ISSN: 1415-7314
ISSN online: 2317-6717

ENGEVISTA
Página da revista: http://www.uff.br/engevista/seer/

Avaliação do formato de armazenamento *Compressed Sparse Row* para resolução de sistemas de equações lineares esparsos

Evaluation of the compressed sparse row storage format for solving sparse linear systems

Gylles Ricardo Ströher[1]

Thays Rolim Mendes[2]

Neyva Maria Lopes Romeiro[3]

Resumo: Os esquemas de compressão de matrizes possibilitam armazenar matrizes esparsas em vetores de forma que apenas os elementos não nulos das matrizes são armazenados, provendo assim uma redução significativa do consumo de memória computacional para o armazenamento de matrizes esparsas. Dentre os esquemas existentes, o implementado no desenvolvimento do presente trabalho foi o *Compressed Sparse Row* (CSR), o qual armazena apenas os elementos não nulos da matriz em três vetores. O esquema CSR foi implementado em associação com três métodos iterativos de resolução de sistemas lineares, Jacob, Gauss-Seidel e Gradiente Conjugado. Os resultados obtidos sinalizam para qual ordem e grau de esparsidade mínimos o esquema CSR se torna vantajoso, em relação à redução do consumo de memória computacional e os resultados também indicam que como as operações com os elementos nulos são suprimidas, o tempo de processamento para a resolução de sistemas lineares esparsos pode ser significativamente reduzido com o esquema de compressão explorado.

Palavras-chave: *Compressed Sparse Row*; Métodos Numéricos; Sistemas Lineares Esparsos; Compressão de Matrizes

[1] Universidade Tecnológica Federal do Paraná
[2] Universidade Tecnológica Federal do Paraná
[3] Universidade Estadual de Londrina

ISSN: 1415-7314
ISSN online: 2317-6717

Abstract: The sparse matrix compression formats enable to store sparse arrays in vectors which only non-zero elements of the matrix are stored, providing a significant reduction in memory consumption for storing sparse matrices. Among the existing schemes, the implemented in the development of this work was the Compressed Sparse Row (CSR), which stores only the nonzero elements of the matrix defining three vectors. CSR scheme was implemented in association with three iterative methods to solve linear systems, Jacob, Gauss-Seidel, and Conjugate Gradient. The obtained results indicate to what order and minimum sparsity degree the CSR scheme becomes advantageous in relation to the reduction of computational memory consumption and the results also indicated that as the operations with zero elements are suppressed, the processing time for solving sparse linear systems can be significantly reduced with the compression scheme used.

Keywords: Compressed Sparse Row; Numerical Methods; Sparse Linear Systems; Matrix Compression

---

ISSN: 1415-7314
ISSN online: 2317-6717

1. Introdução

Matrizes esparsas recorrentemente surgem de forma direta no processo de solução de muitos problemas práticos de engenharia, por exemplo, em modelo de balanço de massa e de energia, sistemas de distribuição de eletricidade, água e gases. Matrizes esparsas também surgem de forma indireta, como nos casos em que equações diferenciais ordinárias ou parciais são resolvidas numericamente aplicando métodos como de diferenças finitas, elementos finitos ou volumes finitos. Após a discretização das equações diferenciais, um sistema linear do tipo, A*x=b, é geralmente gerado e dependendo das dimensões e características do problema a matriz A é esparsa e de grande dimensão.

A fim de exemplificação, considere o caso da resolução da equação de Laplace bidimensional ($\partial^2\theta/\partial x^2+\partial^2\theta/\partial y^2$) em coordenadas cartesianas com condições de contorno Dirichlet, este caso pode representar vários problemas práticos de engenharia, como a transferência de calor em um meio sólido ou transferência de massa de um soluto em meio estagnado. Utilizando-se a técnica de volumes finitos (Versteeg e Malalasekera, 2007) com malhas estruturadas de 10x10, 20x20 e 40x40 volumes, os índices de esparsidade das matrizes obtidas são de 0,95, 0,9875 e 0,9968, respectivamente. Isto é, apenas 5%, 1,25% e 0,312% são elementos não nulos. Em particular para a malha mais refinada, a matriz possui 256.10^4 elementos sendo apenas 8.000 diferentes de zero. Em Duff (1977) é apresentada uma longa lista de aplicações práticas, com referências, em que sistemas lineares esparsos surgem no processo de solução de problemas. Ainda como afirma Rizwan (2007), estima-se que em 75% dos problemas científicos a solução de um sistema linear de equações aparece em algum estágio da solução.

Historicamente, o interesse em matrizes esparsas se intensificou na década de 50, quando pesquisadores de sistemas de potência elétrico começaram a resolver problemas reais usando computadores. Foi verificado que quando problemas reais eram modelados por meio de sistemas de equações lineares, as matrizes resultantes eram esparsas com uma estrutura de elementos não nulos altamente padronizada. O tema matrizes esparsas tornou-se então cada vez mais importante. Na década de 60 a primeira conferência sobre este tema foi realizada pela IBM Research Center em 1968. Desde então, surgiram diversas técnicas para o armazenamento de matrizes associados à resolução de sistemas de equações lineares, Connor (1984).

Segundo Saad (2003), ainda na década de 50 e 60 os métodos diretos para resolução de sistemas lineares eram preferidos em aplicações reais por causa de sua robustez. No entanto, foram desenvolvidos uma série de métodos iterativos e a necessidade de se resolver grandes sistemas lineares provocou uma visível e rápida mudança na direção dos métodos iterativos em muitas aplicações. Essa tendência pode ser observada até a década de 1960 e 1970, quando dois

ISSN: 1415-7314
ISSN online: 2317-6717

acontecimentos revolucionaram os métodos de solução para grandes sistemas lineares. Primeiro foi à intenção de se tirar proveito da esparsidade dos sistemas para projetar métodos diretos de resolução de sistemas lineares esparsos, que pode ser bem mais eficiente que os métodos diretos convencionais, o segundo foi o surgimento do método do Gradiente Conjugado.

Atualmente, dentre os métodos iterativos mais empregados nos sistemas de engenharia destacam-se o método de Jacob, Gauss-Seidel, Successive Over-Relaxation SOR e Gradiente Conjugado e suas derivações. A fim de tirar proveito da esparsidade de sistemas lineares e também com o intuito de economizar memória computacional, surgiram diversos formatos ou também denominados esquemas de compressão de matrizes. De acordo com Saad (1994) há mais de treze formatos de armazenamento diferentes para matrizes. Entre estes podem-se destacar: o CSR *Compressed Sparse Row* (CSR), *Compressed Sparse Colunm* (CSC), *Compressed Diagonal Storage* (CDS) *Skyline*, Saad (2003), Solin (2006), *Compressed Sparse Vector* (CSV), Farzaneh et al. (2009).

De acordo com o tipo da matriz alguns formatos podem ser mais adequados do que outros, por exemplo, o CDS, segundo Coelho (2015), é mais indicado para matrizes do tipo banda, em que somente a diagonal principal e um pequeno número de diagonais acima e abaixo da diagonal principal são não nulas. Já o formato Skyline, de acordo com Bathe e Wilson (1976) é interessante para matrizes simétricas, decorrente da análise de elementos finitos. Independentemente do esquema de compressão, fundamentalmente estes visam aproveitar o grande número de elementos nulos da matriz e armazenar somente os elementos não nulos, diminuindo de forma significativa o número de entradas do algoritmo de solução do sistema de equações lineares e suprimindo as operações aritméticas desnecessárias entre elementos nulos. Contudo, existem pouquíssimos trabalhos disponíveis relatando um estudo sistemático do uso de formatos de compressão de matrizes em função do tamanho do sistema linear, índice de esparsidade e métodos iterativos. Não há referências indicando para quais valores de esparsidade e tamanhos de matrizes os esquemas de compressão tornam-se vantajosos, uma vez que a compressão demanda tempo computacional para ser executada, também é importante ter indicativos de quanto seria a redução no tempo de processamento de solução de sistemas lineares por método iterativos que operem no formato de armazenamento comprimido.

Neste contexto, o presente trabalho apresenta um estudo do formato CSR aplicado aos métodos de resolução de sistemas lineares Jacob, Gauss-Seidel e Gradiente Conjugado. Foram comparados o tempo de processamento dos métodos iterativos com e sem o formato CSR para atingir o critério de parada especificado para diversas esparsidades e ordens de matrizes. Adicionalmente, foi também avaliada a redução de memória ou não para o armazenamento das matrizes com o armazenamento no formato CSR.

ISSN: 1415-7314
ISSN online: 2317-6717

2. Métodos

2.1 Formato de armazenamento CSR

O formato CSR, também conhecido como CRS, foi originalmente sugerido por Rose (1972) e posteriormente por Brameller (1976). Este formato é o mais popular esquema de armazenamento para grandes matrizes esparsas. O armazenamento neste formato requer a construção de três vetores AA, JR e JC de comprimento Nz, Nz e n+1 respectivamente, em que n é o número de linhas e Nz é o tal de elementos não nulos da matriz A. O vetor AA $\in$ R, contém os valores nos elementos não nulos de A armazenados linha por linha, JR $\in$ N contém os índices da coluna a qual corresponde o elemento não nulo e JC $\in$ N contém os índices da posição do primeiro elemento não nulo de cada coluna. Para exemplificação, considere a matriz quadrada 5 x 5 mostrada na Fig. 1.

$$A = \begin{bmatrix} 1 & 0 & 2 & 0 & 0 \\ 0 & 3 & 0 & 4 & 0 \\ 5 & 0 & 6 & 0 & 7 \\ 0 & 0 & 8 & 9 & 0 \\ 10 & 0 & 0 & 0 & 11 \end{bmatrix}$$

Figura 1: Matriz exemplo em que são armazenados os elementos nulos e não nulos.

O formato CSR armazenará a matriz mostrada na Fig. 1 na seguinte forma:

AA = [1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11.]
JR = [1 3 2 4 1 3 5 3 4 1 5]
JC = [1 3 5 8 10]

O formato CSR foi escolhido no presente trabalho, pois, em princípio, pode ser aplicado a qualquer tipo de matriz esparsa, não sendo limitado a matrizes do tipo quadradas, ou simétricas, ou do tipo banda. Como comenta George (2007) o formato CSR não faz absolutamente nenhuma suposição da estrutura da esparsidade das matrizes e não armazena nenhum elemento desnecessário, sendo assim um método mais genérico. Como uma das propostas do presente trabalho é o desenvolvimento de algoritmos que possam ser aplicados a qualquer tipo de matrizes bidimensionais esparsas, o formato CSR atende a tal requisito.

ISSN: 1415-7314
ISSN online: 2317-6717

2.2 Métodos iterativos de resolução de sistemas de equações lineares

Os métodos iterativos explorados no presente trabalho foram: Jacob, Gauss-Seidel e Gradiente Conjugado nas suas versões disponíveis em Sperandio et al. (2003) para os dois primeiros e em Cunha (2000) para o terceiro método. Em virtude da limitação do número de páginas optou-se em suprimir a apresentação dos algoritmos dos métodos iterativos utilizados. O leitor interessado pode consulta-los nas referências citadas anteriormente.

O método Gradiente Conjugado possui uma particularidade adicional para a matriz dos coeficientes, a mesma necessariamente deve ser simétrica, assim o sistema linear inicialmente gerado foi condicionado para atender tal particularidade. Ambos os lados do sistema representado por A*x=b foi multiplicado por At , ou seja, At*A*x= At*b, em que At representa a transposta de A. Desta forma foi necessário elaborar um algoritmo, utilizando o sistema de armazenamento CSR, tanto para o produto A*x, quanto para o produto da matriz transposta At *A*x, tarefas um tanto quanto não triviais. Como afirma Connor (1984) códigos de matrizes esparsas não são fáceis de desenvolver, em geral, uma sub-rotina para matriz esparsa é 3 a 10 vezes mais longa do que sua matriz densa equivalente sendo de difícil entendimento e modificação.

2.3 Geração do sistema de equações lineares

Para a geração de sistemas lineares do tipo A*x = b com $A \in R^{n\times n}$ e $b \in R^{n}$, foi elaborado um algoritmo a partir de alguns ajustes no sistema linear obtido da discretização por diferenças finitas da seguinte equação diferencial parcial.

$$\frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} + a\varphi + b(x,y) = 0 \qquad (1)$$

em que as condições de contorno foram:

$$\frac{\partial\varphi}{\partial\eta} = c \qquad (2)$$

η representa a direção normal a cada aresta do domínio computacional e *a*,*b* e *c* são constantes. Os ajustes realizados foram necessários para permitir a variação do índice de esparsidade, uma vez que a discretização da Eq. (1) e (2) via diferenças finitas, resulta em um sistema linear de alta esparsidade. O algoritmo adapta o sistema linear de acordo com o grau

ISSN: 1415-7314
ISSN online: 2317-6717

(índice) de esparsidade definido pelo usuário. A adaptação gera de forma randômica elementos que reduzem a esparsidade das matrizes. O índice de esparsidade (GE) pode ser definido como:

$$GE = 100\% \frac{Número\ de\ Elementos\ Nulos}{Número\ total\ de\ Elementos} \quad (3)$$

Apenas para exemplificação, na Fig. 2 são mostradas duas matrizes 25x25 com esparsidade de 30% geradas pelo código. Pode-se observar que as duas matrizes são diferentes, e que a distribuição dos elementos não nulos é diferente, contudo ambas as matrizes possuem mesmo grau de esparsidade.

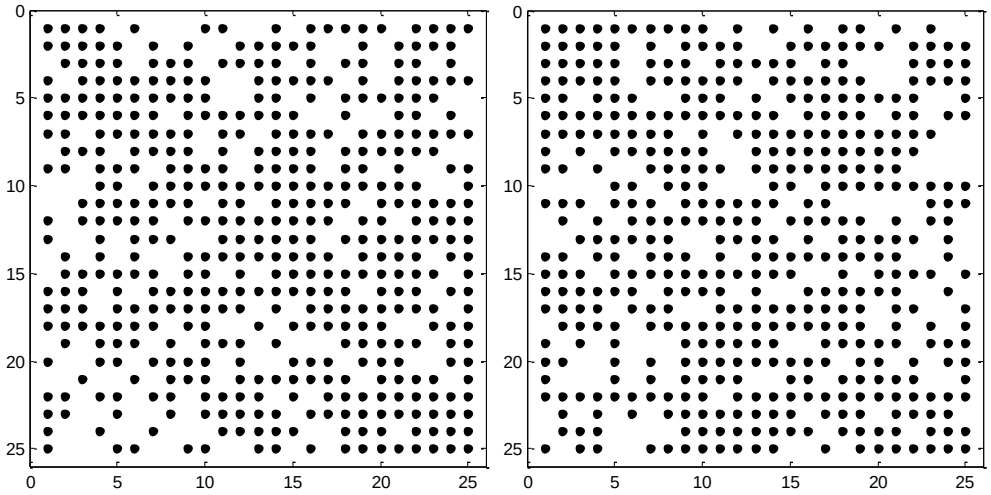

Figura 2: Matrizes geradas para composição dos sistemas lineares, matrizes 25x25 de esparsidade 30%.

3. Resultados

No intuito de prover maior clareza na apresentação e discussão dos resultados, esta seção foi subdividida em Armazenamento e Solução de Sistemas de Equações Lineares. Todos os resultados apresentados foram obtidos com um notebook com as seguintes configurações: Processador *Core* i5 2.6 Ghz, memória *ram* 8 GB e Sistema operacional Windows 7 - 64 bits. A linguagem de programação utilizada foi a do ambiente Matlab R12, entretanto não se utilizou comandos ou funções prontas disponíveis deste ambiente de programação, desta forma os algoritmos elaborados permitem fácil adaptação para outras linguagens. É importante mencionar que o uso adequado de funções prontas do Matlab pode alterar significativamente a velocidade de processamento, entretanto os códigos desenvolvidos ficariam restritos ao ambiente de programação do Matlab. Informações adicionais sobre o desempenho de códigos no ambiente Matlab, quanto à velocidade de processamento serão discutidas na próxima seção, o leitor

ISSN: 1415-7314
ISSN online: 2317-6717

interessado também pode consultar para informações adicionais as referências Kouatchou (2009) e Altman (2015).

3.1. Armazenamento

As matrizes geradas por meio do algoritmo apresentado na seção anterior foram comprimidas utilizando o método CSR. Matrizes quadradas foram geradas com ordens de 100x100, 400x400, 900x900, 1600x1600 e 2500x2500 com índices de esparsidade de 30, 60, 90 e 98%. Os resultados quanto o consumo de memória computacional com e sem compressão e o tempo de processamento para o armazenamento são apresentados na Tab. 1. Se faz importante mencionar que o tempo de processamento foi obtido pela média simples de experimentos realizados em quadruplicata.

Tabela 1: Memória Consumida para o Armazenamento de Matrizes.

Ordem da Matriz	GE(%)	Memória Utilizada Padrão (MBytes)	Memória Utilizada CSR (MBytes)	Redução de Consumo de Memória (%)	Tempo de Processamento para Compressão (s)
100x100	30	0.64	0.69	-7.4	0.0110
	60	0.64	0.41	35.2	0.0069
	90	0.64	0.14	77.9	0.0023
	98	0.64	0.07	89.7	0.0017
400x400	30	10.24	10.84	-5.8	0.1593
	60	10.24	6.25	39.0	0.0992
	90	10.24	1.71	83.3	0.0343
	98	10.24	0.50	95.1	0.0131
900x900	30	51.84	54.66	-5.4	0.8315
	60	51.84	31.36	39.5	0.5187
	90	51.84	8.20	84.2	0.1705
	98	51.84	2.00	96.1	0.0648
1600x1600	30	163.84	172.41	-5.2	2.7331
	60	163.84	98.62	39.8	1.6568
	90	163.84	25.26	84.6	0.6853
	98	163.84	5.71	96.5	0.2350
2500x2500	30	400.00	420.40	-5.1	6.6144
	60	400.00	240.86	39.8	4.0329
	90	400.00	61.03	84.7	1.4011
	98	400.00	13.25	96.7	0.5284

ISSN: 1415-7314
ISSN online: 2317-6717

Os dados da Tabela 1 indicam quando é vantajosa, em termos de economia de memória, a utilização da compressão CSR, verifica-se que há economia para matrizes com grau de esparsidade acima de 50%, independente da ordem da matriz, isto se justifica devido ao fato que no formato CSR são armazenados todos os elementos não nulos, suas respectivas colunas e a posição do primeiro elemento não nulo de cada linha. Assim, para matrizes de menor esparsidade a compressão CSR acaba gerando um número maior de elementos para serem armazenados. Os resultados apontam que o número de elementos salvos no formado CSR pode ser calculado a partir da seguinte equação:

$$Nº\ Elementos = \frac{2(100-Esparsidade)ordem^2}{100} + ordem \quad (4)$$

em que Esparsidade varia de 0 a 100%.

Ainda em relação à economia de memória, verifica-se que para diferentes ordens a porcentagem de redução é praticamente a mesma, sendo dependente apenas do grau de esparsidade.

Os resultados quanto ao tempo de processamento médio sumarizado na Tab. 1 sinalizam que o tempo para compressão depende fortemente da ordem e do índice de esparsidade da matriz. Naturalmente, a dependência quanto à ordem se deve ao fato do número de elementos da matriz ser proporcional a ordem da matriz. Já a dependência do índice de esparsidade se deve a operação de armazenamento dos valores não nulos nos vetores do formato CSR, ou seja, um tempo maior de processamento é dedicado para a alocação de memória para o elemento em questão.

Faz-se importante comentar que a forma a qual o armazenamento é realizado no ambiente de programação pode influenciar o tempo de processamento como um todo. Especificamente no caso do ambiente Matlab, se faz necessário tomar algumas medidas para o processamento mais rápido de um algoritmo, como por exemplo: pré-alocar a memória necessária para armazenar os elementos dos vetores e matrizes. Devido à forma na qual matrizes são representadas internamente no Matlab, não havendo pré-alocação, a cada nova alocação de um elemento, são atualizados todos os outros elementos que já foram alocados, tornando-se um gargalo de desempenho, o que pode prejudicar significativamente qualquer aumento potencial de velocidade de processamento.

Outra boa prática recomendada no desenvolvimento de códigos no ambiente Matlab é preferencialmente fazer o armazenamento em uma coluna com *n* linhas ao invés de uma linha com *n* colunas, uma vez que o Matlab, diferente de outras linguagens, executa o armazenamento em linha. Como comenta Altman (2015), embora no Matlab nenhum espaço seja alocado para

ISSN: 1415-7314
ISSN online: 2317-6717

índices não utilizados, o espaço é realmente alocado para todos os índices de coluna possíveis como uma sobrecarga global. Por esta razão, é geralmente melhor definir os dados de tal modo que a dimensão maior seja armazenada como uma linha de dados, em vez de colunas, isto é, assegurar uma dimensão menor para a coluna.

Ainda em relação ao tempo de processamento com o uso do ambiente Matlab, é importante destacar que o uso de funções prontas disponíveis no Matlab, geralmente são executadas em um tempo de processamento menor que uma função com o mesmo objetivo elaborada por um usuário. Em um relatório elaborado pela Kouatchou (2009) foi comparado o tempo de processamento da operação de multiplicação de uma matriz quadrada por um vetor, sendo as ordens das matrizes de 1000x1000, 1200x1200 e 1500x1500, usando as linguagens gfortan, Java, Phyton, ifort e Matlab, este último usando a função de multiplicação disponível no Matlab, e também para comparação, os algoritmos de multiplicação *trip do-loop, matmul, O3* com *triple do-loop* e e *O3* com *matmul* utilizado nas outras linguagens testadas. Em resumo, os resultados mostraram que o tempo de processamento utilizando a função de multiplicação pronta do Matlab foi a mais rápida em todos os testes, apenas para exemplificar no caso da matriz de ordem 1500 o tempo de processamento utilizando a função intrínseca foi de 0,97 s enquanto utilizando o algoritmo *trip do-loop* foi de 385,95 s.

Neste contexto, é possível reduzir ainda mais o tempo de processamento para a compressão de matrizes obtidos no presente trabalho se utilizado as funções intrínsecas do Matlab, entretanto, tal opção não foi executada no presente trabalho para permitir que os algoritmos desenvolvidos também possam ser facilmente adaptados para serem utilizados em outras linguagens.

3.2. Solução de Sistemas de Equações Lineares

A fim de verificar o desempenho em termos de tempo de processamento do formato CSR associado aos métodos iterativos de Jacob, Gauss-Seidel e Gradiente Conjugado, experimentos numéricos foram realizados em quadruplicata com e sem o esquema CSR. Entretanto, antes da apresentação dos resultados é importante destacar um fator importante quanto ao formato CSR. O formato de armazenamento da matriz dos coeficientes não modifica a forma com o qual os métodos iterativos trabalham, ou seja, nenhuma alteração é realizada na essência do método iterativo, apenas a forma em que os vetores são processados é alterada. Desta forma, o perfil do resíduo em função do número de interações deve ser o mesmo, independente do uso ou não do formato CSR. Para exemplificação, na Fig. 3 são apresentados os resíduos obtidos em função do número de iterações do método Gauss-Seidel com e sem o referido esquema de compressão para o caso de uma matriz 20 x 20 e esparsidade de 98%.

ISSN: 1415-7314
ISSN online: 2317-6717

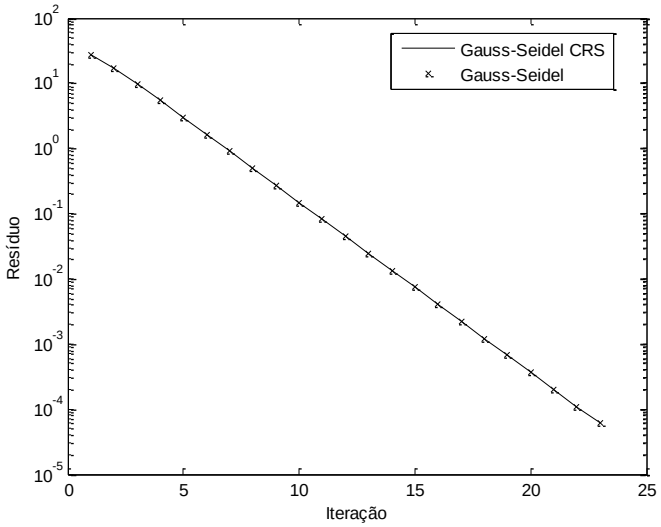

Figura 3: Resíduo obtido pelo método Gauss-Seidel, matriz 20x20 e índice de esparsidade de 98%. Resíduo definido como máximo$|A*x^{(i)} - b| <= 10^{-4}$.

Na Tab. 2 são sumarizados os resultados referentes aos experimentos, sendo indicadas a ordem e esparsidade das matrizes, o tempo de processamento de máquina para resolução do sistema linear, bem como o número de iterações para os três métodos iterativos. É importante ressaltar que para qualquer um dos três métodos testados o número de iterações não se modifica com ou sem a compressão CSR. Novamente, isso ocorre porque a essência dos métodos não é alterada, apenas a forma de armazenamento da matriz é modificada.

Tabela 2: Tempo de Processamento para Resolução do Sistema de Equações Lineares e número de iterações.

Ordem da Matriz	GE (%)	Jacob (s)	CRS Jacob (s)	Jacob Iterações	Gauss-Seidel (s)	CRS Gauss-Seidel (s)	Gauss-Seidel Iterações	Grad. Conj. (s)	CRS Grad. Conj.(s)	Grad. Conj. Iterações
100x100	30	0.12	0.13	141	0.07	0.08	76	0.01	0.02	10
	60	0.08	0.06	90	0.04	0.04	49	0.02	0.02	13
	90	0.03	0.01	35	0.02	0.01	21	0.02	0.01	18
	98	0.02	0.01	20	0.01	0.005	13	0.02	0.01	17
400x400	30	10.57	8.65	618	5.85	5.10	333	0.17	0.30	9
	60	5.57	3.11	379	3.52	1.88	204	0.22	0.20	11
	90	1.88	0.32	117	1.07	0.20	64	0.34	0.08	17
	98	0.68	0.05	40	0.40	0.04	23	0.44	0.04	23
900x900	30	126.87	91.43	1331	70.07	53.66	719	0.95	1.48	9
	60	75.76	32.55	814	41.94	19.18	439	1.11	0.95	10
	90	22.37	2.75	244	12.38	1.66	132	1.68	0.35	15
	98	6.19	0.26	68	3.52	0.19	38	2.51	0.16	23

ISSN: 1415-7314
ISSN online: 2317-6717

1600x1600	30	706.09	449.97	2247	388.78	261.76	1220	3.17	4.58	8
	60	447.57	170.92	1376	246.81	100.60	744	3.51	2.86	9
	90	131.00	13.97	408	72.75	8.36	219	5.03	1.17	13
	98	33.63	1.09	106	18.87	0.74	58	8.09	0.47	21
2500x2500	30	2502.40	1624.66	3342	1398.13	947.94	1822	7.61	11.01	8
	60	1739.38	633.29	2053	950.59	370.99	1113	8.52	6.90	9
	90	480.56	49.14	607	264.20	28.98	327	11.50	2.47	12
	98	119.63	3.22	153	66.41	2.17	83	17.47	1.00	19

Exceto para o método de CSR Gradiente Conjugado, os dados da Tab. 2 sugerem que quanto maior a esparsidade menor é o número de iterações e menor é o tempo para resolução do sistema, o que é uma consequência do número de zeros em matrizes muito esparsas.

Torna-se interessante também mostrar explicitamente a relação de redução do tempo de processamento para quando os métodos iterativos operam no formato CSR, na Tab. 3 são indicadas a redução em porcentagem do tempo que cada método quando operado no formato tradicional e no formato CSR e também a razão entre o número de iterações do método de Jacob com os métodos Gauss-Seidel e Gradiente Conjugado.

Em se tratando da influência da compressão, observa-se tanto na Tab. 02 e 03, que exceto para o Gradiente Conjugado com matrizes de esparsidade de 30% e para o método de Jacob para a matriz 100x100 de esparsidade 30%, a formatação CSR reduz significativamente o tempo de processamento para resolução para resolução do sistema linear. É possível analisar ainda que essa redução é diretamente proporcional ao índice de esparsidade da matriz. Isso ocorre porque o objetivo da compressão é eliminar as operações em que o elemento da matriz é zero, quanto maior o grau de esparsidade maior o número de zeros e portanto maior o número de operações suprimidas. Como exemplo, para o caso das matrizes de ordem 2500x2500 a redução média no tempo de processamento para as esparsidades de 30, 60 90 e 98% são respectivamente, 34%, 61%, 89% e 96%, coincidentemente, os valores de esparsidade e de redução de tempo de processamento são numericamente muito próximos.

Tabela 3: Desempenho da formatação CSR e dos métodos iterativos

		Redução de Tempo de Processamento (%)			Razão entre o número de iterações	
Ordem da Matriz	**GE (%)**	**Jacob**	**Gauss-Seidel**	**Grad. Conjug.**	**Jacob/ (Gauss-Seidel)**	**Jacob/ (Grad. Conjug.)**
100x100	30	-8.5	-20.4	-65.3	1.8	13.8
	60	30.0	6.3	-17.4	1.8	6.9
	90	64.3	60.1	51.1	1.7	2.0

ISSN: 1415-7314
ISSN online: 2317-6717

	98	73.7	71.8	58.8	1.5	1.2
400x400	30	18.2	12.8	-71.9	1.9	68.7
	60	44.1	46.4	10.7	1.9	33.6
	90	82.9	81.7	76.1	1.8	6.9
	98	93.1	90.5	90.7	1.7	1.8
900x900	30	27.9	23.4	-55.2	1.9	156.6
	60	57.0	54.3	13.9	1.9	81.4
	90	87.7	86.6	79.1	1.9	16.0
	98	95.7	94.7	93.6	1.8	3.0
1600x1600	30	36.3	32.7	-44.5	1.8	280.8
	60	61.8	59.2	18.4	1.8	152.9
	90	89.3	88.5	76.8	1.9	31.3
	98	96.8	96.1	94.2	1.8	5.0
2500x2500	30	35.1	32.2	-44.6	1.8	417.8
	60	63.6	61.0	19.1	1.8	228.1
	90	89.8	89.0	78.5	1.9	49.6
	98	97.3	96.7	94.3	1.8	8.2
MédiaSimples:		**61.8**	**58.2**	**27.8**	**1.8**	**78.3**
Desv. Padrão:		30.9	34.3	59.5	0.1	114.1

Ainda em relação à redução no tempo de processamento na utilização do formato CSR, é possível observar que quanto maior a ordem e o grau de esparsidade da matriz maior será a vantagem da aplicação da compressão.

Como destacado nos parágrafos anteriores o método de Gradiente Conjugado, mostrou um comportamento sistêmico, porém distinto dos outros dois métodos iterativos, o número de iterações necessário para atingir o resíduo especificado aumentou com o índice de esparsidade das matrizes, comportamento este contrário ao observado com os métodos de Jacob e Gauss-Seidel. Pode-se também observar que para os casos de esparsidade 30% o formato CRS não proveu redução no tempo de processamento, no entanto, para esparsidades maiores o formato CSR proveu uma significativa redução.

Ainda em relação ao comportamento do método Gradiente Conjugado, observa-se que o tempo de processamento aumenta com a esparsidade das matrizes, no seu formato original, isto é, sem a formatação CSR. Entretanto, o inverso ocorre quando o método está formatado na forma CSR, sugerindo, neste caso, que o tempo de processamento não é diretamente relacionado apenas ao número de interações, mas sim ao número de operações realizadas a cada ciclo do método, consequentemente, mesmo com um número maior de iterações para matrizes mais esparsas, o número de operações matemáticas e de armazenamento realizadas dentro de cada iteração é significativamente menor para matrizes mais esparsas do que para matrizes menos esparsas. Dentro de cada ciclo ou iteração do método Gradiente Conjugado, sem levar em conta

ISSN: 1415-7314
ISSN online: 2317-6717

o pré-condicionamento, comentado na seção anterior, para transformar a matriz A em uma matriz simétrica, são necessárias as seguintes operações matemáticas: o produto da matriz A por um vetor, dois produtos internos e três somas de vetores, operações estas que consomem mais tempo à medida que o sistema linear se torna menos esparso.

Em relação à comparação dos três métodos iterativos, os resultados da Tab. 2 e 3 indicam que o método Gradiente Conjugado proveu um desempenho superior aos outros dois métodos, em geral o método Jacob realizou 1,8 ± 0,1 vezes mais iterações que o método Gauss-Seidel, entretanto, para o método Gradiente Conjugado a razão do número de iterações não se mostrou tão homogênea, variando de 1,2vezes para matrizes 100x100 de esparsidade de 98% e 417,8 vezes para matrizes 2500x2500 de esparsidade 30%.

4. Conclusão

No presente trabalho foi apresentado o problema de armazenamento e de resolução de grandes sistemas de equações lineares esparsos por meio dos métodos iterativos: Jacob, Gauss-Seidel e Gradiente Conjugado. Os resultados em geral sugerem fortemente para vantagens na utilização da compressão/formato CSR em termos economia de memória computacional e de tempo de processamento, principalmente para matrizes de altas ordens e alto índice de esparsidade. Pormenorizadamente, conclui-se que: (i) em relação à redução do consumo de memória computacional, o esquema CSR se mostrou vantajoso para matriz com grau de esparsidade maior que 50%, independente da ordem da matriz; (ii) exceto para matrizes de ordem 100x100 de esparsidade inferiores a 60%, mesmo nos casos que não houve redução do armazenamento de dados com uso do formato CSR, uma vez que as operações com elementos nulos foram suprimidas, ocorreu uma redução significativa de 6,3 a 96,7% no tempo de processamento para a resolução dos sistemas lineares por meio dos métodos Jacob e Gauss-Seidel; (iii) o tempo de processamento do método Gradiente no formato CSR se tornou cada vez menor com índice de esparsidade do sistema linear; (iv) o método Jacob realizou 1,8 ± 0,1 vezes mais iterações que o método Gauss-Seidel, entretanto, para o método Gradiente Conjugado a razão do número de iterações para o método de Jacob não se mostrou tão homogênea, variando de 1,2 a 417,8 vezes; (v) dentre os três métodos iterativos testados, o Gradiente Conjugado foi o que atingiu o resíduo especificado com o menor número de iterações e menor tempo de processamento, seguido pelos métodos de Gauss-Seidel e de Jacob.

5. Referências bibliográficas

ALTMAN, Y. M. 2015. *Accelerating MATLAB Performance: 1001 tips to speed up MATLAB programs*. Taylor and Francis Group, New York.

ISSN: 1415-7314
ISSN online: 2317-6717

BATHE, K.-J. & WILSON, E. L. 1976. *Numerical Methods in Finite Element Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey.

BRAMELLER, A., ALLAN, R.N., HAMAM Y.M., Sparsity, Pitman, New York, 1976.

COELHO, M. A. O. 2014. *Métodos Iterativos para Resolver Sistemas de Equações Algébricas Lineares em Estruturas Esparsas.* Dissertação de Mestrado, Universidade Federal Fluminense.

CUNHA, M. C. C. 2000. *Métodos Numéricos*. Campinas, Editora Unicamp.

DEREK O. C., Report on the Dublin matrix theory conference, March 1984: An introduction to sparse matrices, Pages 271-272

DUFF, I.S. 1997. A survey of Sparse Matriz Research, Proc. IEE, v 65, p. 500-535.

FARZANEH, A., KHEIRI H. & SHAHMERSI M. A. 2009. An Efficient Storage Format for Large Sparse Matrices, *Commun. Fac. Sci. Univ. Ank. Series A1*, 58, Number 2, 1-10.

GEORGE R., 2007. Robs algorithm, *Applied Mathematics and Computation*, 189, 314–325.

KOUATCHOU, J. 2009. Comparing Python, NumPy, Matlab, Fortran. Nasa Report 1762.

RIZWAN, B. 2007. *Introduction to Numerical Analysis Using MATLAB,* Massachusetts, Jones and Bartlett.

ROSE, D.J. WILLOUGHBY R.N. 1972. *Sparse Matrix and its Applications*, Plenum Pres, New York.

SAAD, Y. 1994. A Basic Tool Kit for Sparse Matrix Computations.

SAAD, Y. 2003. Iterative Methods for Sparse Linear Systems: Second Edition, Society for Industrial and Applied Mathematics Philadelphia, PA, USA.

ŜOLÍN, P. 2006. *Partial Differential Equations and the Finite Element Method*, Hoboken John Wiley & Sons.

SPERANDIO, D., MENDES, J. T., SILVA, L. H. M. 2003. *Cálculo Numérico - Características Matemáticas e Computacionais dos Métodos Numéricos*. Prentice Hall.

VERSTEEG, H.K. AND MALALASEKERA. 2007. *An Introduction to Computacional Fluid Dynamics: The Finite Volume Method*, Second Edition, Prentice Hall, England.