BUILDING PRECONDITIONERS USING BASIS MATRIX METHOD

Volodymyr Kudin, Vsevolod Bohaienko

***Abstract**: New class of preconditioners for iterative algorithms of sparse linear systems solution built using basis matrix method and incomplete decomposition methodology has been proposed. Algorithms with static and dynamic restriction set along with additional refinement procedures have been presented. Results of developed algorithms' testing carried out on matrices from Tim Davis Matrix Collection have been given. Basing on received results, matrix classes, applying proposed preconditioners on which resulted in iterative algorithms speed-up and/or accuracy increase, have been identified*

***Keywords**: linear systems, sparse matrices, poor conditioned matrices, preconditioners, basis matrix method.*

***ACM Classification Keywords**: H.4.2 Information Systems Applications: Types of Systems: Decision Support.*

Introduction

Matrix computations, especially linear systems solution problems, appear while doing mathematical modelling of most physical processes. In many cases, such linear systems have ill-conditioned large sparse matrices and are usually solved by iterative methods such as conjugate gradients (CG) or stabilized biconjugate gradients (BiCGstab) [Saad, 2003].

Preconditioning [Ke Chen, 2005] – methods based on multiplying linear systems' matrix by another matrix that results in condition number decrease, is a main technique used to achieve better convergence rate and/or accuracy of iterative methods. While inverse matrix is an ideal preconditioner, most preconditioners are built as its approximation. Most common methods are incomplete decompositions (e.g. incomplete LU decomposition) and incomplete inversions (e.g. polynomial preconditioners), among others multigrid and wavelet preconditioners can be distinguished.

It's worth noting that efficiency of preconditioners usage in most cases can't be theoretically proved, so there is a problem of experimental finding of matrix classes for which given preconditioner is efficient in sense of computations speed-up or solution accuracy improvement.

Preconditioners built upon basis matrix method

Incomplete decomposition is one of preconditioner building methods which consists in a restriction of a set of matrix elements changing in decomposition process, e.g. LU [Saad, 2003], QR [Ke Chen, 2005], or while applying inversion procedure, e.g. Gram-Schmidt method [Qiaohua Liu, 2013]. This yields to a matrix, which approximates inverse matrix and can be used as a preconditioner.

Such method can also be applied to algorithms of basis matrix method [Kudin, 2007]. Consider linear system $Ax=b$ where $\dim A=[n,n]$, $\dim x=\dim b=[n,1]$. Basis matrix methods' algorithm which results in matrix A^{-1}, inverse of A, may be stated as follows: Let $A_{б}^{(i)}$, $\dim A_{б}^{(i)}=[n,n]$ be a basis matrix on i-th iteration of algorithm, $A_{б}^{(i)}[j]$ be a column of matrix $A_{б}^{(i)}$, a_i be a row of matrix A, $A_{б}^{(0)}=A_{б}^{0}$. When i-th iteration of algorithm can be written as follows:1) $v=diag\left(AA_{б}^{(i-1)}\right)$; 2) $k=\max\limits_{i}|v_i|$; 3) $\alpha=a_kA_{б}^{(i-1)}$;

4) $A_б^{(i)}[k] = A_б^{(i-1)}[k] / \alpha_k$; 5) $A_б^{(i)}[l] = A_б^{(i-1)}[l] - \alpha_l A_б^{(i)}[k],\ l \neq k$;

After execution of n iterations, algorithm results in $A_б^{(n)} = A^{-1}$.

Algorithm of incomplete basis matrix method can be obtained from aforementioned algorithm by changing step 5 as follows: 5a) $A_б^{(i)}[l]_j = A_б^{(i-1)}[l]_j - \alpha_l A_б^{(i)}[k]_j,\ l \neq k,\ (l, j) \in R$, where R is a set of matrix elements' indices. Here, after execution of n iterations, algorithm results in $A_б^{(n)} \approx A^{-1}$.

Definition of set R is needed to obtain a particular algorithm. By analogy with incomplete LU decomposition algorithms [Saad, 1994], consider following incomplete basis matrix methods' algorithms: IBMM0, where R is a set of matrix A non-zero elements, and IBMM1, where R is a set of matrix AA^T non-zero elements.

Another variant of incomplete basis matrix methods' algorithm (also by analogy with other incomplete decomposition algorithms) is an algorithm with dynamically changing R. Let R be restricted in such way that (condition1) number of (l, j) elements in it can't be bigger that number of non-zero elements in l-th row of matrix A. This can be taken into account by changing step 5 as follows:

5б) If $(l, j) \in R$ or $(l, j) \notin R$, but condition 1 is met only for column l, then do following transformation: $A_б^{(i)}[l]_j = A_б^{(i-1)}[l]_j - \alpha_l A_б^{(i)}[k]_j,\ l \neq k,\ R = R \cup (l, j)$. Here element will be changed and added to the set if set does not contain it;

If $(l, j) \notin R$ and condition 1 is not met, then transformation $A_б^{(i-1)}[l]_m = 0, A_б^{(i)}[l]_j = A_б^{(i-1)}[l]_j - \alpha_l A_б^{(i)}[k]_j,\ R = \left(R - (l, m)\right) \cup (l, j)$ must be done if $\exists m : \left| A_б^{(i-1)}[l]_m \right| < \left| A_б^{(i-1)}[l]_j - \alpha_l A_б^{(i)}[k]_j \right|$: element will be changed and added to the set if column contains an element with lower absolute value. That element at the same time will be set to zero and removed from set R. We'll call such algorithm IBMMd.

Consider additional procedures which extend IBMM0, IBMM1 and IBMMd algorithms.

Condition $a_k A_б^{(i)}[l] = \begin{cases} 1,\ l = k \\ 0, l \neq k \end{cases}$ is met on every iteration of basis matrix method, but not during incomplete transformation. To make this condition met in IBMM0 and IBMM1 algorithms' iterations, following step must be added:

6) If $(l, j) \notin R$, $a_{kj} \neq 0$ and $\exists m = \max_j \left| a_{kj} \right|, A_б^{(i-1)}[l]_j \neq 0$ then transformation $A_б^{(i)}[l]_m = A_б^{(i)}[l]_m - \alpha_l A_б^{(i)}[k]_j a_{kj} / a_{km}$ must be done.

In the case of IBMMd algorithm, step 6 must be applied when $(l, j) \notin R$, condition 1 is not met and an element with lower absolute value does not exist in a column, or while substituting a non-zero element:

6a) If $(l, j) \notin R$, condition 1 is not met, $\neg \exists m : \left| A_б^{(i-1)}[l]_m \right| < \left| A_б^{(i-1)}[l]_j - \alpha_l A_б^{(i)}[k]_j \right|$ and $\exists m = \max_j \left| a_{kj} \right|, A_б^{(i-1)}[l]_j \neq 0$, then do following transformation: $A_б^{(i)}[l]_m = A_б^{(i)}[l]_m - \alpha_l A_б^{(i)}[k]_j a_{kj} / a_{km}$.

6б) If $(l, j) \notin R$, condition 1 is not met, $\exists m : \left| A_б^{(i-1)}[l]_m \right| < \left| A_б^{(i-1)}[l]_j - \alpha_l A_б^{(i)}[k]_j \right|$ and $\exists p = \max_j \left| a_{kj} \right|, A_б^{(i-1)}[l]_j \neq 0,\ j \neq m$, then do following transformation $A_б^{(i)}[l]_p = A_б^{(i)}[l]_p - \alpha_l A_б^{(i)}[k]_m a_{km} / a_{kp}$.

We'll designate algorithms with correction step 6 as IBMM0+c, IBMM1+c, IBMMd+c.

Taking into consideration efficiency of Jacobi preconditioner while solving many linear systems, it can be combined with incomplete basis matrix methods' algorithms. In such case, left preconditioner will take the following form: $\left[diagA_{б}^{(n)}A\right]^{-1}A_{б}^{(n)}$. We'll designate such algorithms as “+rs”.

Refinement procedure [Bohaienko, 2009] consisting in iterative algorithm execution with $A_{б}^{(0)}=A_{б}^{(n)}$, can be also applied to incomplete basis matrix methods' algorithms.

Testing of preconditioners efficiency

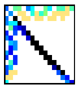
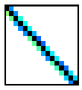
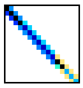
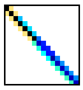
Efficiency of matrices, generated by developed algorithms, as left preconditioners has been tested on matrices from *Tim Davis Matrix Collection* (http://www.cise.ufl.edu/research/sparse/matrices/index.html) while solving corresponding linear problems using Bicgstab algorithm. Number of iterations after which given accuracy was achieved is given in Table 1. Maximal iteration number was set to 1000. All elements of vector b was equal to 1, and $A_{б}^{0}=I$. Here information about matrix is given to the right of its' name, nnz(A) is a number of non-zero elements and cond(A) is a condition number.

Table 1. Number of iterations after which accuracy ε was achieved

			1	1	1	1	1	1	1	2	2	2	2	2	2	Number of refinement iteration
			-	-	-	-	-	-	+	-	-	-	-	+	+	Correction steps
Matrix and $\log_{10}\varepsilon$	-	+	-	-	-	+	+	+	-	-	-	+	+	-	+	Jacobi preconditioner
	-	-	0	1	d	0	1	d	0	0	1	0	1	1	1	Algorithm: 0 – IBMM0, 1 – IBMM1, d - IBMMd
bfwa62	Collection *Bai*, dimA=[62,62], nnz(A)=450, cond(A)=553, electro-dynamical problem															
-2	26	20	57	-	-	-	-	-	-	-	-	-	-	-	-	
-4	35	23	61	-	-	-	-	-	-	-	-	-	-	-	-	
-6	37	31	-	-	-	-	-	-	-	-	-	-	-	-	-	
-8	42	33	-	-	-	-	-	-	-	-	-	-	-	-	-	
-10	44	34	-	-	-	-	-	-	-	-	-	-	-	-	-	
bfwa398	Collection *Bai*, dimA=[398,398], nnz(A)=3678, cond(A)=2993, electro-dynamical problem															
-2	72	54	269	-	226	313	-	195	-	-	-	-	-	-	-	
-4	103	78	295	-	230	317	-	213	-	-	-	-	-	-	-	
-6	107	87	304	-	266	317	-	245	-	-	-	-	-	-	-	
-8	108	89	-	-	286	-	-	281	-	-	-	-	-	-	-	
-10	118	90	-	-	-	-	-	-	-	-	-	-	-	-	-	
olm100	Collection *Bai*, dimA=[100,100], nnz(A)=396, cond(A)=15275, hydrodynamic problem															
-2	82	-	73	-	-	-	-	-	-	56	-	-	-	35	58	
-4	83	-	79	-	-	-	-	-	-	56	-	-	-	35	61	
-6	-	-	-	-	-	-	-	-	-	-	-	-	-	35	61	
poli	Collection *Grund*, dimA=[4008,4008], nnz(A)=8188, cond(A)=311, economical problem															
-6	17	17	8	6	-	8	7	-	-	8	8	8	10	-	-	
-10	20	20	9	9	-	9	9	-	-	9	9	9	12	-	-	
-14	-	-	-	-	-	-	-	-	-	-	-	9	12	-	-	

A more detailed look has been taken on developed algorithms' efficiency when applying them to matrices from *Averous* collection: *epb0*, *epb1*, *epb2*, *epb3*. Characteristics of these matrices which arise from thermodynamical problems are given in Table 2.

Table 2. Characteristics of *Averous* collection matrices

Matrix	Number of rows	nnz(A)	cond(A)	Sparsity pattern
epb0	1794	7764	64165	
epb1	14734	95053	5940	
epb2	25228	175027	2618	
epb3	84617	463625	-	

Data, same as in Table 1, are given in Table 3.

Table 3. Number of iterations after which accuracy ε was achieved for *Averous* collection matrices

			1	1	1	1	1	1	2	2	2	Number of refinement iteration
			-	-	-	-	-	-	-	-	-	Correction steps
Matrix and $\log_{10}\varepsilon$	-	+	-	-	-	+	+	+	-	-	+	Jacobi preconditioner
	-	-	0	1	d	0	1	d	0	1	0	Algorithm: 0 – IBMM0 1 – IBMM1 d - IBMMd
epb0												
-4	-	918	-	-	-	663	-	-	490	-	490	
-6	-	992	-	-	-	675	-	-	526	-	518	
-8	-	-	-	-	-	772	-	-	548	-	528	
-10	-	-	-	-	-	815	-	-	585	-	567	
-12	-	-	-	-	-	838	-	-	588	-	575	
epb1												
-4	358	321	231	210	125	187	199	133	255	272	279	
-6	426	407	274	260	160	233	228	166	383	385	399	
-8	517	488	343	303	198	295	262	201	434	505	423	

-10	541	517	363	348	203	299	325	212	466	518	430	
-12	573	552	405	352	211	336	331	221	490	664	488	
epb2												
-4	192	111	91	96	-	85	95	-	96	-	111	
-6	259	128	116	118	-	111	119	-	139	-	148	
-8	315	166	133	132	-	133	130	-	181	-	176	
-10	461	191	162	159	-	157	175	-	198	-	231	
epb3												
7	120	99	105	-	-	-	-	-	103	-	-	
5	291	258	428	-	-	-	-	-	542	-	-	
3	845	297	511	-	-	-	-	-	566	-	-	
1	-	616	569	-	-	-	-	-	661	-	-	

Change of residual logarithm while solving linear system with *epb0* matrix applying different preconditioners is presented on Figure 1.

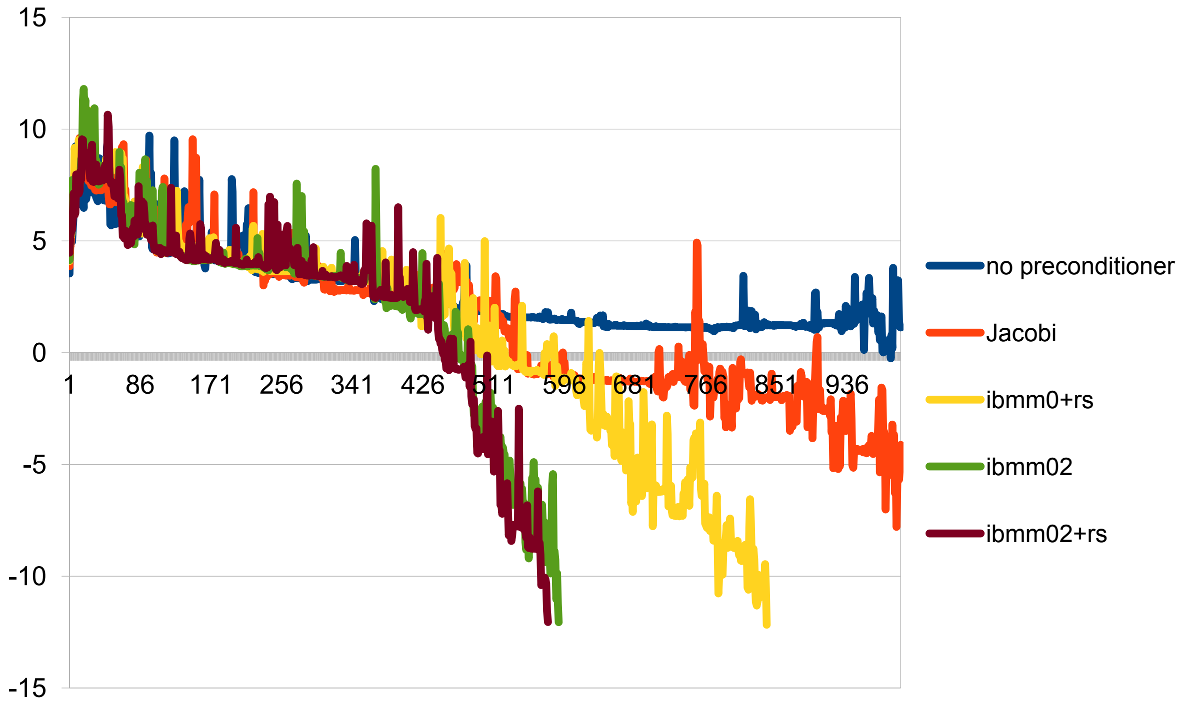


Figure1. $\log_{10} \varepsilon$ while solving linear system with epb0 matrix depending on iteration number

Conclusions

According to numerical experiments' results, different modifications of developed algorithms are efficient while using them with different matrices.

It's worth noting that, in general, they are more efficient when applying to matrices with bigger conditional number. Particularly, when using them with relatively poor conditioned matrices olm100 and epb0, several orders of magnitude increase of accuracy was achieved and applying to epb1 matrix yields to twice less number of iterations needed to achieve given accuracy. On the other side, no positive effects were observed when applying them to relatively good conditioned matrices bfwa62, bfwa398, epb2 and epb3.

Reduction of needed number of iterations along with accuracy increase was observed upon applying incomplete basis matrix methods' preconditioners too relatively good conditioned matrix *poli* which arises from economical modelling problem. Taking this feature into consideration, similar matrices could be used as an object of further research.

Acknowledgements

The work is published with the financial support of the project ITHEA XXI Institute of Information Theory and Applications FOI ITHEA Bulgaria www.ithea.org Association and the creators and users of intellectual systems ADUIS Ukraine www.aduis.com.ua.

Bibliography

[Bohaienko, 2009] Vsevolod A. Bohaienko, Vladimir I. Kudin, Vasiliy V. Skopetsky Analysis of Computational Schemes for Geohydrodynamics Processes Modeling // Journal of Automation and Information Sciences - Volume 41, 2009 Issue 8 – pp. 1-12.

[Ke Chen, 2005] Ke Chen Matrix Preconditioning Techniques and Applications. - Cambridge University Press, 2005. - 592 p.

[Kudin, 2007] Kudin V.I., Liashko S.I., Iatsenko Iu.P., Kharytonenko N.V. Metod shtuchnykh bazysbyh matryts'// Dopovidi NAN Ukraiiny, 9, 2007, pp.30-34, in ukrainian

[Saad, 1994] Saad Y. ILUT: a dual threshold incomplete LU factorization // Numer. Linear Algebra Appl. 1(4), 387–402 [Qiaohua Liu, 2013] Qiaohua Liu, Fudao Zhang Incomplete hyperbolic Gram–Schmidt-based preconditioners for the solution of large indefinite least squares problems // Journal of Computational and Applied Mathematics 250 (2013) 210–216

[Saad, 2003] Saad Y. Iterative methods for sparse linear systems, 2 edition. - Society for Industrial and Applied Mathematics, 2003. – 528 p.

Authors' Information

Volodymyr Kudin *– Doctor of Science, senior researcher, Taras Shevchenko National University of Kyiv; e-mail:V_I_Kudin@mail.ru*

Scientific interests: optimization methods, systems analysis, methods of computations, decision making, mathematical economics.

Vsevolod Bohaienko *– PhD, senior researcher, V.M. Glushkov Institute of Cybernetics of NAS of Ukraine; e-mail: sevab@ukr.net*

Scientific interests: mathematical modeling in ecology and energetic, parallel algorithms, linear algebra, optimization methods.