardamos o valor de cada elemento da diagonal em um vetor, pois é conveniente que ela seja armazenada em um array, tornando seu processo de utilização mais simples. Para maior entendimento observe que o pré-condicionamento à direita é feita da seguinte forma:

$$\begin{aligned} AM^{-1}Mx &= b \\ x' &= Mx \end{aligned} \qquad (1)$$

Temos que $A' = AM^{-1}$, na sua forma matricial é da seguinte forma:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} . \begin{bmatrix} 1/m_{11} & 0 & \dots & 0 \\ 0 & 1/m_{22} & \dots & 0 \\ & & & \\ 0 & 0 & \dots & 1/m_{nn} \end{bmatrix} .$$

É possível notar que como a matriz M^{-1} possui elementos somente em sua diagonal, o resultado será a simples multiplicação de um elemento de A pelo elemento da diagonal de M^{-1} pela mesma linha. Assim, a fim de tornar tanto o cálculo como o armazenamento mais eficiente podemos armazenar essa matriz diagonal em um vetor, deixando a multiplicação da seguinte forma:

$$\begin{bmatrix} a_{11}\dfrac{1}{m_{11}} & a_{12}\dfrac{1}{m_{22}} & \dots & a_{1n}\dfrac{1}{m_{nn}} \\ a_{21}\dfrac{1}{m_{11}} & a_{22}\dfrac{1}{m_{22}} & \dots & a_{2n}\dfrac{1}{m_{nn}} \\ & & & \\ a_{n1}\dfrac{1}{m_{11}} & a_{n2}\dfrac{1}{m_{22}} & \dots & a_{nn}\dfrac{1}{m_{nn}} \end{bmatrix} .$$

Logo, resolvido o sistema $A'x' = b$, encontra-se o vetor x usando (1).

2.2 – Pré-Condicionador Lumped

O pré-condicionador Lumped é uma variante do pré-condicionador Diagonal também de fácil construção e de custo baixo descrito em Anjos *et. al*., (2006). De acordo com os autores, os elementos da diagonal são a soma de todas as entradas da coluna correspondente dado por:

$$m_{ij} = \begin{cases} \sum_{j=1}^{n} a_{ij}, \ se\ i = j, \\ 0\ , \ se\ i \neq j. \end{cases}$$

A construção deste pré-condicionador foi de maneira muito semelhantemente ao de Jacobi. Os elementos da diagonal da matriz M, formados pela soma de todas as entradas da coluna correspondente, são invertidos $\dfrac{1}{\sum_{j=1}^{n} a_{ij}}$, armazenados em um vetor e utilizados. Observe que neste caso, somente a soma deve ser não nula, diferentemente do pré-condicionador Diagonal, algo que é mais difícil de ocorrer.

Da mesma forma que o pré-condicionador Diagonal, ele também foi implementado utilizando somente um vetor para armazenar seus valores, cujos resultados encontram-se na próxima seção.

3. EXPERIMENTOS COMPUTACIONAIS

O GMRES(*m*) Pré-Condicionado foi implementado em linguagem C e compilado no GCC com as estruturas de dados CSR. Os dados foram alocados dinamicamente em função da dimensão do problema (*n*) e do parâmetro *m*. Os dados foram fornecidos por arquivo. Os resultados foram obtidos em notebook com processador i7-650OU 2.5 GHz com 8 GB de memória RAM.

A Tabela 1 apresenta as características dos problemas que resolvemos do repositório *MatrixMarket* disponível no endereço (*http://math.nist.gov/MatrixMarket/data/*), que consistem de problemas reais e difíceis de serem resolvidos.

Tabela 1 – Propriedades dos Problemas

Problemas		**Matriz Simétrica**	**Dimensão da matriz**	**Elementos não-nulos**	**Esparsidade**
Pequenos	***b1_ss***	não	7	15	30,612%
	fidap005	não	27	279	38,272%
	fidapm05	não	42	520	29,478%
	e05r0000	não	236	5856	10,514%
Médios	***sherman1***	sim	1000	3750	0,375%
	sherman2	não	1080	23094	1,980%
	sherman4	não	1104	3786	0,311%
	add20	não	2395	17319	0,302%
	cavity10	não	2597	76367	1,132%
Grandes	***sherman5***	não	3312	20793	0,190%
	e20r2000	não	4241	131556	0,731%
	c-26	sim	4307	34537	0,186%
	add32	não	4960	23884	0,097%
	sherman3	não	5005	20033	0,080%
	memplus	não	17758	126150	0,040%

Note que temos apenas dois problemas em que a matriz dos coeficientes é simétrica e que quanto maior o problema maior a esparsidade.

Para estes problemas utilizamos o vetor *b* como sendo aquele fornecido pela própria fonte, pois para outros sistemas do repositório o vetor independente não está presente. Alguns pesquisadores sugerem utilizar o vetor unitário quando este não está disponível, mas é um caso muito particular e desta forma acreditamos que pode ter algum tipo de influência no comportamento do método GMRES(*m*).

Como houve várias situações nos resultados obtidos em termos de não convergência, insuficiência de memória e convergência com diferentes valores para o resíduo, apresentamos a seguinte legenda para as tabelas que virão a seguir: E: memória insuficiente; NC+: GMRES(*m*) não convergiu. Norma do resíduo muito alta; NC-: GMRES(*m*) não convergiu. Norma do resíduo na casa das unidades; C1: GMRES(*m*) convergiu com norma do resíduo de 1E-01 a 1E-04; C2: GMRES(*m*) convergiu com norma do resíduo de 1E-05 a 1E-08; C3: GMRES(*m*) convergiu com norma do resíduo de 1E-09 a 1E-12; C4: GMRES(*m*) convergiu com norma do resíduo abaixo de 1E-13; I: Total de iterações; R: Norma do resíduo.

Como já foi observado por Sousa e Silva (2016) e Silva *et al*., (2017) o parâmetro *m* tem enorme influência na convergência do método, no número de iterações e principalmente no tempo computacional. Assim, iremos apresentar somente os testes que envolvem os valores de *m* (porcentagem em função da dimensão do sistema) que proporcionam, de modo geral, melhor desempenho, em termos de iterações, para o método GMERS(*m*).

Os resultados estão divididos em três partes. Na primeira temos os resultados para o GMRES(*m*) sem pré-condicionamento, na segunda parte apresentamos os experimentos relacionados ao pré-condicionador Diagonal, e por último temos os resultados com o pré-condicionador Lumped.

3.1 – GMRES(*m*) sem Pré-condicionador

A Tabela a seguir apresenta o número de iterações requeridas pelo GMRES(*m*) sem pré-condicionamento bem como a norma do resíduo para os problemas do repositório que trabalhamos.

Tabela 2 – Iterações e Norma do Resíduo para o GMRES(*m*) Sem Pré-condicionador

Problemas		**Dimensão do parâmetro *m* (% de *n*)**							
		1		**5**		**10**		**15**	
		I	**R**	**I**	**R**	**I**	**R**	**I**	**R**
Pequenos	***b1_ss***	NC^+		NC^+		NC^+		NC^-	
	fidap005	27	C1	27	C1	27	C1	27	C1
	fidapm05	42	C1	NC^+		NC^+		NC^+	
	e05r0000	NC^+		E		NC^+		146	C3
Médios	***sherman1***	1000	C2	49	C3	12	C3	6	C3
	sherman2	NC^+		NC^+		E		E	
	sherman4	154	C3	6	C3	2	C3	2	C3
	add20	1	C3	1	C4	1	C4	1	C4
	cavity10	1	C3	1	C3	1	C4	1	C4
Grandes	***sherman5***	1798	C3	79	C3	25	C3	13	C3
	e20r2000	E		E		E		E	
	c-26	NC^+		E		E		E	
	add32	1	C4	1	C4	1	C4	1	C4
	sherman3	2783	C3	30	C3	4	C3	3	C3
	memplus	1	C4	1	C4	E		E	

Note que não houve convergência para alguns problemas e ocorreu estouro de memória para o problema *e20r2000* para todos os valores de *m* . Nos problemas médios e grandes o método GMRES(*m*) realizou uma única iteração em varias situações, ou seja, um excelente desempenho. Observe também uma certa influência do parâmetro *m* em relação ao número de iterações e convergência.

3.2 – GMRES(*m*) com o Pré-condicionador Diagonal

A Tabela 3 exibe os resultados com o método GMRES(*m*) com a estrutura de dados CSR usando agora o pré-condicionador Diagonal para os mesmos problemas envolvendo iterações e resíduo.

Tabela 3 – Iterações e Norma do Resíduo para o GMRES(*m*) com o Pré-condicionador Diagonal

Problemas		**Dimensão do parâmetro *m* (% de *n*)**							
		1		**5**		**10**		**15**	
		I	**R**	**I**	**R**	**I**	**R**	**I**	**R**
Pequenos	***b1_ss***	Diagonal Nula							
	fidap005	27	C1	27	C1	27	C1	27	C1
	fidapm05	Diagonal Nula							
	e05r0000	Diagonal Nula							

Médios	***sherman1***	215	C3	23	C3	7	C3	3	C3
	sherman2	NC^{+}		E		E		E	
	sherman4	52	C3	5	C3	1	C3	38	C3
	add20	1	C4	1	C3	E		E	
	cavity10	Diagonal Nula							
Grandes	***sherman5***	25	C3	2	C3	E		E	
	e20r2000	Diagonal Nula							
	c-26	1	C4	1	C4	1	C4	1	C4
	add32	1	C4	1	C4	E		E	
	sherman3	56	C3	6	C3	4	C3	4	C3
	memplus	1	C3	E		E		E	

Podemos perceber que não foi possível utilizar o pré-condicionador em alguns casos, pois há a presença de pelo menos um elemento nulo na diagonal. Notamos que para os problemas médios houve uma redução muita significativa na quantidade de iterações, quando utiliza-se este pré-condicionador, principalmente quando *m* vale 1% ou 5%. Nos problemas maiores *sherman5* e *sherman3* temos uma enorme redução na quantidade de iterações, saindo de 1798 para 25 e 2783 para 56 respectivamente. Isto é, o uso desta técnica reduz drasticamente o esforço do método GMRES(*m*) para alguns problemas.

A seguir, na Tabela 4 vamos comparar o tempo computacional (em segundos) do GMRES(*m*) com e sem o pré-condicionador Diagonal. Considere que: T.S. representa o tempo de resolução para o GMRES(*m*) sem Pré-condicionamento, T. D. representa o tempo de resolução para o GMRES(*m*) com o Pré-condicionador Diagonal e DN significa que algum elemento da diagonal é nulo.

Tabela 4 – Tempo GMRES(*m*): Sem Pré-condicionador × Com Pré-condicionador Diagonal

Problemas		**Dimensão do parâmetro *m* (% de *n*)**							
		1		**5**		**10**		**15**	
		T. S.	**T. D.**	**T. S.**	**T. D.**	**T. S.**	**T. D.**	**T. S.**	**T. D.**
Pequenos	***b1_ss***	-	DN	-	DN	-	DN	-	DN
	fidap005	0,000	0,000	0,015	0,000	0,000	0,000	0,000	0,000
	fidapm05	0,000	DN	-	DN	-	DN	-	DN
	e05r0000	-	DN	-	DN	-	DN	0,766	DN
Médios	***sherman1***	0,764	0,166	1,219	0,516	2,672	1,501	5,501	2,751
	sherman2	-	-	-	-	-	-	-	-
	sherman4	0,172	0,062	0,200	0,171	0,625	0,312	2,719	51,643
	add20	0,015	0,016	0,500	0,500	6,250	-	30,987	-
	cavity10	0,031	DN	0,735	DN	9,422	DN	43,465	DN
Grandes	***sherman5***	47,848	0,641	136,532	3,454	589,479	-	2057,961	-
	e20r2000	-	DN	-	DN	-	DN	-	DN
	c-26	-	0,000	-	0,000	-	0,000	-	0,015
	add32	0,079	0,093	8,907	8,799	159,484	-	766,568	-
	sherman3	236,839	4,766	277,213	55,504	666,279	668,391	2419,022	3239,812
	memplus	6,630	6,251	1790,633	-	-	-	-	-

Os resultados mostram que para os problemas maiores os tempos de resolução aumentam bem, principalmente quando o parâmetro *m* assume os maiores valores.

Para os problemas médios notamos a queda das iterações e dos tempos, como por exemplo no caso do *sherman*, pois fez menos iterações. Observe que no caso do *add20* o pré-condicionador não influenciou no tempo de resolução, já que também o resolveu em uma única iteração. Para o problema *sherman4* com valor de *m* igual a 15% de *n*, temos que o tempo por iteração do GMRES(*m*) com pré-condicionador Diagonal é aproximadamente de

1,35 segundos, equivalente ao GMRES(*m*) sem pré-condicionador. Ou seja, esta técnica não tem um custo computacional relevante por iteração.

Com relação aos problemas maiores observamos uma queda brusca nos tempos de resolução, isto é, já que o número de iterações foi bem reduzido, um excelente resultado, principalmente por este ser o grupo das maiores matrizes, em especial para os problemas *sherman5* e *sherman3*. Outro resultado promissor foi o caso do *c-26*, em que antes, sem pré-condicionador o método GMRES(*m*) não conseguiu convergir em nenhuma situação, porém com o pré-condicionador Diagonal, obteve-se o resultado com um tempo de resolução praticamente nulo.

3.3 – GMRES(*m*) com Pré-condicionador Lumped

A Tabela 5 exibem o número de iterações e resíduo com o método GMRES(*m*) usando o pré-condicionador Lumped.

Tabela 5 – Iterações e Norma do Resíduo para o GMRES(*m*) com o Pré-condicionador Lumped

Problemas		**Dimensão do parâmetro *m* (% de *n*)**							
		1		**5**		**10**		**15**	
		I	**R**	**I**	**R**	**I**	**R**	**I**	**R**
Pequenos	***b1_ss***	NC^+		NC^+		NC^+		NC^+	
	fidap005	NC^+		NC^-		27	C1	27	C1
	fidapm05	42	C1	NC^+		NC^-		42	C1
	e05r0000	NC^+		NC^-		NC^-		NC^-	
Médios	***sherman1***	E		E		E		E	
	sherman2	NC^+		NC^+		E		E	
	sherman4	208	C3	24	C3	10	C3	3	C3
	add20	2	C3	1	C3	1	C3	1	C3
	cavity10	E		E		E		E	
Grandes	***sherman5***	E		361	C3	17	C3	10	C3
	e20r2000	E		E		E		E	
	c-26	NC^+		E		E		E	
	add32	1	C4	1	C4	1	C4	1	C3
	sherman3	128	C3	11	C3	3	C3	2	C3
	memplus	E		E		E		E	

Observamos que para os problemas pequenos este pré-condicionador como o Diagonal não se mostrou muito eficiente, pois o GMRES(*m*) convergiu para apenas dois problema.

Com relação aos problemas médios, este pré-condicionador foi inferior ao pré-condicionador Diagonal. E ainda em algumas situações ele acabou por aumentar o número de iterações, como o caso do problema *sherman4* (52 iterações com o Diagonal e agora 208 iterações), mas ainda assim muito inferior as 154 iterações sem o uso de pré-concionador

Cabe relatar também que o problema *fidapm05* foi resolvido com este pré-condicionador, o que não ocorreu com o anterior, e para os problemas *sherman4* e *sherman3* com *m* = 15% ele também foi melhor. Notamos também para os problemas maiores que este pré-condicionador não teve um desempenho muito bom, havendo-se muitos casos estouro de memória.

No geral, o desempenho do pré-condicionador Lumped foi inferior ao pré-condicionador anterior, mas em várias situações foi muito melhor sem o uso de pré-condicionamento.

A seguir, vamos apresentar na Tabela 6 a comparação de tempo computacional (em segundos) para o GMRES(*m*) envolvendo os dois pré-condicionadores utilizados. Considere

que T. L. representa o tempo de resolução para o GMRES(*m*) com o Pré-condicionador Lumped.

Tabela 6 – Tempo GMRES(*m*): Com Pré-condicionador Diagonal × Com Pré-condicionador Lumped

Problemas		**Dimensão do parâmetro *m* (% de *n*)**							
		1		**5**		**10**		**15**	
		T. D.	**T. L.**	**T. D.**	**T. L.**	**T. D.**	**T. L.**	**T. D.**	**T. L.**
Pequenos	***b1_ss***	DN	-	DN	-	DN	-	DN	-
	fidap005	0,000	-	0,000	-	0,000	0,000	0,000	0,000
	fidapm05	DN	0,000	DN	-	DN	-	DN	0,004
	e05r0000	DN	-	DN	-	DN	-	DN	-
Médios	***sherman1***	0,166	-	0,516	-	1,501	-	2,751	-
	sherman2	-	-	-	-	-	-	-	-
	sherman4	0,062	0,219	0,171	0,797	0,312	3,204	51,643	4,047
	add20	0,016	0,032	0,500	0,516	-	6,188	-	30,987
	cavity10	DN	-	DN	-	DN	-	DN	-
Grandes	***sherman5***	0,641	-	3,454	642,597	-	414,336	-	1648,060
	e20r2000	DN	-	DN	-	DN	-	DN	-
	c-26	0,000	-	0,000	-	0,000	-	0,015	-
	add32	0,093	0,078	8,799	8,814	-	158,605	-	765,839
	sherman3	4,766	10,798	55,504	101,319	668,391	501,912	3239,812	1596,633
	memplus	6,251	-	-	-	-	-	-	-

Notamos que para os problemas pequenos os tempos continuaram os mesmos, sendo eles praticamente nulos.

Nos problemas médios ocorreu o esperado, visualizando os resultados das iterações e da qualidade de solução: os tempos ficaram próximos do Pré-condicionador Diagonal ou até mesmo pioraram, com exceção do *sherman4* e *sherman3* com *m* igual a 15% de *n*. Para estes problemas o Pré-condicionador Lumped reduziu de 38 para 3 iterações e de 4 iterações para apenas duas, para os problemas *sherman4* e do *sherman3* respectivamente. Nestes dois casos o tempo por iteração foi praticamente o mesmo, na ordem 1,35 segundos para o primeiro problema (*sherman4*) e para o segundo (*sherman3*) foi de 798,3 segundos. Ou seja, para estas duas instancias, o desempenho do Pré-condicionador Lumped foi muito superior ao Diagonal.

Com relação aos problemas grandes é possível ver que no caso do *add32*, o método com este pré-condicionador obteve praticamente o mesmo tempo do GMRES(*m*) sem pré-condicionamento.

Portanto, no geral, o uso de ambos pré-condicionadores no método GMRES(*m*) reduz de maneira muito significante o número de iterações e consequentemente o tempo de resolução do problema.

4. CONCLUSÕES E CONSIDERAÇÕES FINAIS

Neste trabalho tratamos o método GMRES reinicializado utilizando dois pré-condicionadores simples para resolver sistemas lineares esparsos com o uso de uma estrutura de dados muito eficiente para reduzir o tempo computacional.

Para incorporar a implementação dos Pré-condicionadores ao GMRES(*m*) foram feitas duas modificações no algoritmo em que se utiliza a inversa da matriz pré-condicionadora. Assim, trabalhamos com pré-condicionadores fáceis de construir e utilizar, pois o tempo computacional por iteração poderia aumentar muito sem ter a possibilidade real de reduzir o número de iterações caso utilizássemos outro pré-condicionador.

Foram pesquisados e incorporados dois pré-condicionadores diferentes a fim de observar sua influência na qualidade da solução obtida, na quantidade de iterações requisitadas pelo método e no tempo computacional. Através dos resultados obtidos podemos destacar que não é possível afirmar que um determinado pré-condicionador é o melhor para todos os tipos de problemas. Mas, no geral, os experimentos computacionais revelam que o método GMRES(*m*) Pré-condicionado obteve sucesso nos problemas resolvidos, oriundos de aplicações práticas. O método convergiu na maioria dos problemas testados realizando um baixíssimo número de iterações com uma ótima qualidade das soluções e com tempo computacional por iteração muito bom.

Portanto, os resultados mostram um excelente desempenho do método GMRES(*m*) com a estrutura de Dados CSR com o uso dos pré-condicionadores Diagonal e Lumped. Pois, tivemos para vários problemas a convergência do método, a melhora na qualidade da solução com base na norma do resíduo e principalmente na redução do número de iterações sem aumentar o tempo computacional.

O trabalho anterior (Silva *et al*., 2017) demonstrou a necessidade de utilizar uma estrutura de dados eficiente para implementar o método iterativo com a vistas a redução do tempo computacional. Assim, a combinação da estrutura de dados CSR com o uso dos Pré-condicionadores se mostrou muito eficiente para os problemas que foram solucionados. Como trabalho futuro, investigaremos outros pré-condicionadores mais elaborados, desde que não sejam muito caros tanto para construção quanto sua aplicação.

Agradecimentos

Este trabalho teve o apoio da FAPERJ através do Edital de Iniciação Científica 2016.

REFERENCES

Anjos, G. R., Mangiavacchi, N., Pontes, J., Soares, C. B. P. (2006) Simulação Numérica das Equações de Saint-Venant Utilizando o Método dos Elementos Finitos. 16o POSMEC. FEMEC/UFU, Uberlândia-MG.

Carvalho, L. M., Fortes, W. R., Mangiavacchi, N. (2008) Precondicionadores e solucionadores para problemas de reservatório. In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, 31., 2008, Belém. Anais... . Belém: Sbmac. p. 363 - 369.

Gonçalez, T. T. (2005) "*Algoritmos Adaptativos para o Método GMRES(m)*" Dissertação de Mestrado. Instituto de Matemática. Universidade Federal do Rio Grande do Sul.

Medeiros, E. N. (2014) NI-GMRES Precondicionado. 61 f. Dissertação (Mestrado) – Curso de Matemática Aplicada e Estatística, Universidade Federal do Rio Grande do Norte, Natal.

Paz, A. R.A (2012) "*Aplicação do Método GMRES na Solução de Problemas de Estabilidade em Sistemas de Energia Elétrica*" Tese de Doutorado. Departamento de Engenharia Elétrica. PUC-Rio.

Pessanha, J. E. O., PAZ, A. A. e PRADA, R. (2012) *"Aplicacao do Método GMRES em Estudos de Estabilidade de Sistemas de Energia Elétrica"*. Revista Controle e Automação. Vol 23. No 3.

Pini, G., Gambolati, G. (1990) Is a simple diagonal scaling the best preconditioner for conjugate gradients on supercomputers? Advances in Water Resources, 13(3):147–153.

Saad, Y. (2003) "*Iterative Methods for Sparse Linear Systems*" Second Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Saad, Y. e Schultz, M. (1986) GMRES: A Generalized Minimal Residual Algorithm for Solving NonSymmetric Linear Systems, *SIAM J. Sci. Stat. Comput*., 7, 3, 856-869.

Silva, L. H., Sousa, R. S., Junior, C. A. C., Corrêa, R. A. P (2017) Método GMRES Reinicializado com a Estrutura de Dados CSR. In: Encontro Nacional De Modelagem Computacional. Friburgo: ENMC.

Sousa, R. S. (2005) "*Métodos Tipo Dual Simplex para Problemas de Otimização Linear Canalizados*" Tese de Doutorado, ICMC-USP.

Sousa, R. S., Oishi, C. M. e Arenales, M. N. (2006) Método Dual Simplex com Busca Unidimensional Utilizando o Gradiente Bi-Conjugado. Anais do XXXVIII SBPO *Simpósio Brasileiro de Pesquisa Operacional*.

Sousa, R. S.; Silva, L. H. (2016) Método GMRES para Sistemas Lineares Esparsos. In: Simpósio de Engenharia de Produção. SIMPEP, Bauru.

GMRES METHOD REINITIALIZED WITH JACOBI AND LUMPED PRECONDITIONERS

Abstract. *The resolution of sparse linear systems is of great interest in several areas of science. In order to solve sparse linear systems, iterative methods can be more efficient than direct methods, such as the GMRES method presented by Saad and Schultz (1986). An efficient way to implement a large and sparse linear systems resolution method is to store only the nonzero elements using the appropriate data structure so that it can save time and memory. Thus, in this paper we implemented the GMRES method reinitialized with the CSR data structure and incorporated the Diagonal and Lumped preconditioners to improve the quality of the solution found and make the convergence faster. The results showed that the method converged to several problems and that the use of the preconditioners greatly improved the performance of the GMRES method.*

Keywords: *Linear System, GMRES Method, Preconditioner*