POLYNOMIAL PRECONDITIONERS BASED ON FACTORIZED SPARSE APPROXIMATE INVERSES *

J. CERDÁN †, J. MARÍN ‡, AND A. MARTÍNEZ §

Abstract. Let $Ax = b$ be a linear system where A is a symmetric positive definite matrix. An additive polynomial preconditioner for the Conjugate Gradient method based on multisplittings is proposed. The multisplittings are obtained by computing some factorized sparse approximate inverses of the coefficient matrix. Namely, splittings of the form $A = (\bar{Z}\bar{Z}^T)^{-1} - N$, $\bar{Z}\bar{Z}^T \approx A^{-1}$ induced by the AINV and FSAI factorized approximate inverse preconditioners applied to diagonal blocks of A are used. The applicability of this preconditioner is studied. Moreover, the results of the numerical experiments obtained on a Cray T3E for a representative set of matrices are presented. Specifically, structural analysis and transport/diffusion problems are considered. The effect of the *Reverse Cuthill-McKee* and *Multiple Minimum Degree* orderings is evaluated.

Key words. Multisplitting, Parallel Polynomial Preconditioner, Factorized Sparse Approximate Inverse.

AMS subject classifications. 15A12, 65F10, 65F35, 65Y05

1. Introduction.

The main goal of preconditioning a given linear system

$$Ax = b \tag{1.1}$$

is to reduce the amount of time needed to get a good approximation of the system solution when it is obtained by applying some iterative method. Thus, the iterative method is applied to the new preconditioned system

$$\mathcal{M}^{-1}Ax = \mathcal{M}^{-1}b, \qquad \text{or} \quad A\mathcal{M}^{-1}y = b, \qquad x = \mathcal{M}^{-1}y \ , \tag{1.2}$$

where $\mathcal{M}^{-1}$ is the preconditioning matrix. If the preconditioned matrix $\hat{A} = \mathcal{M}^{-1}A$ (or $\hat{A} = A\mathcal{M}^{-1}$) has a smaller condition number than A or its eigenvalues are better clustered, then a small number of iterations are needed to achieve convergence to a solution.

When the matrix A is symmetric positive definite (spd), one of the most competitive and widely used methods for solving (1.1) is the Preconditioned Conjugate Gradient method (PCG). This minimization method of conjugate directions consists of solving the new linear system

$$\hat{A}\hat{x} = \hat{b}$$

where $\hat{A} = SAS^T$, $\hat{x} = S^{-T}x$ and $\hat{b} = Sb$. In this case, the preconditioning matrix $\mathcal{M} = (S^TS)^{-1}$ also has to be symmetric positive definite. The preconditioning step is done by solving an auxiliary linear system

$$\mathcal{M}s = r \ , \tag{1.3}$$

---

*This work was supported by DGES project PB97–0334

†Departament de Matemàtica Aplicada, Universitat Politècnica de València, 46071 València, Spain (`jcerdan@mat.upv.es`).

‡Correspondence author: Departament de Matemàtica Aplicada, ETSIA, Universitat Politècnica de València, 46071 València, Spain (`jmarinma@mat.upv.es`).

§Dipartamento di Metodi e Modelli Matematici per le Scienze Applicate. Università di Padova. Via Belzoni, 7 Padova, Italia (`acalomar@dmsa.unipd.it`).

where $r = b - Ax$ is the residual corresponding to the approximate solution obtained in the previous iteration. Since this represents an overhead it is crucial to use techniques for constructing and applying preconditioners easily. This last requirement is even more important in parallel computing. Several preconditioning techniques have been proposed in the literature. Many of the most popular preconditioners, such as those based on incomplete factorizations of A, provide good convergence rates but they are highly sequential, making difficult their efficient implementation on parallel computers. Another technique is polynomial preconditioning based on multisplittings [2],[15],[16]. Polynomial preconditioners are attractive for parallel machines, specially on distributed memory computers, since they require only matrix-vector products. Another attractive feature is that they are easily combined with other preconditioners, as we do in this work, and require almost no additional storage besides that needed for the coefficient matrix A. Their main drawback is that they are not as effective as those based on incomplete factorizations of A at reducing the number of iterations.

In the last years there has been considerable interest on another class of preconditioners based on sparse approximate inverses. These methods approximate the inverse of matrix A, as with polynomial preconditioning, but in this case a matrix $\mathcal{M} \approx A^{-1}$ is explicitly computed and stored. There are different techniques to construct an approximate inverse. Among them, factorized approximate inverse methods seem to be more effective than the non-factorized ones, as it has been pointed out by Benzi and Tuma in [6]. In this work we consider two factorized sparse approximate inverses. The first one is the FSAI preconditioner introduced by Kolotilina and Yeremin [17]. These authors proposed a method to obtain a factorized sparse approximate inverse based on minimizing the Frobenius norm $\|I - G_L L_A\|_F$, where G_L is a sparse approximation to the inverse of the lower triangular Cholesky factor of A, L_A. The second method we will consider corresponds to the AINV preconditioner introduced by Benzi *et al.* [4]. This method builds a factorized approximate inverse for symmetric positive definite matrices by means of an incomplete conjugation process, of the form $G = \bar{Z}\bar{Z}^T \approx A^{-1}$, where $\bar{Z}$ is a triangular matrix and corresponds, when the conjugation process is applied completely, to the inverse of the Cholesky factor of A. Both preconditioners preserve the symmetry and positive definiteness of the original matrix, and therefore they may be used for the Preconditioned Conjugate Gradient method.

Approximate inverses induce splittings of the form $A = G^{-1} - N$, that can be used for polynomial preconditioning. Our goal in this paper is to construct an additive polynomial preconditioner based on a multisplitting obtained from some factorized approximate inverses of A.

The paper is organized as follows. In Section 2 we recall some basic definitions which we refer to later. In Section 3 we describe the proposed additive polynomial preconditioner. In Section 4 the experimental results obtained on a CRAY T3E are presented for matrices arising in a variety of applications, namely, from structural analysis and transport/diffusion problems. Finally, the main conclusions of the work are stated in Section 5.

2. Preliminaries. Let us recall in this section some basic definitions which will be used later.

We say that a matrix B is nonnegative, denoted $B \geq O$, if all its entries are nonnegative. A nonsingular matrix $A \in \mathbb{R}^{n \times n}$ is called M-matrix if all its off-diagonal entries are non-positive and $A^{-1} \geq O$. The splitting $A = M - N$ is *regular* or *weakly* *regular* if $M^{-1} \geq O$ and $N \geq O$, or $M^{-1} \geq O$ and $M^{-1}N \geq O$, respectively. Finally, we recall the definitions of multisplitting and conformable multisplitting as defined in [19] and [11], respectively.

DEFINITION 2.1. *Let A, M_k, N_k and D_k, $k = 1, 2, \ldots, p$, be matrices of size $n \times n$. Then (M_k, N_k, D_k) is said to be a multisplitting of A if*

(i) $A = M_k - N_k$, where M_k is nonsingular, $k = 1, 2, \ldots, p$,

(ii) D_k, $k = 1, 2, \ldots, p$, are nonnegative diagonal matrices such that $\sum_{k=1}^{p} D_k = I$.

In the case that the p splittings (i) are regular (weakly regular), the multisplitting is called regular (weakly regular).

DEFINITION 2.2. *The multisplitting (M_k, N_k, D_k) is said to be* block diagonal conformable *if, for every k:*

(i) M_k is block diagonal, $M_k = \begin{bmatrix} M_{11}^{(k)} & & & \\ & M_{22}^{(k)} & & \\ & & \ddots & \\ & & & M_{qq}^{(k)} \end{bmatrix}$

(ii) D_k is block scalar, that is $D_k = \begin{bmatrix} D_{11}^{(k)} & & & \\ & D_{22}^{(k)} & & \\ & & \ddots & \\ & & & D_{qq}^{(k)} \end{bmatrix}$

where $D_{ii}^{(k)} = d_i^{(k)} I$, and

(iii) the size of $M_{ii}^{(k)}$ coincides with the size of $D_{ii}^{(k)}$, $i = 1, 2, \ldots, q$.

We note that the blocks $M_{ii}^{(k)}$ (and hence $D_{ii}^{(k)}$), $i = 1, 2, \ldots, q$, can have different sizes.

3. Factorized Sparse Approximate Inverse Polynomial Preconditioner. Let A be symmetric positive definite. Consider A partitioned into $p \times p$ blocks, $A = (A_{ij})$, $i, j = 1, 2, \ldots, p$, with diagonal blocks A_{ii} being square and not all necessarily having the same size. Let $A_o = \mathrm{diag}(A_{ii})$ be the matrix obtained from A removing the off-diagonal blocks. Obviously A_o is also spd.

Let $A_o = LL^T$ be the Cholesky factorization of the matrix A_o. We have $A_o = LL^T = \mathrm{diag}(L_{ii})\,\mathrm{diag}(L_{ii}^T)$ and $A_o^{-1} = L^{-T}L^{-1} = \mathrm{diag}(L_{ii}^{-T})\,\mathrm{diag}(L_{ii}^{-1})$, with $i = 1, 2, \ldots, p$. By approximating the inverse of L_{ii} with the matrix Z_{ii}, i.e. $Z_{ii} \approx L_{ii}^{-1}$, we obtain the factorized approximate inverse of the matrix A_o given by

$$A_o^{-1} \approx G = \begin{bmatrix} G_{11} & & & \\ & G_{22} & & \\ & & \ddots & \\ & & & G_{pp} \end{bmatrix}. \tag{3.1}$$

where $G_{ii} = Z_{ii}^T Z_{ii}$ denotes the approximate inverse of the i-th diagonal block of the matrix A.

Since $G^{-1} \approx A_o \approx A$, the matrix G induces the splitting

$$A = G^{-1} - N. \tag{3.2}$$

We note that changing the size or sparsity patten of the diagonal blocks we obtain a different splitting. Considering p splittings of the form (3.2) and suitable weighting matrices D_k, $k = 1, 2, \ldots, p$, we obtain the multisplitting

$$(G_k^{-1}, N_k, D_k), \tag{3.3}$$

where the matrices D_k, $k = 1, 2, \ldots, p$, are chosen in such a way that the resulting multisplitting is block diagonal conformable. The expression for the m-step additive polynomial preconditioner based on this multisplitting is given by

$$\mathcal{M}_m^{-1} = (I + H + \cdots + H^{m-1})W, \tag{3.4}$$

where

$$W = \sum_{k=1}^{p} D_k G_k \quad \text{and} \quad H = \sum_{k=1}^{p} D_k G_k N_k.$$

It can be easily checked that

$$H = I - WA \tag{3.5}$$

The following results from [11] give sufficient conditions for $\mathcal{M}_m^{-1}$ to be symmetric positive definite.

THEOREM 3.1. *Let A be symmetric positive definite. Consider the multisplitting (G_k^{-1}, N_k, D_k). If $D_k G_k^{-1} = G_k^{-1} D_k$ for $k = 1, 2, \ldots, p$, then the preconditioning matrix $\mathcal{M}_m^{-1}$ defined by (3.4) is symmetric.* We note that Theorem 3.1 holds when the multisplitting is block diagonal conformable.

THEOREM 3.2. *Let A be symmetric positive definite. Consider a block diagonal conformable multisplitting (G_k^{-1}, N_k, D_k). Then, the matrix $\mathcal{M}_m^{-1}$ defined by (3.4) is positive definite if, and only if, the matrix $I + H + \cdots + H^{m-1}$ has only positive eigenvalues.*

THEOREM 3.3. *Let A be symmetric positive definite. Consider a block diagonal conformable multisplitting (G_k^{-1}, N_k, D_k). Then*

(i) If m is odd, the matrix $I + H + \cdots + H^{m-1}$ has only positive eigenvalues.

(ii) If m is even, the matrix $I + H + \cdots + H^{m-1}$ has only positive eigenvalues if, and only if, $\rho(H) < 1$. Next theorem, together with Theorem 3.1, provides the validity of the polynomial preconditioner for weak regular multisplittings.

THEOREM 3.4. *Let A be symmetric positive definite. Consider a block diagonal conformable multisplitting (G_k^{-1}, N_k, D_k). If the multisplitting is weakly regular then the matrix $\mathcal{M}_m^{-1}$ is positive definite.*

Proof. Since the splittings $A = G_k^{-1} - N_k$ are weakly regular the matrix $H = \sum_{k=0}^{p} D_k G_k N_k$ is nonnegative and, from the Perron-Frobenius theory, $\rho(H)$ is a nonnegative eigenvalue of H (see for instance [10]). Indeed, since the matrix WA is positive definite, from equation (3.5) we observe that the eigenvalues of H are of the form $1 - \alpha$, with $\alpha > 0$. Thus, $\rho(H) < 1$. In these conditions the proof follows from Theorems 3.2 and 3.3 $\square$

REMARK 3.5. *If A is a symmetric M-matrix then the splitting $A = G^{-1} - N$, where G is an approximate inverse of A obtained by FSAI, is weakly regular (see Lemma 4.1. of [17]). Consider the block diagonal conformable multisplitting (G_k^{-1}, N_k, D_k) defined in (3.3). Assuming that the diagonal blocks of matrix G_k defined in (3.1) have been obtained applying FSAI to the diagonal blocks of A_0, then each diagonal block G_{ii}, $i = 1, \ldots, p$ induces a weakly regular splitting of A_{ii}. Thus $G_{ii} \geq O$, $i = 1, 2, \ldots, p$ and $G_k \geq O$. Besides it is easy to prove that the product $G_k N_k$ is also nonnegative. Hence each splitting of the form (3.2) is weakly regular, and the multisplitting is as well. Thus, from Theorems 3.1 and 3.4 it follows that the preconditioner (3.4) is symmetric positive definite.*

4. Numerical Experiments. We describe in this section the results obtained with an additive polynomial preconditioner based on multisplittings. The multisplitting (G_k^{-1}, N_k, D_k) is obtained as described in section 2, by taking $G_k = G$, $N_k = N$, $k = 1, 2, \ldots, p$ where G is given in (3.1) and $N = G^{-1} - A$. The weighting matrices D_k have been selected as follows

$$D_k = \begin{bmatrix} O & & & & \\ & \ddots & & \text{\Large O} & \\ & & I_{kk} & & \\ & \text{\Large O} & & \ddots & \\ & & & & O \end{bmatrix}.$$

Each processor stores a block of approximately n/p contiguous rows of the matrix A, where n is the matrix dimension, and p is the number of processors. The construction of the matrix G_k^{-1} is done in parallel since each processor computes an approximate inverse of the square diagonal block corresponding to the rows locally stored. Therefore fill-in can occur only inside the diagonal blocks, preserving a good deal of sparsity. Vectors x and b are distributed accordingly to the matrix distribution. The same approach was proposed in [11] but using an incomplete Cholesky factorization of the diagonal blocks. Our approach, since the application of the factorized approximate inverse only requires matrix-vector products, offers a higher degree of parallelism if a vector multiprocessor is used and/or the number of processors computing the diagonal block approximate inverses is smaller than the number of processors applying these factorizations. Furthermore, matrix-vector multiplies are more easily tuned for optimal performance on cache-based architectures than triangular solves.

Table 4.1
Set of matrices tested. n is the number of rows, nnz the total number of nonzero elements.

Matrix	n	nnz	Application
BCSSTK21	3600	26600	Structural analysis
NASA2146	2146	72250	Structural analysis
FDM2	32010	159466	Computational chemistry
DANTE	22065	268945	Boltzmann transport problem
MIXED	28600	142204	Groundwater fluid flow

We present the results of numerical experiments for matrices arising in a variety of applications. BCSSTK21 and NASA2146 are finite element matrices arising from structural analysis problems, and they come from the Harwell-Boeing [13] and Tim Davis's [21] collections, respectively. Matrix FDM2 was provided by Carsten Ullrich. It is an M-matrix corresponding to a finite difference discretization with a highly non-uniform mesh of a Kohn-Sham equation, making the problem relatively hard to solve. Matrix DANTE comes from DANTE, a finite element neutron transport code from Los Alamos National Laboratory. MIXED is a mixed finite element discretization of Richard's equation, which models fluid flow in groundwater. The order n and the number nnz of nonzero elements of these matrices are given in Table 4.1. All the matrices were rescaled by dividing their elements by the absolute value of their largest nonzero entry. The right hand side of each linear system was computed from the solution vector of all ones. Codes were written in Fortran 90 compiled with -O3 optimization, using the MPI message passing library for communications [18]. The results were obtained by running the codes on a Cray T3E. The time, computed using the `mpi_wtime()` function, is given in seconds and it corresponds to the elapsed time.

TABLE 4.2
Experimental results obtained for matrix FDM2.

P	m	AINV					FSAI				
		size	Tcp	Ts	Tt	Iter.	size	Tcp	Ts	Tt	Iter.
		Original Ordering									
1		169127	0.49	12.04	12.53	257	95738	1.82	13.1	14.92	353
2		168513	0.29	5.53	5.82	217	95446	0.91	6.87	7.78	355
4	1	165316	0.18	3.42	3.60	231	94862	0.46	3.72	4.18	360
8		155486	0.10	1.90	2.0	240	93694	0.25	2.16	2.41	369
16		137344	0.07	1.11	1.18	283	91358	0.12	1.35	1.47	384
2		168513	0.29	5.51	5.80	115	95446	0.91	6.92	7.83	192
4	2	165316	0.18	3.24	3.42	118	94862	0.46	3.65	4.11	194
8		155486	0.10	1.74	1.84	121	93694	0.25	2.0	2.25	195
16		137344	0.07	0.95	1.02	140	91358	0.12	1.20	1.32	200
		Reverse Cuthill-McKee									
1							95738	1.81	13.0	14.90	355
2		165042	0.28	5.41	5.69	223	95299	0.91	6.83	7.74	360
4	1	161543	0.17	3.09	3.26	232	94588	0.45	3.76	4.21	371
8		154493	0.10	1.93	2.03	240	93211	0.25	2.17	2.42	379
16		140163	0.07	1.19	1.26	267	90560	0.12	1.33	1.45	387
2		165042	0.28	5.28	5.56	116	95299	0.91	6.79	7.70	192
4	2	161543	0.17	2.98	3.15	119	94588	0.45	3.59	4.04	193
8		154493	0.10	1.79	1.89	125	93211	0.25	1.97	2.22	196
16		140163	0.07	1.05	1.12	136	90560	0.12	1.17	1.29	188

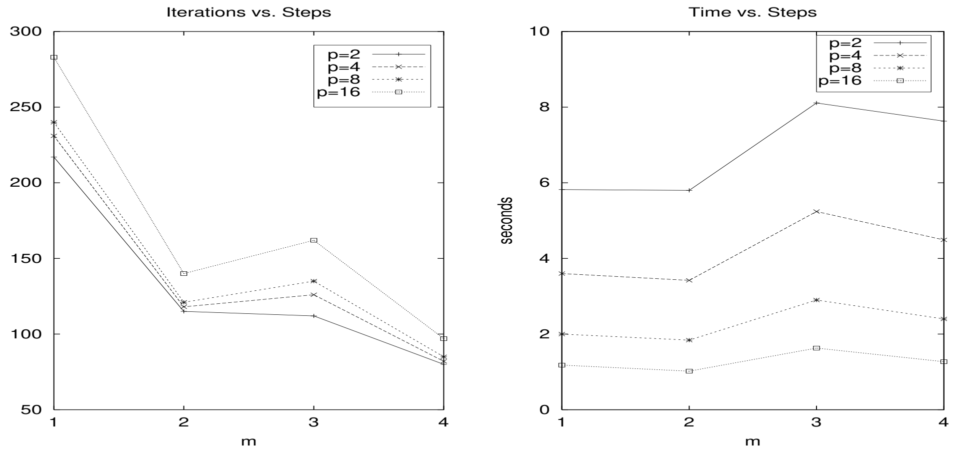

FIG. 4.1. *Number of iterations and time needed to converge for AINV based polynomial preconditioner applied to matrix FDM2. m is the number of steps.*

With the tests performed we study the following three main aspects: first, how effective is the proposed preconditioner as compared with the sequential case. By sequential case we mean the case in which one processor, computing either AINV or FSAI preconditioners to the whole matrix, obtains the system solution with the PCG method. Second, we investigate the optimal number of steps of the polynomial preconditioner for the problems tested. Finally, we study whether it is possible to improve the results by performing different reorderings[3] of the coefficient matrix prior to the construction of the multisplitting. The last question comes from the observa-

[3]We focus on symmetric permutations of the coefficient matrix A, i.e., of the form P^TAP for a

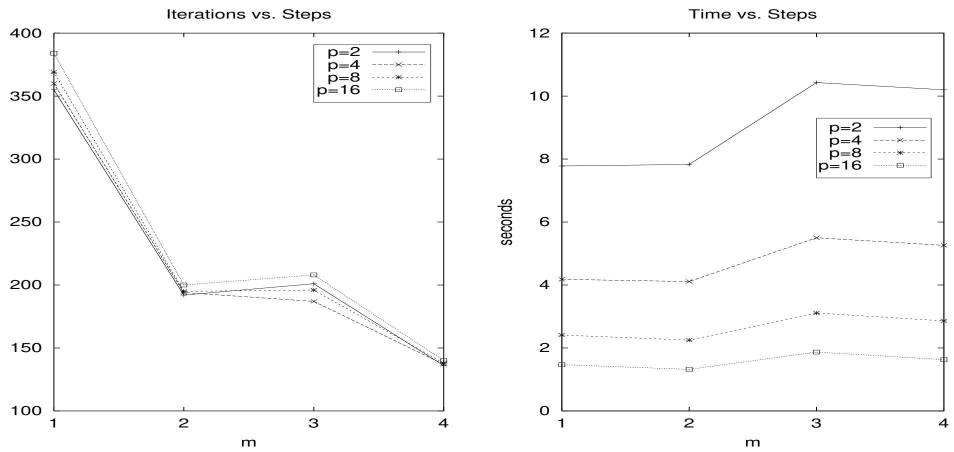

FIG. 4.2. *Number of iterations and time needed to converge for FSAI based polynomial preconditioner applied to matrix FDM2. m is the number of steps.*

TABLE 4.3
Experimental results obtained for matrix DANTE.

		AINV					FSAI				
P	m	size	Tcp	Ts	Tt	Iter.	size	Tcp	Ts	Tt	Iter.
		Original Ordering									
1		87729	1.10	2.26	3.36	54	145505	2.65	2.16	4.81	45
2		41931	0.17	1.92	2.09	83	62707	0.84	1.85	2.69	76
4	1	27324	0.08	1.30	1.38	88	35630	0.42	1.26	1.68	85
8		27317	0.06	0.83	0.89	88	35032	0.21	0.81	1.02	86
16		27303	0.05	0.48	0.53	88	34123	0.11	0.47	0.58	86
2		41931	0.17	215.8	215.97	4964	62707	0.84	35.57	36.41	775
4	2	27324	0.08	116.2	116.28	4249	35630	0.42	52.70	53.12	1925
8		27317	0.06	86.1	86.16	4883	35032	0.21	50.30	50.51	2861
16		27303	0.05	37.73	37.78	3777	34123	0.11	36.14	36.25	3671
		Reverse Cuthill-McKee									
1		156246	0.81	1.94	2.75	41	145505	2.62	1.96	4.58	45
2		147442	0.41	1.26	1.67	52	141759	1.25	1.16	2.41	52
4	1	132891	0.22	0.78	1.0	57	135585	0.59	0.68	1.27	55
8		110110	0.11	0.48	0.59	65	123970	0.29	0.42	0.71	60
16		81855	0.07	0.36	0.43	80	101208	0.14	0.36	0.50	71
2		147442	0.41	101.0	101.41	2182	141759	1.25	1.15	2.40	27
4	2	132891	0.22	56.5	56.72	2292	135585	0.59	0.66	1.25	28
8		110110	0.11	28.5	28.61	2175	123970	0.29	0.39	0.68	31
16		81855	0.07	18.4	18.47	2425	101208	0.14	1.36	1.5	164

tion that the polynomial preconditioner may work better for matrices with elements located around the main diagonal (for instance banded matrices), since our preconditioner consists in computing an approximate inverse to diagonal blocks of A. In this sense, the use of Cuthill-McKee type orderings could be interesting. On the other hand, in [7] the authors show that Multiple Minimum Degree (MMD) and Nested Dissection orderings are quite good for AINV. Thus, the MMD ordering has been also considered. However MMD ordering may not be in general convenient for our approach. Since MMD scatters the matrix elements, when the matrix is distributed

permutation matrix P.

Table 4.4
Experimental results obtained for matrix MIXED.

P	m	AINV size	AINV Tcp	AINV Ts	AINV Tt	AINV Iter.	FSAI size	FSAI Tcp	FSAI Ts	FSAI Tt	FSAI Iter.
		Original Ordering									
1		201765	0.46	11.6	12.06	255	85402	1.67	14.98	16.65	448
2		200296	0.25	6.20	6.45	266	85162	0.83	7.9	8.73	456
4	1	197387	0.15	3.37	3.52	272	84682	0.42	4.29	4.71	460
8		191580	0.09	1.91	2.0	285	83722	0.21	2.41	2.62	465
16		179883	0.07	1.2	1.27	310	81802	0.11	1.47	1.58	475
2		200296	0.25	6.22	6.47	141	85162	0.83	7.76	8.59	242
4	2	197387	0.15	3.31	3.46	144	84682	0.42	4.09	4.51	242
8		191580	0.09	1.83	1.91	151	83722	0.21	2.26	2.47	244
16		179883	0.07	1.1	1.17	163	81802	0.11	1.3	1.41	249
		Reverse Cuthill-McKee									
1		229348	0.49	11.52	12.01	247	85402	1.66	15.16	16.82	458
2		227115	0.26	6.16	6.42	254	85244	0.83	8.1	8.93	474
4	1	223278	0.15	3.49	3.64	267	84966	0.42	4.61	5.03	481
8		216070	0.09	1.92	2.01	282	84452	0.21	2.46	2.67	486
16		202011	0.06	1.2	1.26	314	83442	0.1	1.49	1.59	495
2		227115	0.26	5.94	6.20	130	85244	0.83	7.84	8.67	249
4	2	223278	0.15	3.25	3.40	135	84966	0.42	4.33	4.75	250
8		216070	0.09	1.76	1.85	143	84452	0.21	2.25	2.46	252
16		202011	0.06	1.06	1.12	158	83442	0.1	1.3	1.4	257

Table 4.5
Experimental results obtained for matrix BCSSTK21.

P	m	AINV size	AINV Tcp	AINV Ts	AINV Tt	AINV Iter.	FSAI size	FSAI Tcp	FSAI Ts	FSAI Tt	FSAI Iter.
		Original Ordering									
1		56069	0.24	10.4	10.64	1066	15100	0.26	0.85	1.11	163
2		54408	0.13	4.53	4.66	888	14900	0.13	0.54	0.67	178
4	1	35030	0.07	0.75	0.82	301	14479	0.07	0.40	0.47	206
8		26019	0.05	0.36	0.41	231	13630	0.04	0.30	0.34	219
16		17615	0.04	0.27	0.31	237	11959	0.02	0.25	0.27	246
		Reverse Cuthill-McKee									
1		48493	0.17	3.14	20.4	352	15100	0.24	0.86	1.10	163
2		48483	0.12	1.79	1.91	351	15100	0.13	0.47	0.60	167
4	1	43880	0.08	1.09	1.17	390	14717	0.07	0.41	0.48	213
8		36028	0.05	0.71	0.76	433	14038	0.03	0.27	0.30	205
16		24954	0.04	0.36	0.40	317	12776	0.02	0.21	0.23	214
		Multiple Minimum Degree									
1		15789	0.15	1.25	1.40	188	15100	0.27	0.97	1.24	161
2		4588	0.05	1.16	1.21	327	4584	0.09	1.13	1.22	322
4	1	4120	0.04	0.82	0.86	325	4081	0.05	0.82	0.87	322
8		3861	0.04	0.62	0.66	321	3880	0.03	0.62	0.65	320
16		3793	0.04	0.59	0.63	319	3818	0.02	0.60	0.62	318

among the processors by blocks of rows, this ordering does not preserve data locality for matrix-vector products. This fact could have a negative impact not only for our preconditioner, but also in all cases in which a sparse matrix-vector product must be computed in parallel. Therefore, more elaborate strategies for data distribution among the processors should be used [14]. The results presented below show that Reverse Cuthill-McKee (RCM) ordering in general performs better than MMD and original ordering for all the matrices tested. However, for structural analysis matrices MMD was the best ordering when using AINV to compute the block diagonal

TABLE 4.6
Experimental results obtained for matrix NASA2146.

P	m	AINV size	AINV Tcp	AINV Ts	AINV Tt	AINV Iter.	FSAI size	FSAI Tcp	FSAI Ts	FSAI Tt	FSAI Iter.
		Original Ordering									
1		31956	0.25	2.61	2.86	307	37198	0.80	0.72	1.52	83
2		25706	0.13	1.29	1.42	288	36328	0.39	0.41	0.80	91
4	1	19606	0.07	0.90	0.97	372	34591	0.19	0.26	0.46	96
8		13389	0.05	0.38	0.43	257	31128	0.09	0.19	0.28	105
16		9684	0.04	0.31	0.35	262	24182	0.04	0.16	0.20	121
		Reverse Cuthill-McKee									
1		35177	0.35	3.60	3.95	410	37198	0.80	0.71	1.51	82
2		30533	0.22	2.40	2.62	496	35706	0.38	0.42	0.80	92
4	1	26404	0.21	2.09	2.30	689	33775	0.18	0.27	0.45	99
8		15037	0.05	0.46	0.51	291	29411	0.08	0.21	0.29	115
16		11363	0.05	0.35	0.40	272	21325	0.04	0.20	0.24	143
		Multiple Minimum Degree									
1		23264	0.29	2.21	2.50	227	37198	0.97	0.71	1.68	77
2		9618	0.07	0.78	0.85	202	20146	0.20	0.65	0.85	149
4	1	7034	0.05	0.44	0.49	173	14603	0.06	0.42	0.48	157
8		6308	0.04	0.36	0.40	172	12686	0.05	0.34	0.39	156
16		5456	0.04	0.34	0.38	173	11015	0.02	0.33	0.35	161

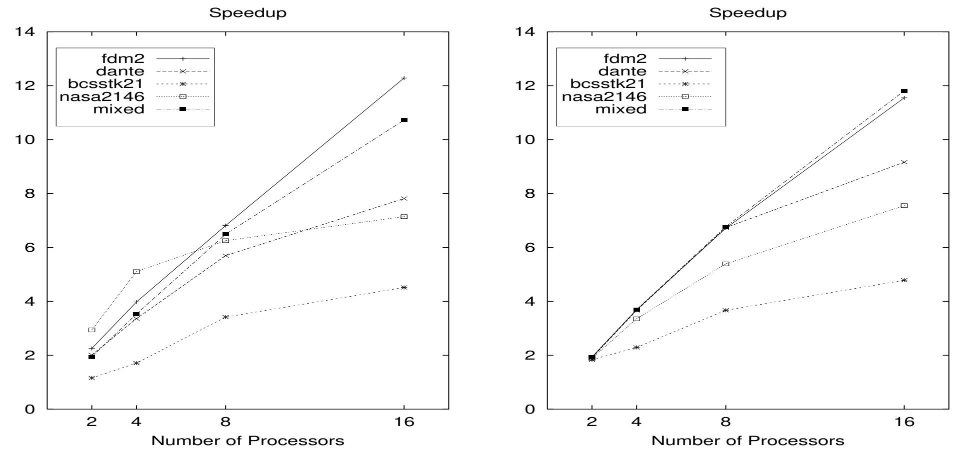

FIG. 4.3. *Speedup obtained for AINV(left) and FSAI(right) based polynomial preconditioners for the problems tested.*

approximate inverse.

In Tables 4.2–4.4 we show the time needed for computing the preconditioner, denoted T_{cp}, the time for obtaining the solution, T_s, and the total time $T_t = T_{cp} + T_s$, for different number of processors P. The number of iterations required to reduce the initial residual vector by at least eight orders of magnitude is denoted by $Iter$. The number of nonzero elements of the approximate inverse factor $size$ and the number of steps of the polynomial preconditioner m are also indicated. There is a table for each matrix, where the results for the AINV and FSAI based polynomial preconditioners are jointly shown. Results with Reverse Cuthill-McKee and Multiple Minimum Degree orderings are also included. For the AINV preconditioner the drop tolerance was set to 0.1, except for BCSSTK21 with MMD ordering. In this case the value 0.2 was used. As was observed in [6], the value 0.1 usually leads to the best results. Conjugate gradient iterations with right preconditioning was used in all cases.

Concerning the optimal number of steps of the polynomial preconditioner we found that for matrices arising from diffusion and transport problems the best results were obtained with the 2-step polynomial preconditioner. Although the largest reduction on the number of iterations is obtained with 4 steps, the number of matrix-vector products performed is not compensated. Observe that the preconditioner works quite well even if the hypothesis of Theorem 3.4 are guaranteed to be verified only in the case of FDM2. In Figures 4.1 and 4.2 we illustrate this behaviour for matrix FDM2. For DANTE, the results of the polynomial preconditioner based on AINV with more than 1 step were poor.

Figure 4.3 shows the speedup for the best parallel execution compared with the sequential case for the matrices tested. It can be seen that up to eight processors the speedups are quite good, even with efficiencies higher than 100% with 2 processors. Since the matrices considered are not very large the speedup with 16 processors is smaller, but still quite good for FDM2 and MIXED.

Concerning the use of orderings, RCM seems to be better than original ordering for transport/diffusion problems. Specially interesting are the results for matrix DANTE. For this matrix only RCM caused a reduction on the number of iterations when passing from 1 to 2 steps. The MMD ordering leads to very sparse factors, resulting in poor convergence results that are not shown. For structural analysis matrices, the AINV based preconditioner was more sensitive to the ordering used than FSAI. For these problems MMD was in general the more effective ordering on reducing the number of iterations. However, the running times were not always reduced as well. The explanation comes from the fact that with this ordering we lose data locality for matrix-vector products. We think that the reduction on the number of iterations caused by MMD could be better exploited with other data distribution strategies.

Comparing the performance of the preconditioner from the point of view of the number of iterations performed, we see that the FSAI based preconditioner is more sensitive to the number of diagonal blocks (processors), specially for structural analysis problems. In contrast, when using AINV for this kind of problems the convergence rate does not decreases significantly, or even increases as it can be observed for matrix NASA2146. For diffusion and transport problems the number of iterations does not increase dramatically when varying the number of blocks. We observe that there are no big differences between AINV and FSAI based preconditioners. For the problems tested we found that using AINV is slightly better for transport/diffusion problems, and FSAI for structural analysis.

Finally we note that the time for computing the preconditioner scales almost perfectly. Although FSAI can be easily parallelized, in the case of AINV, since it is based on a biconjugation process and therefore inherently sequential, its computation in parallel is far from trivial. In this sense, our approach seems to be an interesting option to use this preconditioner on parallel computers. Other existing parallel approaches are for instance [8], where is presented a two-level version based on the Schur complement of the coefficient matrix, and [9] where the preconditioner is sequentially computed and then distributed among processors.

5. Conclusions. We have presented a parallel polynomial preconditioner constructed using certain multisplittings of the coefficient matrix A. The multisplittings are obtained from the splittings induced by factorized approximate inverses applied to diagonal blocks of A. Some theoretical results concerning the properties of the polynomial preconditioner under certain conditions have been given, and numerical experiments using two popular factorized approximate inverses have been presented. Specifically, the AINV and FSAI preconditioners were used to build the polynomial preconditioner. The results show that the proposed preconditioner could be a good choice for solving in parallel problems arising from structural analysis and transport/diffusion problems. For the problems tested, we found that AINV based polynomial preconditioner is better for transport/diffusion problems, while FSAI based preconditioner performs better for structural analysis. In any case, there were no big differences among them. It is interesting to note that the degradation in the convergence rate for increasing number of blocks is very small. Regarding AINV, it is known that its computation is inherently sequential, making difficult its use on parallel environments. In this way, the preconditioner presented can help to avoid this difficulty. The influence of RCM and MMD orderings has been evaluated, showing that RCM is in general more convenient for the preconditioner proposed. In this sense, we note that other orderings producing dense diagonal blocks, as TPABLO [12], could be beneficial for the proposed preconditioner. Finally, introducing overlap among the blocks is a natural extension to be investigated. We note that the one-step polynomial preconditioner matches with an additive Schwarz type method used as preconditioner. In [3] the authors give sufficient conditions for such a preconditioner be symmetric positive definite.

Acknowledgments: We want to thank Michele Benzi and Miroslav Tuma for providing us their research code for computing the AINV and FSAI approximate inverses.

REFERENCES

[1] L. Adams, M-step Preconditioned Conjugate Gradient Methods, *SIAM Journal on Scientific and Statistical Computing*, 6, 452–462, 1985.

[2] L. Adams and E. Ong, Additive Polynomial Preconditioners for Parallel Computers. *Parallel Computing*, 9, 333–345, 1988/89.

[3] M. Benzi, A. Frommer, R. Nabben and D. Szyld, Algebraic theory of multiplicative schwarz methods, *to appear in Numerische Mathematik*.

[4] M. Benzi, C.D. Meyer and M. Tuma, A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method, *SIAM Journal on Scientific Computing*, 17(5), 1135–1149, 1996.

[5] M. Benzi, R. Kohuia and M. Tuma, An assessment of some preconditioning techniques in shell problems, *Commun. Numer. Meth. Engng.*, 14, 897–906, 1998.

[6] M. Benzi and M. Tuma, A Comparative Study of Sparse Approximate Inverse Preconditioners, *Applied Numerical Mathematics* 30(2-3), 305–340, 1999.

[7] M. Benzi and M. Tuma, Orderings for factorized sparse approximate inverse preconditioners, *SIAM J. Scientific Computing* 21(5), 1851–1868, 2000.

[8] M. Benzi, J. Marín and M. Tuma, Parallel preconditioning with Factorized Sparse Approximate Inverses, *in Iterative Methods in Scientific Computation IV, D. R. Kincaid and A. C. Elster, eds., IMACS Series in Computational and Applied Mathematics*, Vol. 5, IMACS, New Brunswick, NJ (1999), pp. 167–178.

[9] L. Bergamaschi, G. Pini and F. Sartoretto, Approximate inverse preconditioning in the parallel solution of sparse eigenproblems, *Numer. Linear Algebra Appl.*, 7, 99–116, 2000.

[10] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences,* SIAM, Philadelphia, 1994.

[11] R. Bru, C. Corral, A. Martínez and J. Mas, Multisplitting Preconditioners based on Incomplete Choleski Factorizations, *SIAM Journal on Matrix Analysis and Applications*, 16(4), 1210–1222, 1995.

[12] Hwajeong Choi and Daniel B. Szyld, Application of threshold partitioning of sparse matrices to Markov chains, *In proceedings of the IEEE International Computer Performance and Dependability Symposium, IPDS'96, Urbana-Champaign, Illinois, September 4–6, 1996*, 158–165, IEEE Computer Society Press, Los Alamitos, California, 1996.

[13] I. S. Duff, R. G. Grimes and J. G. Lewis, Sparse matrix test problems, *ACM Transactions on*

Mathematical Software, 15, 1–14, 1989.

[14] B. Hendrickson, R. Leland and S. Plimpton, An efficient parallel algorithm for matrix-vector multiplication, *Int. Journal High Speed Comput.*, 7, 73–88, 1995.

[15] C. Huang and D. O'Leary, A Krylov Multisplitting Algorithm for Solving Linear Systems of Equations, *Linear Algebra and its Applications*, 194, 9–29, 1993.

[16] O. Johnson, C. Micchelli and G. Paul, Polynomial Preconditioners for Conjugate Gradient Calculations, *SIAM Journal on Numerical Analysis*, 20, 362–376, 1983.

[17] L. Yu. Kolotilina and A. Yu. Yeremin, Factorized Sparse Approximate Inverse Preconditioning I. Theory, *SIAM Journal on Matrix Analysis and Applications*, 14(1), 45–58, 1993.

[18] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard*, Technical report, 1994.

[19] D. O'Leary and R. White, Multi-Splittings of Matrices and Parallel Solution of Linear Equations, *SIAM Journal on Algebraic and Discrete Methods*, 6(4), 630–640, 1985.

[20] J. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems,* Plenum Press, New York, 1988.

[21] T. Davis, **http://www.cise.ufl.edu/ davis/sparse/,** Sparse matrix collection, *NA Digest*, 94(42), 1994.