Stephen D. Shank

Department of Mathematics
Temple University, Philadelphia, PA
http://math.temple.edu/~sshank

Joint Work with:
David Fritzsche, formerly of Temple University, now at WestLB (private sector)
Andreas Frommer, University of Wuppertal, Germany
Daniel B. Szyld, Temple University, Philadelphia, PA

June 19th, 2011
Overview

Goal: precondition an arbitrary linear system

\[ A x = b \]

\( A \in \mathbb{R}^{n \times n} \) is nonsingular, sparse and non-symmetric. A class of preconditioners that depend only on the associated graph of \( A \) will be discussed, with related topics including:

1. Algebraic Schwarz methods as a simple abstraction of classical domain decomposition algorithms
2. Graph partitioning algorithms and their applications as block preconditioners
3. A proposed method for strengthening a given block preconditioner by adding overlap
4. A few implementation details and numerical results
Throughout we assume that all matrices have been scaled by the Harwell’s Subroutine Library’s MC64\(^1\) routine.

This permutes and scales the matrix such that (at least) the following holds:

- every diagonal element of the scaled/permuted matrix has value absolute value 1
- off diagonal elements are less than or equal to 1 in absolute value

**Reference**

Every such matrix possesses an associated weighted digraph $G(V, E)$, defined by setting $V = \{1, \cdots, n\}$, $E = \{(i, j) | a_{ij} \neq 0, i \neq j\}$, $w_E((i, j)) = |a_{ij}|$.

Intuitive definitions are made for notions such as vertex/node, adjacency, incidence, block/subgraph, connectivity, etc.
Subsets \( \{V_i\}_{i=1}^{q} \) of \( V \) are called a

- **decomposition** of \( V \) if \( \bigcup_{i=1}^{q} V_i = V \)
- **partition** if additionally \( V_i \cap V_j = \emptyset \) for \( i \neq j \).

A decomposition that is not a partition is said to have **overlap**.

Let \( S = \{s_1, \ldots, s_m\} \subseteq V \). The **restriction operator** \( R_S \in \mathbb{R}^{m \times n} \) associated with \( S \) is defined by

\[
(R_S)_{ij} := \delta_{s_i,j} = \begin{cases} 1 & \text{if } s_i = j, \\ 0 & \text{otherwise.} \end{cases}
\]

\( R_S^T \) is the corresponding **prolongation operator**. We observe that \( R_{V_i} A R_{V_i}^T := A_i \) is the submatrix of \( A \) associated with the node set \( V_i \).
Example: Let $A \in \mathbb{R}^{5 \times 5}$ and set $V = \{1, 2, 3, 4, 5\}$ and $S = \{2, 3, 5\}$. Then

$$R_S \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} x_2 \\ x_3 \\ x_5 \end{pmatrix}$$

and

$$R_S^T \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ x \\ y \\ 0 \\ z \end{pmatrix}$$

and

$$R_SAR_S^T = \begin{pmatrix} a_{22} & a_{23} & a_{25} \\ a_{32} & a_{33} & a_{35} \\ a_{52} & a_{53} & a_{55} \end{pmatrix}$$
Given a decomposition and initial vector $x_0$, the algebraic Schwarz methods cycle through the $q$ sets of vertices $V_i$, updating an approximation $x_k$ by adding a prolonged correction of a local error equation:

### Algorithms for Algebraic Additive and Multiplicative Schwarz

**Additive Schwarz (AS)**

\[
\text{for } k = 0, 1, 2, \ldots \text{ do} \\
\text{for } i = 1, \ldots, q \text{ do} \\
\quad \text{Solve } A_i e_i = R V_i (b - A x_k) \\
\text{end for} \\
\text{Set } x_{k+1} = x_k + \sum_{i=1}^{q} R_{V_i}^T e_i \\
\text{end for}
\]

**Multiplicative Schwarz (MS)**

\[
\text{for } k = 0, 1, 2, \ldots \text{ do} \\
\quad x_{k+1} = x_k \\
\text{for } i = 1, \ldots, q \text{ do} \\
\quad \text{Solve } A_i e_i = R V_i (b - A x_{k+1}) \\
\quad x_{k+1} := x_{k+1} + R_{V_i}^T e_i \\
\text{end for} \\
\text{end for}
\]

Can be viewed as an algebraic analogue of domain decomposition methods.
Either of the previous can be written as a stationary iteration

\[ x_{k+1} = Gx_k + f \]

and convergence theory exists\(^2\) for

- a number of particular applications
- M-matrices and symmetric positive definite matrices

Bottom line: AS and MS provide approximations to \( A^{-1}b \) and can therefore be used as preconditioners.

There is also the *Restricted Additive Schwarz* (RAS) method, where \( V_i \subset W_i \) and restriction is performed relative to \( W_i \), while prolongation is performed only relative to \( V_i \).

**Reference**

2) M. Benzi, A. Frommer, R. Nabben, and D. B. Szyld, Algebraic theory of multiplicative Schwarz methods, Numerische Mathematik, 89 (2001)
The question arises: how to obtain the sets \( V_i \)? Many possibilities exist, including

- \( V_i = \{i\} \), yielding the classic Jacobi and Gauss-Seidel iterations
- PABLO and its derivatives, e.g. XPABLO \(^3\), which yield dense subgraphs (many edges) with large weights
- METIS\(^4\), which was designed to reduce fill-in from elimination

References


An example of a partitioned matrix

As an example, here is a color spy plot of the matrix ohne2 from the University of Florida Sparse Matrix Collection before and after the MC64 and XPABLO algorithms.

**Figure:** The matrix ohne2 before and after scaling and permuting
Towards obtaining a decomposition

Desire to maintain connectivity

Let $G(A) = (V, E)$ be the digraph of a matrix $A \in \mathbb{R}^{n \times n}$. Let $\{V_i\}_{i=1, \ldots, q}$ be a decomposition of $V$. Let $j \in \{1, \ldots, q\}$ be such that $G|_{V_j}$ is not connected. Let $C_1, \ldots, C_m$ be the connected components of $G|_{V_j}$. Then

$$MS(V_1, \ldots, V_q) = MS(V_1, \ldots, V_{j-1}, C_1, \ldots, C_m, V_{j+1}, \ldots, V_q)$$

This result is interpreted as saying that if one wishes to grow a block, one should attempt to maintain connectivity within this block’s subgraph.

Accordingly, define the *adjacency set* of a subset $S \subset V$ as

$$\text{adj}(S) = \{j \in V : j \not\in S \text{ and } j \text{ is adjacent to some } i \in S\}.$$ 

The notion of a *level set* is used to generalize the notion of an adjacency set.
Definition of level sets

The $k$th level set $L_k(S)$ with respect to $S$ is defined as

$$L_k(S) := \begin{cases} 
S & \text{if } k = 0, \\
\text{adj}(S) & \text{if } k = 1, \\
\text{adj}(L_{k-1}(S)) \setminus L_{k-2}(S) & \text{if } k > 1.
\end{cases}$$

Figure: Level sets $L_0(S)$ through $L_4(S)$ with respect to the node set $S$, and their incident edges.
A possible method arises where overlap is obtained by growing every $V_i$ to $W_i$ by taking $\ell \geq 1$ level sets of each such subblock\(^5\).

Poses an issue for certain test matrices:

Timings for RAEFSKY2

<table>
<thead>
<tr>
<th>Rounds of overlap</th>
<th>Time</th>
<th>Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero rounds</td>
<td>0.665</td>
<td>39</td>
</tr>
<tr>
<td>one round</td>
<td>0.846</td>
<td>16</td>
</tr>
<tr>
<td>two rounds</td>
<td>1.388</td>
<td>10</td>
</tr>
<tr>
<td>three rounds</td>
<td>2.230</td>
<td>2</td>
</tr>
</tbody>
</table>

References

Alternative: Provide a stronger criterion for adding a vertex to a block!

**Sketch of OBGP(\(\ell\)) algorithm**

```plaintext
for \(i = 1, \ldots, q\) do
    \(B := V_i\)
    for \(k = 1, \ldots, \ell\) do
        determine \(L := \text{adj}(B)\) \{set of candidate nodes\}
        select some nodes \(N \subset L\) for inclusion
        \(B := B \cup N\)
    end for
end for
\(W_i := B\)
```

Functions \(\mu : \mathbb{N} \rightarrow \mathbb{N}\) and \(\nu : \mathbb{N} \rightarrow \mathbb{N}\) are chosen a priori so that letting \(B^{(k)}\) denote the block at the \(k\)th step,

- Only \(\mu(|B^{(k-1)}|)\) nodes are added at each step
- Only \(\nu(|B^{(0)}|)\) total nodes are added

Note that only nodes in the union of the first \(\ell\) level sets may be chosen.
Criteria for selection

Let \( G = (V, E) \) be a directed graph with edge weights \( w_E(e), e \in E \).

Given \( S \subset V \), we define inc\((S) = \bigcup_{v \in S} \{ e \in E : e \text{ incident to } v \} \)

Definition of weight with respect to a block

Let \( G = (V, E) \) be a directed graph with edge weights \( w_E(e), e \in E \). Let \( B \subset V \) be a set of nodes and \( j \in V \). The weight \( w(j, B) \) of \( j \) with respect to \( B \) is defined as

\[
    w(j, B) := \sum_{e \in \text{inc}(\{j\} \cup B)} |w_E(e)|.
\]

This measures, in some sense, the “strength of connectivity” of the node \( j \) with respect to the set \( B \).

In OBGP, only nodes of largest weight (as defined above) are included in a given round.
Implementation details

A progressive implementation is possible; details are omitted.

The data structure chosen for the set $L$ of candidate notes of a given block is a binary heap, which possesses several nice properties:

- Can iterate over nodes in $L$, which need not be done in a particular order
- Insert or remove a node, done in $O(\log(|L|))$ complexity
- Find nodes of largest weight, also $O(\log(|L|))$ complexity

In our experiments we took $1 \leq \ell \leq 20$ and set $\mu(|B^{(k-1)}|) = \left\lceil \alpha \cdot \sqrt{|B^{(k-1)}|} \right\rceil$, for $\alpha \in [.5, 5]$, a natural choice which can be shown to have a nice consequence on complexity:

**Complexity of OBGP**

Let $L$ be stored in a heap. Then algorithm OBGP($\ell$) can be implemented in such a way that the time complexity is $O(q \cdot (\text{nnz}(A) + n \log n))$. 
Associated costs are

- Scale/permute via MC64
- Running the block partitioning algorithm, obtaining $q$ blocks
- Growing the blocks, based on $\alpha, \ell, \mu$ and $\nu$ to obtain adjacent nodes
- Perform an LU decomposition of each associated submatrix
- Applying the preconditioner each round; i.e., perform a linear solve with the LU decomposition obtained from each block
- Costs associated with iterative method

Two noteworthy things are

- Potential for parallelization of the AS and RAS algorithms
- Hybrid approach of using both direct and iterative solvers.
Numerical results, part 1

Several numerical results will be presented. All cases use left preconditioned restarted GMRES(50), stopping when $\frac{\|r_k\|}{\|b\|} \leq \sqrt{\epsilon_M}$.

A snapshot of the timings involved on a particular case:

<table>
<thead>
<tr>
<th>Block partitioner</th>
<th>ohne2</th>
<th>Process</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix ohne2</td>
<td></td>
<td>MC64</td>
<td>1.49</td>
</tr>
<tr>
<td>α</td>
<td></td>
<td>XPABLO</td>
<td>.98</td>
</tr>
<tr>
<td>ℓ</td>
<td></td>
<td>OBGPS</td>
<td>.60</td>
</tr>
<tr>
<td>q</td>
<td></td>
<td>LU for all blocks</td>
<td>20.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GMRES(50)</td>
<td>168.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>192.74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block partitioner</th>
<th>ohne2</th>
<th>Process</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix ohne2</td>
<td></td>
<td>MC64</td>
<td>1.46</td>
</tr>
<tr>
<td>α</td>
<td></td>
<td>XPABLO</td>
<td>.94</td>
</tr>
<tr>
<td>ℓ</td>
<td></td>
<td>OBGPS</td>
<td>.64</td>
</tr>
<tr>
<td>q</td>
<td></td>
<td>LU for all blocks</td>
<td>23.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GMRES(50)</td>
<td>84.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>111.69</td>
</tr>
</tbody>
</table>
Comparison with ILU(τ): We compare with certain matrices from the UF Sparse Matrix Collection, along with matrices arising in a meshfree discretization of the Poisson equation.

MATLAB’s implementation of threshold ILU, ILU(τ), was ran for $\tau = 10^{-1}, 10^{-2}, 10^{-3}$ and $10^{-4}$, with the best time reported.

METIS (M) and XPABLO (X) were ran with OBGP and multiplicative Schwarz, with default parameters $\alpha = 2$ and $\ell = 10$, with the better timing between the two reported.
Restarted GMRES(50) with **ILU** and **OBGP** (default parameters, $\alpha = 2$, $\ell = 10$)

<table>
<thead>
<tr>
<th>Matrix</th>
<th>$\log_{10} \tau$</th>
<th>Time</th>
<th>It</th>
<th>Method</th>
<th>Time</th>
<th>It</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARA-8</td>
<td>-3</td>
<td>34.5</td>
<td>50</td>
<td>M10</td>
<td>18.44</td>
<td>15</td>
</tr>
<tr>
<td>OHNE2</td>
<td>-3</td>
<td>180.1</td>
<td>296</td>
<td>X10</td>
<td>111.83</td>
<td>91</td>
</tr>
<tr>
<td>2D_54019_HIGHK</td>
<td>-3</td>
<td>5.58</td>
<td>46</td>
<td>X10</td>
<td>3.46</td>
<td>28</td>
</tr>
<tr>
<td>3D_51448_3D</td>
<td>-2</td>
<td>5.07</td>
<td>65</td>
<td>X10</td>
<td>3.45</td>
<td>15</td>
</tr>
<tr>
<td>IBM_MATRIX_2</td>
<td>-2</td>
<td>4.84</td>
<td>59</td>
<td>X10</td>
<td>3.77</td>
<td>16</td>
</tr>
<tr>
<td>MATRIX_9</td>
<td>-2</td>
<td>17.3</td>
<td>45</td>
<td>X10</td>
<td>16.89</td>
<td>42</td>
</tr>
<tr>
<td>MATRIX-NEW_3</td>
<td>-2</td>
<td>15.9</td>
<td>89</td>
<td>M10</td>
<td>16.41</td>
<td>34</td>
</tr>
<tr>
<td>LSQ_2D_200000</td>
<td>-3</td>
<td>43.1</td>
<td>63</td>
<td>M10</td>
<td>28.5</td>
<td>28</td>
</tr>
<tr>
<td>LSQ_2D_400000</td>
<td>-2</td>
<td>239.1</td>
<td>273</td>
<td>M10</td>
<td>77.6</td>
<td>39</td>
</tr>
<tr>
<td>MPS_2D_200000</td>
<td>-4</td>
<td>22.3</td>
<td>28</td>
<td>M10</td>
<td>15.79</td>
<td>29</td>
</tr>
<tr>
<td>MPS_2D_400000</td>
<td>-3</td>
<td>91.2</td>
<td>97</td>
<td>M10</td>
<td>51.8</td>
<td>41</td>
</tr>
<tr>
<td>LSQ_3D_100000</td>
<td>-2</td>
<td>39.3</td>
<td>28</td>
<td>X10</td>
<td>28.9</td>
<td>15</td>
</tr>
<tr>
<td>MPS_3D_100000</td>
<td>-2</td>
<td>8.04</td>
<td>30</td>
<td>M10</td>
<td>9.38</td>
<td>15</td>
</tr>
<tr>
<td>MPS_3D_200000</td>
<td>-2</td>
<td>23.1</td>
<td>37</td>
<td>M10</td>
<td>24.5</td>
<td>19</td>
</tr>
</tbody>
</table>
Finally, optimal timings are given from a (somewhat) extensive parameter study of the method.

Best times are reported from the choice of

- $\alpha = .5, 1, 2, 3, 4, 5$ and $\infty$
- $\ell = 0, 1, 2, 3, 4, 5, 10, 15, 20$

where $\alpha = \infty$ denotes including every node in a given level set, and omitted $\ell$ corresponds to no overlap.

Metis (M) and XPABLO (X) were again used as graph partitioners, again with left preconditioning of GMRES(50).
Numerical results, part 3 continued

Timings for AS, RAS, and MS, respectively

<table>
<thead>
<tr>
<th></th>
<th>Md</th>
<th>α</th>
<th>Time</th>
<th>It</th>
<th></th>
<th>Md</th>
<th>α</th>
<th>Time</th>
<th>It</th>
<th></th>
<th>Md</th>
<th>α</th>
<th>Time</th>
<th>It</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M10</td>
<td>2</td>
<td>28.14</td>
<td>39</td>
<td></td>
<td>M10</td>
<td>3</td>
<td>23.7</td>
<td>22</td>
<td></td>
<td>M10</td>
<td>2</td>
<td>18.44</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>X10</td>
<td>3</td>
<td>387.1</td>
<td>375</td>
<td></td>
<td>X10</td>
<td>3</td>
<td>172.4</td>
<td>163</td>
<td></td>
<td>X10</td>
<td>3</td>
<td>82.97</td>
<td>48</td>
</tr>
<tr>
<td>3</td>
<td>M15</td>
<td>2</td>
<td>6.55</td>
<td>48</td>
<td></td>
<td>M20</td>
<td>2</td>
<td>5.4</td>
<td>28</td>
<td></td>
<td>X5</td>
<td>4</td>
<td>3.35</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>X5</td>
<td>1</td>
<td>4.95</td>
<td>47</td>
<td></td>
<td>X2</td>
<td>.5</td>
<td>4.33</td>
<td>44</td>
<td></td>
<td>X10</td>
<td>.5</td>
<td>2.89</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>X4</td>
<td>1</td>
<td>5.78</td>
<td>56</td>
<td></td>
<td>X5</td>
<td>4</td>
<td>4.95</td>
<td>28</td>
<td></td>
<td>X3</td>
<td>2</td>
<td>3.15</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>M2</td>
<td>1</td>
<td>33.28</td>
<td>92</td>
<td></td>
<td>M5</td>
<td>1</td>
<td>26.43</td>
<td>60</td>
<td></td>
<td>X15</td>
<td>1</td>
<td>14.35</td>
<td>47</td>
</tr>
<tr>
<td>7</td>
<td>X</td>
<td></td>
<td>24.24</td>
<td>126</td>
<td></td>
<td>X</td>
<td></td>
<td>25.7</td>
<td>126</td>
<td></td>
<td>X20</td>
<td>1</td>
<td>15.5</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>M3</td>
<td>∞</td>
<td>46.6</td>
<td>70</td>
<td></td>
<td>M3</td>
<td>∞</td>
<td>38.0</td>
<td>48</td>
<td></td>
<td>M3</td>
<td>∞</td>
<td>24.8</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>M20</td>
<td>1</td>
<td>194.6</td>
<td>117</td>
<td></td>
<td>M2</td>
<td>∞</td>
<td>157.9</td>
<td>80</td>
<td></td>
<td>M20</td>
<td>1</td>
<td>89.6</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>M20</td>
<td>4</td>
<td>27.5</td>
<td>41</td>
<td></td>
<td>M20</td>
<td>4</td>
<td>20.1</td>
<td>27</td>
<td></td>
<td>M10</td>
<td>4</td>
<td>13.3</td>
<td>21</td>
</tr>
<tr>
<td>11</td>
<td>M20</td>
<td>5</td>
<td>95.5</td>
<td>53</td>
<td></td>
<td>M20</td>
<td>5</td>
<td>73.3</td>
<td>36</td>
<td></td>
<td>M20</td>
<td>5</td>
<td>45.4</td>
<td>21</td>
</tr>
<tr>
<td>12</td>
<td>M</td>
<td></td>
<td>24.7</td>
<td>39</td>
<td></td>
<td>M1</td>
<td>.5</td>
<td>25.2</td>
<td>38</td>
<td></td>
<td>X</td>
<td></td>
<td>19.8</td>
<td>27</td>
</tr>
<tr>
<td>13</td>
<td>M10</td>
<td>.5</td>
<td>12.9</td>
<td>43</td>
<td></td>
<td>M1</td>
<td>∞</td>
<td>10.8</td>
<td>27</td>
<td></td>
<td>M3</td>
<td>1</td>
<td>8.3</td>
<td>23</td>
</tr>
<tr>
<td>14</td>
<td>M3</td>
<td>.5</td>
<td>35.2</td>
<td>63</td>
<td></td>
<td>M1</td>
<td>∞</td>
<td>32.6</td>
<td>32</td>
<td></td>
<td>M4</td>
<td>.5</td>
<td>21.9</td>
<td>29</td>
</tr>
</tbody>
</table>
A method for potentially strengthening an arbitrary preconditioner based on graph partitioning has been presented.

Hopefully it has been convincingly argued that for such classes of preconditioners, overlap of subgraphs can be a good idea, especially when done in a judicious manner.

Technical report/preprint with more details and experimental results can be found at:

http://www.math.temple.edu/~sshank
http://www.math.temple.edu/~szyld

Code will (hopefully) soon be made available for use within the community.
Thank you!

Thanks for your time!!