Research Article

Michele Benzi* and Bora Uçar

Preconditioning Techniques Based on the Birkhoff–von Neumann Decomposition

DOI: 10.1515/cmam-2016-0040
Received October 26, 2016; accepted November 14, 2016

Abstract: We introduce a class of preconditioners for general sparse matrices based on the Birkhoff–von Neumann decomposition of doubly stochastic matrices. These preconditioners are aimed primarily at solving challenging linear systems with highly unstructured and indefinite coefficient matrices. We present some theoretical results and numerical experiments on linear systems from a variety of applications.

Keywords: Preconditioning, Parallel Computing, Doubly Stochastic Matrix, Bipartite Graphs, Birkhoff–von Neumann Decomposition

MSC 2010: 65F35, 65F08, 65F50, 65Y05, 15B51, 05C70

1 Introduction

We consider the solution of linear systems $Ax = b$ where $A = [a_{ij}] \in \mathbb{R}^{n\times n}$, b is a given vector and x is the unknown vector. Our aim is to develop and investigate preconditioners for Krylov subspace methods for solving such linear systems, where A is highly unstructured and indefinite.

For a given matrix A, we first preprocess it to get a doubly stochastic matrix (whose row and column sums are one). Then using this doubly stochastic matrix, we select some fraction of some of the nonzeros of A to be included in the preconditioner. Our main tools are the well-known Birkhoff–von Neumann (BvN) decomposition (this will be discussed in Section 2 for completeness) and a splitting of the input matrix in the form $A = M - N$ based on its BvN decomposition. When such a splitting is defined, M^{-1} or solvers for $My = z$ are required. We discuss sufficient conditions when such a splitting is convergent and discuss specialized solvers for $My = z$ when these conditions are met. We discuss how to build preconditioners meeting the sufficiency conditions. In case the preconditioners become restrictive in practice, their LU decomposition can be used. Our motivation is that the preconditioner M^{-1} can be applied to vectors via a number of highly concurrent steps, where the number of steps is controlled by the user. Therefore, the preconditioners (or the splittings) can be advantageous for use in many-core computing systems. In the context of splittings, the application of N to vectors also enjoys the same property. These motivations are shared by recent work on ILU preconditioners, where their fine-grained computation [9] and approximate application [2] are investigated for GPU-like systems.

The paper is organized as follows. We first give necessary background (Section 2) on doubly stochastic matrices and the BvN decomposition. We then develop splittings for doubly stochastic matrices (Section 3), where we analyze convergence properties and discuss algorithms to construct the preconditioners. Later in the same section, we discuss how the preconditioners can be used for arbitrary matrices by some preprocessing. Here our approach results in a generalization of the Birkhoff–von Neumann decomposition for matrices with positive and negative entries where the sum of the absolute values of the entries in any given row or

***Corresponding author: Michele Benzi:** Department of Mathematics and Computer Science, Emory University, Atlanta, USA, e-mail: benzi@mathcs.emory.edu
Bora Uçar: LIP, UMR5668 (CNRS – ENS Lyon – UCBL – Université de Lyon – INRIA), Lyon, France, e-mail: bora.ucar@ens-lyon.fr

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

column is one. This generalization could be of interest in other areas. Then, we give experimental results (Section 4) with nonnegative and also arbitrary matrices, and then conclude the paper.

2 Background and Definitions

Here we define several properties of matrices: irreducible, full indecomposable, and doubly stochastic matrices. An $n \times n$ matrix A is reducible if there exists a permutation matrix P such that

$$PAP^T = \begin{bmatrix} A_{1,1} & A_{1,2} \\ O & A_{2,2} \end{bmatrix},$$

where $A_{1,1}$ is an $r \times r$ submatrix, $A_{2,2}$ is an $(n-r) \times (n-r)$ submatrix, and $1 \leq r < n$. If such a permutation matrix does not exist, then A is irreducible [25, Chapter 1]. When A is reducible, either $A_{1,1}$ or $A_{2,2}$ can be reducible as well, and we can recursively identify their diagonal blocks, until all diagonal blocks are irreducible. That is, we can obtain

$$PAP^T = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,s} \\ 0 & A_{2,2} & \cdots & A_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_{s,s} \end{bmatrix},$$

where each $A_{i,i}$ is square and irreducible. This block upper triangular form, with square irreducible diagonal blocks is called Frobenius normal form [19, p. 532].

An $n \times n$ matrix A is fully indecomposable if there exists a permutation matrix Q such that AQ has a zero-free diagonal and is irreducible [7, Chapters 3 and 4]. If A is not fully indecomposable, but nonsingular, it can be permuted into the block upper triangular form

$$PAQ^T = \begin{bmatrix} A_{1,1} & A_{1,2} \\ O & A_{2,2} \end{bmatrix},$$

where each $A_{i,i}$ is fully indecomposable or can be further permuted into the block upper triangular form. If the coefficient matrix of a linear system is not fully indecomposable, the block upper triangular form should be obtained, and only the small systems with the diagonal blocks should be factored for simplicity and efficiency [12, Chapter 6]. We therefore assume without loss of generality that matrix A is fully indecomposable.

An $n \times n$ matrix A is doubly stochastic if $a_{ij} \geq 0$ for all i, j and $Ae = A^T e = e$, where e is the vector of all ones. This means that the row sums and column sums are equal to one. If these sums are less than one, then the matrix A is doubly substochastic. A doubly stochastic matrix is fully indecomposable or is block diagonal where each block is fully indecomposable. By Birkhoff's theorem [4], there exist coefficients $\alpha_1, \alpha_2, \ldots, \alpha_k \in (0, 1)$ with $\sum_{i=1}^{k} \alpha_i = 1$, and permutation matrices $P_1, P_2, \ldots, P_k$ such that

$$A = \alpha_1 P_1 + \alpha_2 P_2 + \cdots + \alpha_k P_k. \tag{1}$$

Such a representation of A as a convex combination of permutation matrices is known as a *Birkhoff–von Neumann decomposition* (BvN); in general, it is not unique. The *Marcus–Ree theorem* states that there are BvN decompositions with $k \leq n^2 - 2n + 2$ for dense matrices; Brualdi and Gibson [6] and Brualdi [5] show that for a fully indecomposable sparse matrix with τ nonzeros, we have BvN decompositions with $k \leq \tau - 2n + 2$.

An $n \times n$ nonnegative, fully indecomposable matrix A can be uniquely scaled with two positive diagonal matrices R and C such that RAC is doubly stochastic [24].

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

3 Splittings of Doubly Stochastic Matrices

3.1 Definition and Properties

Let $b \in \mathbb{R}^n$ be given and consider solving the linear system $Ax = b$ where A is doubly stochastic. Hereafter we assume that A is invertible. After finding a representation of A in the form (1), pick an integer r between 1 and $k - 1$ and split A as

$$A = M - N, \tag{2}$$

where

$$M = \alpha_1 P_1 + \cdots + \alpha_r P_r, \quad N = -\alpha_{r+1} P_{r+1} - \cdots - \alpha_k P_k. \tag{3}$$

Note that M and $-N$ are doubly substochastic matrices.

Definition 1. A splitting of the form (2) with M and N given by (3) is said to be a *doubly substochastic splitting*.

Definition 2. A doubly substochastic splitting $A = M - N$ of a doubly stochastic matrix A is said to be *standard* if M is invertible. We will call such a splitting an *SDS splitting*.

In general, it is not easy to guarantee that a given doubly substochastic splitting is standard, except for some trivial situation such as the case $r = 1$, in which case M is always invertible. We also have a characterization for invertible M when $r = 2$.

Theorem 1. *Let $M = \alpha_1 P_1 + \alpha_2 P_2$. Then, M is invertible if* (i) $\alpha_1 \neq \alpha_2$, *or* (ii) $\alpha_1 = \alpha_2$ *and all the fully indecomposable blocks of M have an odd number of rows (and columns). If any such block is of even order, M is singular.*

Proof. We investigate the two cases separately.

Case (i): Without loss of generality assume that $\alpha_1 > \alpha_2$. We have

$$M = \alpha_1 P_1 + \alpha_2 P_2 = P_1(\alpha_1 I + \alpha_2 P_1^T P_2).$$

The matrix $\alpha_1 I + \alpha_2 P_1^T P_2$ is nonsingular. Indeed, its eigenvalues are of the form $\alpha_1 + \alpha_2 \lambda_j$, where λ_j is the generic eigenvalue of the (orthogonal, doubly stochastic) matrix $P_1^T P_2$, and since $|\lambda_j| = 1$ for all j and $\alpha_1 > \alpha_2$, it follows that $\alpha_1 + \alpha_2 \lambda_j \neq 0$ for all j. Thus, M is invertible.

Case (ii): This is a consequence of the Perron–Frobenius theorem. To see this, observe that we need to show that under the stated conditions the sum $P_1 + P_2 = P_1(I + P_1^T P_2)$ is invertible, i.e., $\lambda = -1$ cannot be an eigenvalue of $P_1^T P_2$. Since both P_1 and P_2 are permutation matrices, $P_1^T P_2$ is also a permutation matrix and the Frobenius normal form $T = \Pi(P_1^T P_2)\Pi^T$ of $P_1^T P_2$, i.e., the block triangular matrix

$$T = \begin{bmatrix} T_{1,1} & T_{1,2} & \cdots & T_{1,s} \\ 0 & T_{2,2} & \cdots & T_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & T_{s,s} \end{bmatrix},$$

has $T_{i,j} = 0$ for $i \neq j$. The eigenvalues of $P_1^T P_2$ are just the eigenvalues of the diagonal blocks $T_{i,i}$ of T. Note that there may be only one such block, corresponding to the case where $P_1^T P_2$ is irreducible. Each diagonal block $T_{i,i}$ is also a permutation matrix. Thus, each $T_{i,i}$ is doubly stochastic, orthogonal, and irreducible. Any matrix of this kind corresponds to a cyclic permutation and has its eigenvalues on the unit circle. If a block has size $r > 1$, its eigenvalues are the pth roots of unity, ε^h, $h = 0, 1, \ldots, p - 1$, with $\varepsilon = e^{2\pi i/p}$, e.g., by the Perron–Frobenius theorem (see [17, p. 53]). But $\lambda = -1$ is a pth root of unity if and only if p is even. Since M is a scalar multiple of $P_1 + P_2 = P_1(I + P_1^T P_2)$, this concludes the proof. □

Note that the fully indecomposable blocks of M mentioned in the theorem are just connected components of its bipartite graph.

It is possible to generalize the condition (i) in the previous theorem as follows.

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

Theorem 2. *A sufficient condition for $M = \sum_{i=1}^{r} \alpha_i P_i$ to be invertible is that one of the α_i with $1 \le i \le r$ be greater than the sum of the remaining ones.*

Proof. Indeed, assuming (without loss of generality) that $\alpha_1 > \alpha_2 + \cdots + \alpha_r$, we have

$$\begin{aligned} M &= \alpha_1 P_1 + \alpha_2 P_2 + \cdots + \alpha_r P_r \\ &= P_1(\alpha_1 I + \alpha_2 P_1^T P_2 + \cdots + \alpha_r P_1^T P_r). \end{aligned}$$

This matrix is invertible if and only if the matrix $\alpha_1 I + \sum_{i=2}^{r} \alpha_i P_1^T P_i$ is invertible. Observing that the eigenvalues λ_j of $\sum_{i=2}^{r} \alpha_i P_1^T P_i$ satisfy

$$|\lambda_j| \le \|\alpha_2 P_1^T P_2 + \cdots + \alpha_r P_1^T P_r\|_2 \le \sum_{i=2}^{r} \alpha_i < \alpha_1$$

(where we have used the triangle inequality and the fact that the 2-norm of an orthogonal matrix is one), it follows that M is invertible. □

Again, this condition is only a sufficient one. It is rather restrictive in practice.

3.2 Convergence Conditions

Let $A = M - N$ be an SDS splitting of A and consider the stationary iterative scheme

$$x^{k+1} = Hx^k + c, \quad H = M^{-1}N, \quad c = M^{-1}b, \tag{4}$$

where $k = 0, 1, \dots$ and x^0 is arbitrary. As is well known, the scheme (4) converges to the solution of $Ax = b$ for any x^0 if and only if $\rho(H) < 1$. Hence, we are interested in conditions that guarantee that the spectral radius of the iteration matrix

$$H = M^{-1}N = -(\alpha_1 P_1 + \cdots + \alpha_r P_r)^{-1}(\alpha_{r+1} P_{r+1} + \cdots + \alpha_k P_k)$$

is strictly less than one. In general, this problem appears to be difficult. We have a necessary condition (Theorem 3), and a sufficient condition (Theorem 4) which is simple but restrictive.

Theorem 3. *For the splitting $A = M - N$ with $M = \sum_{i=1}^{r} \alpha_i P_i$ and $N = -\sum_{i=r+1}^{k} \alpha_i P_i$ to be convergent, it must hold that $\sum_{i=1}^{r} \alpha_i > \sum_{i=r+1}^{k} \alpha_i$.*

Proof. First, observe that since $P_i e = e$ for all i, both M and N have constant row sums:

$$Me = \sum_{i=1}^{r} \alpha_i P_i e = \beta e, \quad Ne = -\sum_{i=1}^{r} \alpha_i P_i e = (\beta - 1)e,$$

with $\beta := \sum_{i=1}^{r} \alpha_i \in (0, 1)$. Next, observe that $M^{-1}Ne = \lambda e$ is equivalent to $Ne = \lambda Me$ or, since we can assume that $\lambda \ne 0$, to $Me = \frac{1}{\lambda} Ne$. Substituting βe for Me and $(\beta - 1)e$ for Ne, we find

$$\beta e = \frac{1}{\lambda}(\beta - 1)e \quad \text{or} \quad \lambda = \frac{\beta - 1}{\beta}.$$

Hence, $\frac{\beta-1}{\beta}$ is an eigenvalue of $H = M^{-1}N$ corresponding to the eigenvector e. Since $|\lambda| = \frac{1-\beta}{\beta}$, we conclude that $\rho(H) \ge 1$ for $\beta \in (0, \frac{1}{2}]$. This concludes the proof. □

Theorem 4. *Suppose that one of the α_i appearing in M is greater than the sum of all the other α_i. Then $\rho(M^{-1}N) < 1$ and the stationary iterative method (4) converges for all x^0 to the unique solution of $Ax = b$.*

Proof. Assuming (without loss of generality) that

$$\alpha_1 > \alpha_2 + \cdots + \alpha_k \tag{5}$$

(which, incidentally, ensures that the matrix M is invertible), we show below that

$$\|H\|_2 = \|(\alpha_1 P_1 + \cdots + \alpha_r P_r)^{-1}(\alpha_{r+1} P_{r+1} + \cdots + \alpha_k P_k)\|_2 < 1. \tag{6}$$

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

This, together with the fact that $\rho(H) \le \|H\|_2$, ensures convergence. To prove (6) we start by observing that

$$M = \alpha_1 P_1 + \cdots + \alpha_r P_r = \alpha_1 P_1 \Big(I + \frac{\alpha_2}{\alpha_1} Q_2 + \cdots + \frac{\alpha_r}{\alpha_1} Q_r \Big),$$

where $Q_i = P_1^T P_i$. Thus, we have

$$M^{-1} = (\alpha_1 P_1 + \cdots + \alpha_r P_r)^{-1} = \frac{1}{\alpha_1}(I - G)^{-1} P_1^T,$$

where we have defined

$$G = -\frac{1}{\alpha_1} \sum_{i=2}^{r} \alpha_i Q_i.$$

Next, we observe that $\|G\|_2 < 1$, since

$$\|G\|_2 \le \frac{1}{\alpha_1} \sum_{i=2}^{r} \alpha_i \|Q_i\|_2 = \frac{\alpha_2 + \cdots + \alpha_r}{\alpha_1} < 1$$

as a consequence of (5). Hence, the expansion

$$(I - G)^{-1} = \sum_{\ell=0}^{\infty} G^\ell$$

is convergent, and moreover

$$\|(I - G)^{-1} P_1^T\|_2 = \|(I - G)^{-1}\|_2 \le \frac{1}{1 - \|G\|_2} \le \frac{1}{1 - \big(\frac{\alpha_2}{\alpha_1} + \cdots + \frac{\alpha_r}{\alpha_1}\big)}.$$

The last inequality follows from the fact that $\|G\|_2 \le \sum_{i=2}^{r} \frac{\alpha_i}{\alpha_1}$, as can be seen using the triangle inequality and the fact that $\|Q_i\|_2 = 1$ for all $i = 2, \ldots, r$.

Hence, we have

$$\|M^{-1}N\|_2 \le \frac{1}{\alpha_1} \frac{1}{1 - \big(\frac{\alpha_2}{\alpha_1} + \cdots + \frac{\alpha_r}{\alpha_1}\big)} \|\alpha_{r+1} P_{r+1} + \cdots + \alpha_k P_k\|_2.$$

Using once again the triangle inequality (applied to the last term on the right of the foregoing expression), we obtain

$$\|M^{-1}N\|_2 \le \frac{\alpha_{r+1} + \cdots + \alpha_k}{\alpha_1 - (\alpha_2 + \cdots + \alpha_r)}.$$

Using condition (5), we immediately obtain

$$\frac{\alpha_{r+1} + \cdots + \alpha_k}{\alpha_1 - (\alpha_2 + \cdots + \alpha_r)} < 1,$$

therefore $\|M^{-1}N\|_2 < 1$ and the iterative scheme (4) is convergent. □

Note that if condition (5) is satisfied, then the value of r in (3) can be chosen arbitrarily; that is, the splitting will converge for any choice of r between 1 and k. In particular, the splitting $A = M - N$ with $M = \alpha_1 P_1$ and $N = -\sum_{i=2}^{k} \alpha_i P_i$ is convergent. It is an open question whether adding more terms to M (that is, using $M = \sum_{i=1}^{r} \alpha_i P_i$ with $r > 1$) will result in a smaller spectral radius of H (and thus in faster asymptotic convergence); notice that adding terms to M will make application of the preconditioner more expensive.

Note that condition (5) is very restrictive. It implies that $\alpha_1 > \frac{1}{2}$, a very strong restriction. It is clear that given a doubly substochastic matrix, in general it will have no representation (1) with $\alpha_1 > \frac{1}{2}$. On the other hand, it is easy to find examples of splittings of doubly substochastic matrices for which $\alpha_1 = \frac{1}{2}$ and the splitting (3) with $r = 1$ is not convergent. Also, we have found examples with $k = 3$, $r = 2$ and $\alpha_1 + \alpha_2 > \alpha_3$ where the splitting did not converge. It is an open problem to identify other, less restrictive conditions on the α_i (with $1 \le i \le r$) that will ensure convergence, where the pattern of the permutations could also be used.

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

3.3 Solving Linear Systems with *M*

The stationary iterations (4) for solving $Ax = b$ or Krylov subspace methods using M as a preconditioner require solving linear systems of the form $Mz = y$. Assume that $M = \alpha_1 P_1 + \alpha_2 P_2$ and $\alpha_1 > \alpha_2$. The stationary iterations $z^{k+1} = \frac{1}{\alpha_1} P_1^T (y - \alpha_2 P_2 z^k)$ are convergent for any starting point, with the rate of $\frac{\alpha_2}{\alpha_1}$. Therefore, if $M = \alpha_1 P_1 + \alpha_2 P_2$ and M is as described in Theorem 1 (i), then we use the above iterations to apply M^{-1} (that is, solve linear systems with M). If M is as described in Theorem 4, then we can still solve $Mz = y$ by stationary iterations, where we use the splitting $M = \alpha_1 P_1 - \sum_{r=2}^{k} \alpha_r P_r$. We note that application of $\sum_{r=2}^{k} \alpha_r P_r$ to a vector y, that is, $z = (\sum_{r=2}^{k} \alpha_r P_r)y$, can be effectively computed in $k - 1$ steps, where at each step, we perform $z \leftarrow z + (\alpha_r P_r)y$. This last operation takes n input entries, scales them and adds them to n different positions in z. As there are no read or write dependencies between these n operations, each step is trivially parallel; especially in shared memory environments, the only parallelization overhead is the creation of threads. Either input can be read in order, or the output can be written in order (by using the inverse of the permutation P_r).

As said before, splitting is guaranteed to work when $\alpha_1 > \frac{1}{2}$. Our experience showed that it does not work in general when $\alpha_1 < \frac{1}{2}$. Therefore, we suggest using M as a preconditioner in Krylov subspace methods. There are two ways to do that. The first is to solve linear systems with M with a direct method by first factorizing M. Considering that the number of nonzeros in M would be much less than that of A, factorization of M could be much more efficient than factorization of A. The second alternative, which we elaborate in Section 3.4, is to build M in such a way that one of the coefficients is larger than the sum of the rest.

3.4 Algorithms for Constructing the Preconditioners

It is desirable to have a small number k in the Birkhoff–von Neumann decomposition while designing the preconditioner. This is because of the fact that if we use splitting, then k determines the number of steps in which we compute the matrix vector products. If we do not use all k permutation matrices, having a few with large coefficients should help to design the preconditioner. The problem of obtaining a Birkhoff–von Neumann decomposition with the minimum number k is a difficult one, as pointed out by Brualdi [5] and has been shown to be NP-complete [15]. This last reference also discusses a heuristic which delivers a small number of permutation matrices. We summarize the heuristic in Algorithm 1.

Algorithm 1. A greedy heuristic for obtaining a BvN decomposition.

Input: A, a doubly stochastic matrix;

Output: a BvN decomposition (1) of A with k permutation matrices.

```
(1) $k \leftarrow 0$
(2) while (nnz($A$) > 0)
(3)     $k \leftarrow k + 1$
(4)     $P_k \leftarrow$ the pattern of a bottleneck perfect matching $M$ in $A$
(5)     $\alpha_k \leftarrow \min M_{i,P_k(i)}$
(6)     $A \leftarrow A - \alpha_k P_k$
(7) endwhile
```

As seen in Algorithm 1, the heuristic proceeds step by step, where at each step j, a bottleneck matching is found to determine α_j and P_j. A bottleneck matching can be found using MC64 [13, 14]. In this heuristic, at line 4, M is a bottleneck perfect matching; that is, the minimum weight of an edge in M is the maximum among all minimum elements of perfect matchings. Also, at line 5, α_k is equal to the bottleneck value of the perfect matching M. A nice property of this heuristic is that it delivers α_is in a non-increasing order; that is $\alpha_j \geq \alpha_{j+1}$ for all $1 \leq j < k$. The worst case running time of a step of this heuristic can be bounded by the worst case running time of a bottleneck matching algorithm. For matrices where $\text{nnz}(A) = O(n)$, the best known algorithm is of time complexity $O(n\sqrt{n \log n})$ (see [16]). We direct the reader to [8, p. 185] for other cases.

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

This heuristic could be used to build an M such that it satisfies the sufficiency condition presented in Theorem 4. That is, we can have M with $\alpha_1/\sum_{i=1}^{k}\alpha_i > \frac{1}{2}$, and hence M^{-1} can be applied with splitting iterations. For this, we start by initializing M to $\alpha_1 P_1$. Then, when the α_j where $j \geq 2$ is obtained at line 5, we add $\alpha_j P_j$ to M if α_1 is still larger than the sum of the other α_j included in M. In practice, we iterate the while loop until k is around 10 and collect α_j's as described above as long as

$$\frac{\alpha_1}{\sum_{P_j \in M}\alpha_j} > \frac{1}{1.9} \approx 0.53.$$

3.5 Arbitrary Coefficient Matrices

Here we discuss how to apply the proposed preconditioning technique to all fully indecomposable sparse matrices.

Let $A \geq 0$ be an $n \times n$ fully indecomposable matrix. The first step is to scale the input matrix with two positive diagonal matrices R and C so that RAC is doubly stochastic, or nearly so. For this step, there are different approaches [20, 21, 24]. Next the linear system $Ax = b$ can be solved by solving $RACx' = Rb$ and recovering $x = Cx'$, where we have a doubly stochastic coefficient matrix.

Suppose A is an $n \times n$ fully indecomposable matrix with positive and negative entries. Then, let $B = \text{abs}(A)$ and consider R and C making RBC doubly stochastic, which has a Birkhoff–von Neumann decomposition $RBC = \sum_{i=1}^{k}\alpha_i P_i$. Then, RAC can be expressed as

$$RAC = \sum_{i=1}^{k}\alpha_i Q_i,$$

where $Q_i = [q_{jk}^{(i)}]_{n\times n}$ is obtained from $P_i = [p_{jk}^{(i)}]_{n\times n}$ as follows:

$$q_{jk}^{(i)} = \text{sign}(a_{jk})p_{jk}^{(i)}.$$

That is, we can use a convex combination of a set of signed permutation matrices to express any fully indecomposable matrix A where $\text{abs}(A)$ is doubly stochastic. We can then use the same construct to use r permutation matrices to define M (for splitting or for defining the preconditioner).

We note that the Theorems 2 and 4 remain valid without changes, since the only property we use is the orthogonality of the P_i (not the nonnegativity). Theorem 1, on the other hand, needs some changes. All we can prove in this more general setting is that if $\alpha_1 \neq \pm\alpha_2$, then $M = \alpha_1 P_1 + \alpha_2 P_2$ is nonsingular. We could not find a simple condition for the case $\alpha_1 = \pm\alpha_2$ since we cannot use the Perron–Frobenius theorem to conclude anything about 1 (or −1) being an eigenvalue of $P_1^T P_2$.

4 Experiments

We are interested in testing our preconditioner on challenging linear systems that pose difficulties for standard preconditioned Krylov subspace methods. We conducted experiments with the preconditioned GMRES of Matlab, without restart. We also use an implementation of the flexible GMRES (FGMRES) [23] when needed.

We used a large set of matrices which come from three sources. The first set contains 22 matrices which were used by Benzi et al. [3]. Another six matrices were used by Manguoglu et al. [22]; they have experiments with larger matrices but we chose only those with $n \leq 20000$. These matrices are shown in Table 1. All these matrices, except `slide`, `two-dom`, `watson4a`, and `watson5a`, are available from the UFL Sparse Matrix Collection [10]. To this set, we add all real, square matrices from the UFL Sparse Matrix Collection which contain "chemical" as the keyword. These matrices pose challenges to Krylov subspace methods. There were a total of 70 such matrices; taking the union with the previously described 28 matrices yield 90 matrices in total. Table 1 shows the size and the number of nonzeros of the largest fully indecomposable block (these are the largest square blocks in the Dulmage–Mendelsohn decomposition of the original matrices) of the first two

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

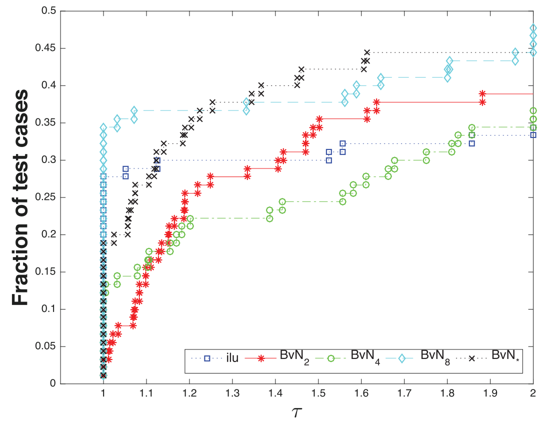
matrix	n_A	nnz$_A$	n	nnz	k
slide	20191	1192535	19140	1191421	900
two-dom	22200	1188152	20880	1186500	938
watson5a	1854	8626	1765	6387	28
watson4a	468	2459	364	1480	134
bp_200	822	3802	40	125	47
gemat11	4929	33108	4578	31425	234
gemat12	4929	33044	4552	31184	260
lns_3937	3937	25407	3558	24002	205
mahindas	1258	7682	589	4744	722
orani678	2529	90158	1830	47823	6150
sherman2	1080	23094	870	19256	334
west0655	655	2808	452	1966	307
west0989	989	3518	720	2604	315
west1505	1505	5414	1099	3988	345
west2021	2021	7310	1500	5453	376
circuit_3	12127	48137	7607	34024	322
bayer09	3083	11767	1210	6001	300
bayer10	13436	71594	10803	62238	4
lhr01	1477	18427	1171	15914	473
lhr02	2954	36875	1171	15914	476
appu	14000	1853104	14000	1853104	1672
raefsky4	19779	1316789	19779	1316789	774
venkat25	62424	1717763	62424	1717763	375
utm5940	5940	83842	5794	83148	291
bundle1	10581	770811	10581	770811	9508
fp	7548	834222	7548	834222	8008
dw8192	8192	41746	8192	41746	155
FEM_3D	17880	430740	17880	430740	329

Table 1. Test matrices, their original size n_A and the size of the largest fully indecomposable block n. The experiments were done using the largest fully indecomposable block (n is the effective size in the following experiments). The last matrix's full name is `FEM_3D_thermal1`.

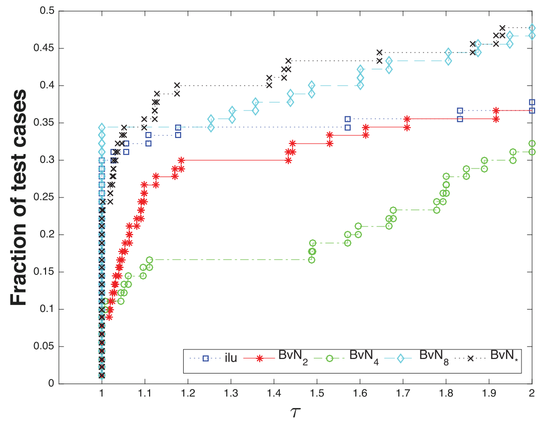
sets of matrices. The experiments are conducted with those largest blocks; from now on a matrix refers to its largest block.

In Sections 4.1 and 4.2, we present two sets of experiments: with nonnegative matrices and with general matrices. Each time, we compare three preconditioners. The first is ILU(0), which forms a basis of comparison. We created ILU(0) preconditioners by first permuting the coefficient matrices to have a maximum product matching in the diagonal (using MC64), as suggested by Benzi et al. [3], and then calling `ilu` of Matlab with the option `nofill`. The second set of preconditioners is obtained by a (generalized) BvN decomposition and taking a few different values of "r" in defining the preconditioner. We use BvN$_r$ to denote these preconditioners, where the first r permutation matrices and the corresponding coefficients are taken from the output of Algorithm 1. Since we cannot say anything special about these preconditioners, we use their LU decomposition to apply the preconditioner. For this purpose, we used MATLAB's LU with four output arguments (which permutes the matrices for numerical stability and fill-in). The third set of preconditioners is obtained by the method proposed in Section 3.4 so that the preconditioners satisfy the sufficiency condition of Theorem 4, and hence we can use the splitting based specialized solver. We use BvN$_*$ to denote these preconditioners. In the experiments with the ILU(0) and BvN$_r$ preconditioners, we used MATLAB's GMRES, whereas with BvN$_*$ we used FGMRES, since we apply BvN$_*$ with the specialized splitting based solver. In all cases, we asked a relative residual of 10^{-6} from (F)GMRES and run them without restart with at most 3000 iterations. We checked the output of (F)GMRES, and marked those with `flag` $\neq 0$ as unsuccessful. For the successful runs, we checked the relative residual $\|Ax - b\|/\|b\|$ for a linear system $Ax = b$, and deemed the runs whose relative residual is larger than 10^{-4} as unsuccessful.

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM



(a) Nonnegative matrices



(b) General matrices

Figure 1. Performance profiles for the number of iterations with ILU(0), BvN$_r$ with $r = 2, 4, 8$, and BvN$_*$.

4.1 Nonnegative Matrices

The first set of experiments is conducted on nonnegative matrices. Let A be a matrix from the data set (Table 1 and another 62 matrices) and $B = \text{abs}(A)$, that is, $b_{ij} = |a_{ij}|$. We scaled B to a doubly stochastic form with the method of Knight and Ruiz [20]. We asked a tolerance of 10^{-8} (so that row and column sums can deviate from 1 by 10^{-8}). We then obtained the Birkhoff–von Neumann decomposition by using Algorithm 1. When the bottleneck value found at a step was smaller than 10^{-10}, we stopped the decomposition process – hence we obtain an "approximate" Birkhoff–von Neumann decomposition of an "approximately" doubly stochastic matrix. This way we obtained $B \approx \alpha_1 P_1 + \alpha_2 P_2 + \cdots + \alpha_k P_k$. Then, let $x^\star$ be a random vector whose entries are from the uniform distribution on the open interval $(0, 1)$, generated using `rand` of Matlab. We then defined $b = Bx^\star$ to be the right-hand side of $Bx = b$.

We report our experiments with GMRES and FGMRES using different permutation matrices on nonnegative matrices in Table 2. The first part of the table gives the number of GMRES iterations with the ILU(0) preconditioner, and with the proposed Birkhoff–von Neumann based preconditioner, using $r = 1, 2, 4, 8, 16, 32$, and 64 permutation matrices and their coefficients. When there were less than r permutation matrices in the BvN decomposition, we used all available permutation matrices. Note that this does not mean that we have an exact inverse, as the BvN decomposition is only approximative. We also report results with FGMRES for the BvN$_*$ preconditioners. For this case, we give the number r of permutation matrices used and the number of FGMRES iterations (under the column "it"). In the second part (the last row of the table), we give the number of successful runs with different preconditioners; here we also give the average number of permutation matrices in BvN$_*$ under the column "r".

Some observations are in order. As seen in Table 2, ILU(0) results in 11 successful runs on the matrices listed in the first part of the table, and 32 successful runs in all 90 matrices. BvN$_r$ preconditioners with differing number of permutation matrices result in at least 53 successful runs in total, where BvN$_*$ obtained the highest number of successful runs, but it has in only a few cases the least number of iterations. In 20 cases in the first part of the table, we see that adding more permutation matrices usually helps in reducing the number of iterations. This is not always the case though. We think that this is due to the preconditioner becoming badly conditioned with the increasing number of permutation matrices; if we use the full BvN decomposition, we will have the same conditioning as in A.

In order to clarify more, we present the performance profiles [11] for ILU(0), BvN$_r$ with $r = 2, 4, 8$, and BvN$_*$ in Figure 1a. A performance profile for a preconditioner shows the fraction of the test cases in which the number of (F)GMRES iterations with the preconditioner is within τ times the smallest number of (F)GMRES iterations observed (with the mentioned set of preconditioners). Therefore, the higher the profile of a preconditioner, the better is its performance. As seen in this figure, BvN$_*$ and BvN$_8$ manifest themselves as the best preconditioners; they are better than others after around $\tau = 1.1$, while BvN$_8$ being always better than others.

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

		BvN$_r$ (iterations)							**BvN$_*$**	
matrix	**ILU(0)**	**1**	**2**	**4**	**8**	**16**	**32**	**64**	***r***	**it.**
slide	–	1102	1158	946	787	567	566	131	10	884
two-dom	1328	–	–	–	2774	2088	1719	1429	–	–
watson5a	–	453	516	610	806	908	908	908	5	621
watson4a	49	146	133	139	124	93	32	5	12	129
bp_200	–	36	26	25	12	7	3	3	8	25
gemat11	176	–	–	2857	2118	1406	664	169	–	–
gemat12	263	–	–	2906	2237	1591	–	145	–	–
lns_3937	349	2067	1286	666	229	56	14	5	10	1138
mahindas	–	303	283	245	173	87	40	21	7	278
orani678	–	371	360	334	303	283	264	221	5	335
sherman2	19	324	238	157	96	51	21	7	9	235
west0655	71	331	275	188	139	–	–	31	9	270
west0989	–	194	167	114	63	35	19	9	8	165
west1505	–	335	254	166	91	49	27	10	9	241
west2021	–	406	323	217	–	47	34	13	8	315
circuit_3	–	–	–	–	–	–	–	1426	–	–
bayer09	124	521	434	291	204	110	28	7	7	393
bayer10	–	–	–	–	–	–	–	–	–	–
lhr01	–	574	350	216	130	86	42	18	7	313
lhr02	–	565	351	219	136	87	45	19	8	314
appu	31	50	50	52	56	62	74	82	7	50
raefsky4	–	–	2728	2735	2444	1559	1739	1230	6	2370
venkat25	–	–	–	–	–	–	–	–	–	–
utm5940	–	2633	2115	–	–	–	153	34	7	2069
bundle1	25	163	150	139	149	174	158	139	5	149
fp	–	–	–	–	–	–	–	–	–	–
dw8192	–	–	–	2398	1516	650	81	8	–	–
FEM_3D	7	37	35	38	72	58	25	6	9	33
		Number of successful runs with all 90 matrices								
	32	58	71	60	57	53	58	61	7	77

Table 2. The number of GMRES iterations with ILU(0) and BvN$_r$ with different number of permutation matrices, and the number of permutation matrices and FGMRES iterations with BvN$_*$. (F)GMRES are run with tolerance 10^{-6}, without restart and with at most 3000 iterations. The symbol "–" flags the cases where (F)GMRES were unsuccessful. *All matrices are nonnegative.*

BvN$_*$ is a little behind ILU(0) at the beginning but then catches up with BvN$_8$ at around 1.2 and finishes as the best alternative.

Although we are mostly concerned with robustness of the proposed preconditioners and their potential for parallel computing, we give a few running time results on a sequential Matlab environment (on a core of an Intel Xeon E5-2695 with 2.30 GHz clock speed). We present the complexity of the preconditioners and the running time of (F)GMRES (total time spent in the iterations) for the cases where ILU(0) resulted in convergence in Table 3. This set of matrices is chosen to give the running time of (F)GMRES with all preconditioners under study. Otherwise, it is not very meaningful to use sophisticated preconditioners when ILU(0) is effective. Furthermore, the applications of ILU(0) and BvN$_r$ require triangular solves, which are efficiently implemented in MATLAB. On the other hand, the application of BvN$_*$ requires permutations and scaling, which should be very efficient in parallel. The running time given in the right side of Table 3 should not be taken at face value. The complexities of BvN$_r$ are given as the ratio $(\mathrm{nnz}(L + U) - n)/\,\mathrm{nnz}(A)$, where L and U are the factors of the preconditioner. In this setting, ILU(0) has always a complexity of 1.0. The complexity of BvN$_*$ is given as $\mathrm{nnz}(M)/\,\mathrm{nnz}(A)$. As seen in this table, the LU-factors of BvN$_r$ with $r \leq 8$ have reasonable number of nonzeros; the complexity of the preconditioners is less than one in all cases except `two-dom`, `lns_3937`, `west0655`, and `appu`. In some preliminary experiments, we have seen large numbers when $r \geq 16$. This was especially important for `appu` where the full LU factorization of A contained $85.67 \cdot \mathrm{nnz}(A)$ nonzeros and M

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

	Complexity of M				Running time				
	BvN$_r$					BvN$_r$			
	2	4	8	BvN$_*$	ILU(0)	2	4	8	BvN$_*$
two-dom	0.03	0.04	1.33	0.08	3219.99	–	–	6108.59	–
watson4a	0.28	0.36	0.44	0.66	0.06	0.24	0.26	0.21	6.99
gemat11	0.23	0.35	0.65	0.61	8.31	–	851.16	458.18	–
gemat12	0.21	0.31	0.58	0.61	10.93	–	825.32	492.63	–
lns_3937	0.29	0.55	1.67	0.63	27.28	131.73	31.42	3.84	519.83
sherman2	0.06	0.10	0.14	0.26	0.02	0.83	0.39	0.16	20.13
west0655	0.42	0.69	1.11	0.62	0.08	0.94	0.46	0.26	16.84
bayer09	0.32	0.52	0.91	0.51	0.41	2.89	1.34	0.70	42.96
appu	0.01	0.02	7.05	0.04	1.59	1.07	1.18	3.13	31.99
bundle1	0.01	0.01	0.02	0.03	0.94	4.70	4.13	4.31	88.82
FEM_3D	0.05	0.08	0.78	0.27	0.36	0.73	0.83	2.23	23.40

Table 3. The complexity of the preconditioners and the running time of the solver. The complexities of BvN$_r$ are given as the ratio $(\text{nnz}(L+U)-n)/\text{nnz}$, where L and U are the factors of the preconditioner (ILU(0) has a complexity of 1.0), and that of BvN$_*$ is given as $\text{nnz}(M)/\text{nnz}(A)$. A sign of "–" in the running time column flags the cases where (F)GMRES were unsuccessful. *All matrices are nonnegative.*

suffered large fill-in. We run the solver with a large error tolerance of 1.0e-1, as this was enough to get the outer FGMRES to converge. The outer FGMRES iterations would change if a lower error tolerance is used in the inner solver, but we do not dwell into this issue here. With the specified parameters, an application of BvN$_*$ required, for the data shown in Table 3, between 128 and 162 iterations.

We now comment on the running time of the preconditioner set up phase. The first step is to apply the scaling method of Knight and Ruiz [20]. The dominant cost in this step is sparse matrix-vector and sparse matrix transpose-vector multiply operations. The second step is to obtain a BvN decomposition, or a partial one, using the algorithms for the bottleneck matching problem from MC64 [13, 14]; these algorithms are highly efficient. The number of sparse matrix-vector and sparse matrix transpose-vector multiply operations (MVP) in the scaling algorithm, the running time of the scaling algorithm with the previous setting, and the running time of the partial BvN decomposition with 16 permutation matrices are shown for the same set of matrices in Table 4. Since an application of BvN$_*$ required between 128 and 162 iterations, and those iterations are no more costly than sparse matrix vector operations, we could relate the running time of FGMRES with BvN$_*$ in Table 3 to the scaling algorithm. For example, for `lns_3937` a matrix vector multiplication should not cost more than 0.27/3732 seconds, while the application of BvN$_*$ required a total of $1138 \cdot 152$ steps of inner solver. In principle the total cost of the inner solver should be around 12.5, but in Table 3, we see 519.83 seconds of total FGMRES time. As seen in Table 4, for this set of matrices the scaling algorithm and the partial BvN decomposition algorithms are fast. There are efficient parallel scaling algorithms [1], and the one that we used [20] can also be efficiently parallelized by tapping into the parallel sparse matrix vector multiplication algorithms.

4.2 General Matrices

In this set of experiments, we used the matrices listed before, while retaining the sign of the nonzero entries. We used the same scaling algorithm as in the previous section and the proposed generalized BvN decomposition (Section 3.5) to construct the preconditioners. As in the previous subsection, we report our experiments with GMRES and FGMRES using different permutation matrices in Table 5. In particular, we report the number of GMRES iterations with the ILU(0) preconditioner, and with the proposed Birkhoff–von Neumann based preconditioner, using $r = 1, 2, 4, 8, 16, 32$, and 64 permutation matrices and their coefficients. We also report results with FGMRES for the BvN$_*$ preconditioners. Again, the number r of permutation matrices used and the number of FGMRES iterations (under the column "it.") are given for BvN$_*$. The last row of the ta-

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

		running time (.s)	
matrix	**MVP**	**Scaling**	**BvN**
two-dom	438	0.67	3.03
watson4a	40368	0.44	0.00
gemat11	2286	0.28	0.04
gemat12	843216	1.01	0.04
lns_3937	3732	0.27	0.04
sherman2	1966	0.06	0.01
west0655	1846	0.03	0.00
bayer09	3626	0.10	0.01
appu	46	0.15	1.57
bundle1	116	0.13	0.78
FEM_3D_th	52	0.04	0.81

Table 4. The total number of sparse matrix-vector and sparse matrix transpose-vector multiply operations (MVP) in the scaling algorithm, the running time of the scaling algorithm and the partial BvN decomposition algorithm in seconds.

		BvN$_r$ it.							BvN$_*$	
matrix	**ILU(0)**	**1**	**2**	**4**	**8**	**16**	**32**	**64**	***r***	**it.**
slide	208	1051	999	745	685	490	783	659	10	663
two-dom	149	579	553	–	610	607	432	360	7	573
watson5a	–	805	839	883	1094	1068	1068	1068	5	947
watson4a	48	160	149	135	120	81	34	5	12	124
bp_200	–	35	26	22	11	6	3	3	8	24
gemat11	239	–	–	–	2486	1588	750	152	–	–
gemat12	344	–	–	–	2607	1615	–	–	–	–
lns_3937	134	1710	969	–	121	23	9	4	10	879
mahindas	–	259	232	180	121	51	–	12	7	232
orani678	–	356	341	320	288	–	225	196	5	316
sherman2	18	327	234	116	65	31	16	8	9	225
west0655	50	298	229	164	101	64	30	–	9	222
west0989	–	199	165	113	63	37	19	8	8	166
west1505	–	343	262	170	92	49	27	12	9	256
west2021	–	411	323	225	115	48	28	12	8	316
circuit_3	–	–	–	–	–	–	–	1178	–	–
bayer09	39	312	193	105	58	–	11	5	7	213
bayer10	–	–	2624	2517	2517	2517	2517	2517	3	2594
lhr01	–	495	314	–	117	65	28	14	7	279
lhr02	–	508	315	–	119	67	28	15	8	277
appu	31	50	50	52	56	62	74	82	7	51
raefsky4	499	1207	722	472	592	403	860	617	6	656
venkat25	163	–	–	1863	–	–	258	–	8	2493
utm5940	–	1972	1588	1173	787	536	196	34	7	1520
bundle1	18	130	123	115	127	137	129	102	5	125
fp	–	–	2953	–	–	–	120	185	6	2779
dw8192	–	–	–	2450	1536	661	82	11	–	–
FEM_3D	7	42	40	40	74	62	27	6	9	38
	Number of successful runs with all 90 matrices									
	35	65	72	51	57	53	56	58	7	82

Table 5. The number of GMRES iterations with different number of permutation matrices in M with tolerance 10^{-6}. We ran GMRES (without restart) for at most 3000 iterations; the symbol "–" flags cases where GMRES did not converge to the required tolerance. *Matrices have negative and positive entries.*

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

	Complexity of M				Running time				
	BvN_r					BvN_r			
	2	4	8	BvN_*	ILU(0)	2	4	8	BvN_*
slide	0.03	0.05	4.04	0.10	38.67	300.65	176.02	148.79	1017.34
two-dom	0.03	0.04	1.31	0.08	56.40	183.60	–	205.53	1122.33
watson4a	0.28	0.36	0.44	0.66	0.05	0.28	0.24	0.19	6.58
gemat11	0.23	0.35	0.65	0.61	16.57	–	–	1548.86	–
gemat12	0.21	0.31	0.58	0.61	34.15	–	–	1578.89	–
lns_3937	0.29	0.55	1.66	0.63	4.68	186.25	–	3.12	359.80
sherman2	0.06	0.10	0.14	0.26	0.02	0.80	0.25	0.09	18.03
west0655	0.42	0.69	1.08	0.62	0.06	0.65	0.38	0.15	13.53
bayer09	0.32	0.52	0.90	0.51	0.05	0.62	0.21	0.08	21.12
appu	0.01	0.02	7.05	0.04	1.64	2.06	2.16	4.64	32.71
raefsky4	0.02	0.02	0.29	0.07	206.02	262.48	116.39	182.10	1027.58
venkat25	0.05	0.53	3.61	0.23	94.29	–	5462.64	–	20382.59
bundle1	0.01	0.01	0.02	0.03	0.50	4.81	4.84	5.46	76.47
FEM_3D	0.05	0.08	0.78	0.27	0.34	1.10	1.10	3.11	30.07

Table 6. The complexity of the preconditioners and the running time of the solver. The complexities of BvN_r are given as the ratio $(\mathrm{nnz}(L+U)-n)/\,\mathrm{nnz}$, where L and U are the factors of the preconditioner (ILU(0) has a complexity of 1.0), and that of BvN_* is given as $\mathrm{nnz}(M)/\,\mathrm{nnz}(A)$. The symbol "–" in the running time column flags the cases where (F)GMRES were unsuccessful. *Matrices have negative and positive entries.*

ble gives the number of successful runs with different preconditioners in all 90 matrices; here the average number of permutation matrices in BvN_* is also given under the column "r". We again use the performance profiles shown in Figure 1b to see the effects of the preconditioners. As seen in this table and the associated performance profile, the preconditioners behave much like they do in the nonnegative case. In particular, BvN_* and BvN_8 deliver the best performance in terms of number of iterations, followed by ILU(0).

We also give the complexity of the preconditioners and the running time of (F)GMRES (total time spent in the iterations) for the cases where ILU(0) resulted in convergence in Table 6. The complexities of the preconditioners remain virtually the same, as expected. As before, the running time is just to give a rough idea in a sequential MATLAB environment. We note that BvN_* required, on average, between 125 and 151 inner iterations. Finally, note that nothing changes for the scaling and BvN decomposition algorithms with respect to the nonnegative case (Table 4 remains as it is for the general case).

4.3 Further Investigations

Here, we give some further experiments to shed light into the behavior of the proposed preconditioners. We compare BvN_* with BvN_r having the same number of permutation matrices. For this purpose, we created a BvN_* preconditioner and used the number r of its permutation matrices to create another preconditioner BvN_r by just taking the first r permutation matrices from a BvN decomposition obtained by Algorithm 1. The results are shown in Table 7, where we give the sum of the coefficients used in constructing the preconditioners and the number of (F)GMRES iterations for a subset of matrices from Table 6, where BvN_* led to a short running time.

As seen in Table 7, BvN_r has a larger sum of coefficients than BvN_* with the same number r of permutation matrices. This is expected as Algorithm 1 finds coefficients in a non-increasing order; hence the first r coefficients are the largest r ones. The number of (F)GMRES iterations is usually inversely proportional to the sum of the coefficients; a higher sum of coefficients usually results in a smaller number of iterations for the same matrix.

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

	$\sum \alpha_i$		it.	
matrix	**BvN$_*$**	**BvN$_r$**	**BvN$_*$**	**BvN$_r$**
watson4a	0.43	0.57	124	111
lns_3937	0.29	0.80	881	68
sherman2	0.24	0.61	226	59
west0655	0.12	0.45	222	97
bayer09	0.19	0.50	216	59
appu	0.09	0.13	50	55
bundle1	0.01	0.01	125	110
FEM_3D	0.35	0.55	39	79

Table 7. Comparing BvN$_*$ with BvN$_r$ having the same number of permutation matrices in terms of the sum of the coefficient of permutation matrices and the number of (F)GMRES iterations (it.). *Matrices have negative and positive entries.*

5 Conclusions and Open Questions

We introduced a class of preconditioners for general sparse matrices based on the Birkhoff–von Neumann decomposition of doubly stochastic matrices. These preconditioners are aimed primarily at solving challenging linear systems with highly unstructured and indefinite coefficient matrices in parallel computing environments. We presented some theoretical results and numerical experiments on linear systems from a variety of applications. We regard this work to be a proof of concept realization; many challenging questions remain to be investigated to render the proposed preconditioners competitive with the standard approaches.

Based on our current theoretical findings, we suggest the use of proposed preconditioners within a Krylov subspace method. There are two ways to go about this. In the first one, the preconditioners are built to satisfy a sufficiency condition that we identified (Theorem 4). This way, the application of the preconditioner requires a small number of highly concurrent steps, where there is no data dependency within a step. In the second alternative, one obtains the LU decomposition of the preconditioners that are built arbitrarily. Here, the proposed preconditioner is therefore constructed as a complete factorization of an incomplete matrix. We demonstrated that using around eight matrices is good enough for this purpose. Beyond that number, the LU decomposition of the preconditioner can become a bottleneck (but remains always cheaper than that of the original matrix). Is there a special way to order these preconditioners for smaller fill-in?

The construction of the preconditioners needs an efficient doubly stochastic scaling algorithm. The known algorithms for this purpose are iterative schemes whose computational cores are sparse matrix-vector multiply operations whose efficient parallelization is well known. For constructing these preconditioners, we then need a bottleneck matching algorithm. The exact algorithms for this purpose are efficient on sequential execution environments, but hard to parallelize efficiently. There are efficiently parallel heuristics for matching problems [18], and more research is needed to parallelize the heuristic for obtaining a BvN decomposition while keeping an eye on the quality of the preconditioner.

Acknowledgment: We thank Alex Pothen for his contributions to this work. This work resulted from the collaborative environment offered by the Dagstuhl Seminar 14461 on High-Performance Graph Algorithms and Applications in Computational Science (November 9–14, 2014).

Funding: The work of Michele Benzi was supported in part by NSF grant DMS-1418889. Bora Uçar was supported in part by French National Research Agency (ANR) project SOLHAR (ANR-13-MONU-0007).

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM

References

[1] P. R. Amestoy, I. S. Duff, D. Ruiz and B. Uçar, A parallel matrix scaling algorithm, in: *High Performance Computing for Computational Science – VECPAR 2008*, Lecture Notes in Comput. Sci. 5336, Springer, Berlin (2008), 301–313.

[2] H. Anzt, E. Chow and J. Dongarra, Iterative sparse triangular solves for preconditioning, in: *Euro-Par 2015: Parallel Processing*, Lecture Notes in Comput. Sci. 9233, Springer, Berlin (2015), 650–651.

[3] M. Benzi, J. C. Haws and M. Tuma, Preconditioning highly indefinite and nonsymmetric matrices, *SIAM J. Sci. Comput.* **22** (2000), no. 4, 1333–1353.

[4] G. Birkhoff, Tres observaciones sobre el algebra lineal, *Univ. Nac. Tucumán Rev. Ser. A* **5** (1946), 147–150.

[5] R. A. Brualdi, Notes on the Birkhoff algorithm for doubly stochastic matrices, *Canad.Math. Bull.* **25** (1982), no. 2, 191–199.

[6] R. A. Brualdi and P. M. Gibson, Convex polyhedra of doubly stochastic matrices. I: Applications of the permanent function, *J. Combin. Theory Ser. A* **22** (1977), no. 2, 194–230.

[7] R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory*, Encyclopedia Math. Appl. 39, Cambridge University Press, Cambridge, 1991.

[8] R. Burkard, M. Dell'Amico and S. Martello, *Assignment Problems*, SIAM, Philadelphia, 2009.

[9] E. Chow and A. Patel, Fine-grained parallel incomplete LU factorization, *SIAM J. Sci. Comput.* **37** (2015), no. 2, C169–C193.

[10] T. A. Davis and Y. Hu, The University of Florida sparse matrix collection, *ACM Trans. Math. Software* **38** (2011), DOI 10.1145/2049662.2049663.

[11] E. D. Dolan and J. J. Moré, Benchmarking optimization software with performance profiles, *Math. Program.* **91** (2002), no. 2, 201–213.

[12] I. S. Duff, A. M. Erisman and J. K. Reid, *Direct Methods for Sparse Matrices*, 2nd ed., Oxford University Press, Oxford, 2017.

[13] I. S. Duff and J. Koster, The design and use of algorithms for permuting large entries to the diagonal of sparse matrices, *SIAM J. Matrix Anal. Appl.* **20** (1999), no. 4, 889–901.

[14] I. S. Duff and J. Koster, On algorithms for permuting large entries to the diagonal of a sparse matrix, *SIAM J. Matrix Anal. Appl.* **22** (2001), 973–996.

[15] F. Dufossé and B. Uçar, Notes on Birkhoff–von Neumann decomposition of doubly stochastic matrices, *Linear Algebra Appl.* **497** (2016), 108–115.

[16] H. N. Gabow and R. E. Tarjan, Algorithms for two bottleneck optimization problems, *J. Algorithms* **9** (1988), no. 3, 411–417.

[17] F. R. Gantmacher, *The Theory of Matrices. Vol. 2*, Chelsea Publishing, New York, 1959.

[18] M. Halappanavar, A. Pothen, A. Azad, F. Manne, J. Langguth and A. M. Khan, Codesign lessons learned from implementing graph matching on multithreaded architectures, *IEEE Computer* **48** (2015), no. 8, 46–55.

[19] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed., Cambridge University, Cambridge, 2013.

[20] P. A. Knight and D. Ruiz, A fast algorithm for matrix balancing, *IMA J. Numer. Anal.* **33** (2013), no. 3, 1029–1047.

[21] P. A. Knight, D. Ruiz and B. Uçar, A symmetry preserving algorithm for matrix scaling, *SIAM J. Matrix Anal. Appl.* **35** (2014), no. 3, 931–955.

[22] M. Manguoglu, M. Koyutürk, A. H. Sameh and A. Grama, Weighted matrix ordering and parallel banded preconditioners for iterative linear system solvers, *SIAM J. Sci. Comput.* **32** (2010), no. 3, 1201–1216.

[23] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, *SIAM J. Sci. Comput.* **14** (1993), no. 2, 461–469.

[24] R. Sinkhorn and P. Knopp, Concerning nonnegative matrices and doubly stochastic matrices, *Pacific J. Math.* **21** (1967), 343–348.

[25] R. S. Varga, *Matrix Iterative Analysis*, 2nd ed., Springer, Berlin, 2000.

Authenticated | benzi@mathcs.emory.edu author's copy
Download Date | 3/28/17 4:01 PM