The Augmented Block-Cimmino Distributed Method

Improvements to the Conjugate Gradient accelerated Block Cimmino Method

Mohamed Zenadi¹, Iain S. Duff², Ronan Guivarch¹, Daniel Ruiz¹,
IRIT - ENSEEIHT¹, CERFACS and Rutherford Appleton Laboratory²

Jun 17, 2013
Block Cimmino: The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]

Partitioning the system $Ax = b$

$$
\begin{pmatrix}
A_1 \\
A_2 \\
\vdots \\
A_p
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
\vdots \\
x_p
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2 \\
\vdots \\
b_p
\end{pmatrix}
$$
Block Cimmino: The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]

Partitioning the system \( Ax = b \)

\[
\begin{pmatrix}
A_1 \\
A_2 \\
\vdots \\
A_p
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
\vdots \\
x_p
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2 \\
\vdots \\
b_p
\end{pmatrix}
\]

Partitions can be obtained by cutting uniformly the matrix (with no permutation) or using a partitioner (we use PaToH in our examples)
Block Cimmino : The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]

The Block Cimmino Iteration

\[
\delta_i^{(k)} = A_i^+ b_i - P_{\mathcal{R}(A_i^T)} x^{(k)}
\]

\[
x^{(k+1)} = x^{(k)} + \nu \sum_{i=1}^{p} \delta_i^{(k)}
\]

where:

\[
A_i^+ = A_i^T (A_i A_i^T)^{-1}
\]

and

\[
P_{\mathcal{R}(A_i^T)} = A_i^+ A_i
\]
Block Cimmino: The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]
- CG acceleration: iteration matrix is \( \sum_{i=1}^{p} A_i^+ A_i = \sum_{i=1}^{p} P_R(A_i^T) \)

Acceleration

Apply CG to solve the SPD system

\[
\sum_{i=1}^{p} A_i^+ A_i \mathbf{x} = \sum_{i=1}^{p} A_i^+ b_i
\]
### Block Cimmino: The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]

- CG acceleration: iteration matrix is \( \sum_{i=1}^{p} A_i^+ A_i = \sum_{i=1}^{p} P_{\mathcal{R}(A_i^T)} \)

- Can also exploit 2nd and 3rd levels of parallelism (sparsity structure, BLAS3 Kernels)

### Projections:

\[
\delta_i^{(k)} = A_i^+ b_i - P_{\mathcal{R}(A_i^T)} x^{(k)}
\]
Block Cimmino: The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]
- CG acceleration: iteration matrix is \( \sum_{i=1}^{p} A_i^+ A_i = \sum_{i=1}^{p} P_{R(A_i^T)} \)
- Can also exploit 2nd and 3rd levels of parallelism (sparsity structure, BLAS3 Kernels)

Projections: \( \delta_i^{(k)} = A_i^+ b_i - P_{R(A_i^T)} x^{(k)} \)

Solve independently using MUMPS the systems for each partition

\[
\begin{bmatrix}
I & A_i^T \\
A_i & 0
\end{bmatrix}
\begin{bmatrix}
u_i \\
v_i
\end{bmatrix} =
\begin{bmatrix}
0 \\
b_i - A_i x
\end{bmatrix}
\]

where: \( u_i = A_i^+ (b_i - A_i x) = \delta_i \)
A path to orthogonality

Issues with Block-Cimmino:

• Convergence is problem dependent
• Erratic convergence behaviour (plateaux based)
• Trail of small eigenvalues
• Multiple solves require a re-run of Block-CG (too expensive)

Proposed solution:

• Enforce numerical orthogonality between partitions by adding extra variables and constraints
• Extract a condensed smaller subsystem (similar to Schur complement techniques) that can be reused for efficient further solves

⇒ Augmented Block Cimmino Distributed solver (ABCD solver)
A path to orthogonality

Issues with Block-Cimmino:

• Convergence is problem dependent
• Erratic convergence behaviour (plateaux based)
• Trail of small eigenvalues
• Multiple solves require a re-run of Block-CG (too expensive)

Proposed solution:

• Enforce numerical orthogonality between partitions by adding extra variables and constraints
• Extract a condensed smaller subsystem (similar to Schur complement techniques) that can be reused for efficient further solves
A path to orthogonality

Issues with Block-Cimmino:

- Convergence is problem dependent
- Erratic convergence behaviour (plateaux based)
- Trail of small eigenvalues
- Multiple solves require a re-run of Block-CG (too expensive)

Proposed solution:

- Enforce numerical orthogonality between partitions by adding extra variables and constraints
- Extract a condensed smaller subsystem (similar to Schur complement techniques) that can be reused for efficient further solves

⇒ Augmented Block Cimmino Distributed solver (ABCD solver)
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions ($j > i$), expand with $C_{i,j} = A_iA_j^T$,
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions \((j > i)\), expand with \(C_{i,j} = A_iA_j^T\), and enforce numerical orthogonality

Illustrative example
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions \((j > i)\), expand with \(C_{i,j} = A_iA_j^T\), and enforce numerical orthogonality to obtain \(\bar{A} = [A \quad C]\)
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions \((j > i)\), expand with \(C_{i,j} = A_i A_j^T\), and enforce numerical orthogonality to obtain \(\bar{A} = [A \quad C]\)
- Add extra constraints to build an equivalent linear system:

\[
\begin{bmatrix}
A & C \\
0 & I
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix} =
\begin{bmatrix}
b \\
0
\end{bmatrix},
\]

where \(y = 0\) ensures the same solution \(x\).
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions \((j > i)\), expand with \(C_{i,j} = A_i A_j^T\), and enforce numerical orthogonality to obtain \(\bar{A} = [A \ C]\)
- Add extra constraints to build an equivalent linear system:

\[
\begin{bmatrix}
A & C \\
0 & I
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix} =
\begin{bmatrix}
b \\
0
\end{bmatrix},
\]

where \(y = 0\) ensures the same solution \(x\).

**Problem**

The extra partition \(Y = [0 \ I]\), linked to the constraints equations, is not orthogonal to the previous partitions in \(\bar{A} = [A \ C]\).
The augmentation process

To enforce this orthogonality, we project the column vectors $Y^T$ onto the null space of $\bar{A} = [A \ C]$ (orthogonal complement of $\mathcal{R}(\bar{A}^T)$) :

$$W^T = (I - P) Y^T,$$

where (as a result of the enforced orthogonality) :

$$P = P_{\mathcal{R}(\bar{A}^T)} = P \bigoplus_{i=1}^{p} \mathcal{R}(\bar{A}_i^T) = \sum_{i=1}^{p} P_{\mathcal{R}(\bar{A}_i^T)}$$

We finally obtain $[A \ C \ B \ S]$, where $[B \ S] = W$, an augmented matrix with mutually numerically orthogonal partitions.
The augmentation process

Illustrative example
The augmentation process

To keep the consistency within the solution of the new system:

\[
\begin{bmatrix}
A & C \\
B & S
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
= 
\begin{bmatrix}
b \\
f
\end{bmatrix}
\]

we compute the right hand side \( f \) as:

\[
f = \begin{bmatrix} B & S \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix} = Y (I - P) \begin{bmatrix} x \\ 0 \end{bmatrix} = -YP \begin{bmatrix} x \\ 0 \end{bmatrix} \quad \text{(since } Y = \begin{bmatrix} 0 & I \end{bmatrix} \text{)}
\]

\[
= -Y \bar{A}^+ \bar{A} \begin{bmatrix} x \\ 0 \end{bmatrix}
\]

\[
f = -Y \bar{A}^+ b
\]
Since all the partitions in the new equivalent linear system

\[
\begin{bmatrix}
  A & C \\
  B & S
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
= 
\begin{bmatrix}
  b \\
  f
\end{bmatrix}
\]

are mutually numerically orthogonal, the Cimmino iteration matrix becomes the Identity matrix, and the solution can be directly obtained as:

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
= \bar{A}^+ b + W^+ f
= \bar{A}^+ b - W^+ Y \bar{A}^+ b
= \sum_{i=1}^{p} \bar{A}_i^+ b_i - W^+ Y \sum_{i=1}^{p} \bar{A}_i^+ b_i
\]
Knowing that $W = [B \quad S] = Y (I - P)$, with $Y = [0 \quad I]$, we have:

\[
WW^T = Y (I - P) (I - P)^T Y^T = Y (I - P)^2 Y^T = Y (I - P) Y^T = [B \quad S] Y^T = S
\]
Knowing that \( W = [B \quad S] = Y (I - P) \), with \( Y = [0 \quad I] \), we have:

\[
WW^T = Y (I - P) (I - P)^T Y^T
\]

\[
= Y (I - P)^2 Y^T
\]

\[
= Y (I - P) Y^T
\]

\[
= [B \quad S] Y^T
\]

\[
= S
\]

Therefore \( S = Y (I - P) Y^T \) and is SPD.
Knowing that $W = \begin{bmatrix} B & S \end{bmatrix} = Y (I - P)$, with $Y = \begin{bmatrix} 0 & I \end{bmatrix}$, we have:

$$WW^T = Y (I - P) (I - P)^T Y^T$$
$$= Y (I - P)^2 Y^T$$
$$= Y (I - P) Y^T$$
$$= \begin{bmatrix} B & S \end{bmatrix} Y^T$$
$$= S$$

Therefore $S = Y (I - P) Y^T$ and is SPD.

And the pseudo inverse $W^+ = W^T (WW^T)^{-1}$ is given by

$$W^+ = W^T S^{-1}$$
$$W^+ = (I - P) Y^T S^{-1}$$
The solution is thus given by:

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix} = \bar{A}^+ b + W^+ f \\
= \bar{A}^+ b - (I - P) Y^T S^{-1} Y \bar{A}^+ b
\]
The solution is thus given by:

\[
\begin{bmatrix}
\chi \\
\gamma
\end{bmatrix} = \tilde{A}^+ b + W^+ f \\
= \tilde{A}^+ b - (I - P) Y^T S^{-1} Y \tilde{A}^+ b
\]

which can be computed through the 4 following steps:
The solution is thus given by:

\[
\begin{bmatrix}
x \\
y
\end{bmatrix} = \bar{A}^+ b + W^+ f
\]

\[
= \bar{A}^+ b - (I - P) Y^T S^{-1} Y \bar{A}^+ b
\]

which can be computed through the 4 following steps:

- Build \( w = \bar{A}^+ b \) and then by simple restriction set \( f = -Yw \)
The solution is thus given by:

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix} = \bar{A}^+ b + W^+ f
\]

\[
= \bar{A}^+ b - (I - P) Y^T S^{-1} Y \bar{A}^+ b
\]

which can be computed through the 4 following steps:

- Build \( w = \bar{A}^+ b \) and then by simple restriction set \( f = -Yw \)
- Solve \( Sz = f \) (\( S \) should be small enough)
The solution is thus given by:

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix} = \tilde{A}^+ b + W^+ f \\
= \tilde{A}^+ b - (I - P) Y^T S^{-1} Y \tilde{A}^+ b
\]

which can be computed through the 4 following steps:

- Build \( w = \tilde{A}^+ b \) and then by simple restriction set \( f = -Yw \)
- Solve \( Sz = f \) (\( S \) should be small enough)
- Expand \( z \) and then project it onto the null space of \( \tilde{A} \) viz.
  \[ u = (I - P) Y^T z \]
Computational Ingredients

The solution is thus given by:

\[
\begin{bmatrix}
x \\
y
\end{bmatrix} = \bar{A}^+ b + W^+ f
\]

\[
= \bar{A}^+ b - (I - P) Y^T S^{-1} Y \bar{A}^+ b
\]

which can be computed through the 4 following steps:

• Build \( w = \bar{A}^+ b \) and then by simple restriction set \( f = -Yw \)
• Solve \( Sz = f \) (\( S \) should be small enough)
• Expand \( z \) and then project it onto the null space of \( \bar{A} \) viz.
  \[
  u = (I - P) Y^T z
  \]
• Then sum \( w + u \) to obtain the solution \( \begin{bmatrix} x \\ y \end{bmatrix} \) (where \( y = 0 \))
Computational Ingredients

The solution is thus given by:

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
= \bar{A}^+ b + W^+ f
\]
\[
= \bar{A}^+ b - (I - P) Y^T S^{-1} Y \bar{A}^+ b
\]

which can be computed through the 4 following steps:

- Build \( w = \bar{A}^+ b \) and then by simple restriction set \( f = -Yw \)
- Solve \( Sz = f \) (\( S \) should be small enough)
- Expand \( z \) and then project it onto the null space of \( \bar{A} \) viz. \( u = (I - P) Y^T z \)
- Then sum \( w + u \) to obtain the solution \( \begin{bmatrix} x \\ y \end{bmatrix} \) (where \( y = 0 \))

Note that we don’t need to build \( B \), only \( S \) is used.
bayer01(57k) - 16 uniform parts - N(S) = 2409
<table>
<thead>
<tr>
<th></th>
<th>BC (blk. size = 16)</th>
<th>ABCD (size $S = 6506$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fact.</td>
<td>0.2 s.</td>
<td>0.2s.</td>
</tr>
<tr>
<td>CG</td>
<td>(521itr) 107.6 s.</td>
<td>-</td>
</tr>
<tr>
<td>Augmentation</td>
<td>-</td>
<td>0.16s.</td>
</tr>
<tr>
<td>Build S</td>
<td>-</td>
<td>5.5s.</td>
</tr>
<tr>
<td>Fact S</td>
<td>-</td>
<td>1.2s.</td>
</tr>
</tbody>
</table>

Thanks to François-Henry Rouet for the hints on Exploit-Sparsity feature (available in the future release of MUMPS).
### bmw3_2 (227k) 16 partitions on 32 cores

<table>
<thead>
<tr>
<th></th>
<th><strong>BC</strong> (blk. size = 1)</th>
<th><strong>ABCD</strong> (size $S = 16695$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fact.</td>
<td>1.7 s.</td>
<td>1.97s.</td>
</tr>
<tr>
<td>CG</td>
<td><em>(Failed)</em> 176.5 s.</td>
<td>-</td>
</tr>
<tr>
<td>Augmentation</td>
<td>-</td>
<td>0.6s.</td>
</tr>
<tr>
<td>Build S</td>
<td>-</td>
<td>40.0s.</td>
</tr>
<tr>
<td>Fact S</td>
<td>-</td>
<td>18.0s.</td>
</tr>
</tbody>
</table>

Thanks to François-Henry Rouet for the hints on Exploit-Sparsity feature (available in the future release of MUMPS).
# ABCD results

Hamrle3 (1.447M), non-symmetric, 64 partitions on 32 cores

<table>
<thead>
<tr>
<th></th>
<th>BC (blk. size = 4)</th>
<th>ABCD (size $S = 54608$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fact.</td>
<td>2.13 s.</td>
<td>3.4s.</td>
</tr>
<tr>
<td>CG</td>
<td>(615itr) 282.90 s.</td>
<td>-</td>
</tr>
<tr>
<td>Augmentation</td>
<td>-</td>
<td>4.6s.</td>
</tr>
<tr>
<td>Build S</td>
<td>-</td>
<td>145.4s.</td>
</tr>
<tr>
<td>Fact S</td>
<td>-</td>
<td>49.1s.</td>
</tr>
</tbody>
</table>

Thanks to François-Henry Rouet for the hints on Exploit-Sparsity feature (available in the future release of MUMPS).
<table>
<thead>
<tr>
<th>BC (blk. size = 4)</th>
<th>ABCD (size S = 54608)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fact.</td>
<td>2.13 s.</td>
</tr>
<tr>
<td>CG</td>
<td>(615itr) 282.90 s.</td>
</tr>
<tr>
<td>Augmentation</td>
<td>-</td>
</tr>
<tr>
<td>Build S</td>
<td>-</td>
</tr>
<tr>
<td>Fact S</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>BC (blk. size = 4)</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Fact.</td>
<td>2.13 s.</td>
</tr>
<tr>
<td>CG</td>
<td>$(615\text{itr})$ 282.90 s.</td>
</tr>
<tr>
<td>Augmentation</td>
<td>-</td>
</tr>
<tr>
<td>Build S</td>
<td>-</td>
</tr>
<tr>
<td>Fact S</td>
<td>-</td>
</tr>
</tbody>
</table>
Hamrle3 (1.447M) 64 partitions on 32 cores - MUMPS_TRUNK+ES

<table>
<thead>
<tr>
<th></th>
<th>BC (blk. size = 4)</th>
<th>ABCD (size S = 54608)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fact.</td>
<td>2.13 s.</td>
<td>2.7s.</td>
</tr>
<tr>
<td>CG (615itr)</td>
<td>282.90 s.</td>
<td>-</td>
</tr>
<tr>
<td>Augmentation</td>
<td>-</td>
<td>4.6s.</td>
</tr>
<tr>
<td>Build S</td>
<td>-</td>
<td>145s. → 58.4s.</td>
</tr>
<tr>
<td>Fact S</td>
<td>-</td>
<td>43.1s.</td>
</tr>
</tbody>
</table>

Thanks to François-Henry ROUET for the hints on Exploit-Sparsity feature (available in the future release of MUMPS)
Thoughts and possible orientations

Main issue: Size of $S$

- sparsity structure (preprocessing, permutations...)
- number and size of partitions
- interconnections between partitions

<table>
<thead>
<tr>
<th>Matrix</th>
<th>N</th>
<th>Pts</th>
<th>Size of $S$</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamrle3</td>
<td>1,447,360</td>
<td>64</td>
<td>54,608</td>
<td>3.8%</td>
</tr>
<tr>
<td>R6</td>
<td>132,106</td>
<td>16</td>
<td>8,536</td>
<td>6.5%</td>
</tr>
<tr>
<td>ohne2</td>
<td>181,343</td>
<td>16</td>
<td>48,920</td>
<td>27%</td>
</tr>
</tbody>
</table>
Thoughts and possible orientations

Main issue: Size of $S$ and its density

- sparsity structure (preprocessing, permutations...)
- number and size of partitions
- interconnections between partitions

<table>
<thead>
<tr>
<th>Matrix</th>
<th>N</th>
<th>Pts</th>
<th>Size of $S$</th>
<th>Ratio</th>
<th>NZ($S$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamrle3</td>
<td>1,447,360</td>
<td>64</td>
<td>54,608</td>
<td>3.8%</td>
<td>90,105,183</td>
</tr>
<tr>
<td>R6</td>
<td>132,106</td>
<td>16</td>
<td>8,536</td>
<td>6.5%</td>
<td>9,836,139</td>
</tr>
<tr>
<td>ohne2</td>
<td>181,343</td>
<td>16</td>
<td>48,920</td>
<td>27%</td>
<td>221,956,502</td>
</tr>
</tbody>
</table>
Thoughts and possible orientations

Main issue: Size of $S$ and its density

- sparsity structure (preprocessing, permutations...)
- number and size of partitions
- interconnections between partitions

Possible solutions

- Relax the augmentation process by reducing the number of columns in $C$ and therefore reduce the size of $S$
- Avoid building $S$ by using implicitly (MV products) in an iterative process (CG, $S$ is SPD)
Relaxation of the augmentation process

Target:

• A reduced size of $S$ with respect to the size of $A$: better control of memory requirements

Issues:

• The augmented partitions $\tilde{A}_i$ lose "partly" their mutual numerical orthogonality
• $(I - P)$ is no longer explicitly available, and must be recovered via an iterative process
Relaxation of the augmentation process

**Target:**
- A reduced size of $S$ with respect to the size of $A$: better control of memory requirements

**Issues:**
- The augmented partitions $\bar{A}_i$ lose "partly" their mutual numerical orthogonality
- $(I - P)$ is no longer explicitly available, and must be recovered via an iterative process

**Results on bayer01**

<table>
<thead>
<tr>
<th>Drop threshold</th>
<th>0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>BC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size of $S$</strong></td>
<td>752</td>
<td>270</td>
<td>77</td>
<td>46</td>
<td>18</td>
<td>-</td>
</tr>
<tr>
<td>$w = \bar{A}^+b$</td>
<td>1</td>
<td>103</td>
<td>282</td>
<td>466</td>
<td>1183</td>
<td>1700</td>
</tr>
<tr>
<td>AVG. iter per column</td>
<td>1</td>
<td>17</td>
<td>45</td>
<td>64</td>
<td>340</td>
<td>-</td>
</tr>
<tr>
<td>Total iterations to build $S$</td>
<td>752</td>
<td>4590</td>
<td>3465</td>
<td>2944</td>
<td>6130</td>
<td>-</td>
</tr>
</tbody>
</table>
Target:
• A reduced size of $S$ with respect to the size of $A$ : better control of memory requirements

Issues:
• The augmented partitions $\tilde{A}_i$ lose "partly" their mutual numerical orthogonality
• $(I - P)$ is no longer explicitly available, and must be recovered via an iterative process

In general :
+ Build a reduced size $S$
+ Outperforms regular Block-Cimmino after a few successive solves (in the bayer01’s case after 15 solves with a drop of 0.3)
- Slower to build (less parallel advantages + iterations)
Iterative solution of $Sz = f$

Recall that $S = Y(I - P)Y^T$ is SPD, therefore the system $Sz = f$ can be solved using CG.
Iterative solution of $Sz = f$

Recall that $S = Y(I - P)Y^T$ is SPD, therefore the system $Sz = f$ can be solved using CG. In the CG iteration, $S$ is used implicitly in the instruction:

$$\alpha_k = \left( r_k^T r_k \right) / \left( p_k^T Sp_k \right)$$
Iterative solution of $Sz = f$

Recall that $S = Y(I - P)Y^T$ is SPD, therefore the system $Sz = f$ can be solved using CG. In the CG iteration, $S$ is used implicitly in the instruction:

$$\alpha_k = \left( r_k^T r_k \right) / \left( p_k^T Sp_k \right)$$

The matrix-vector product can be written as:

$$Sp_k = Y(I - P)Y^T p_k$$

$$= p_k - YPY^T p_k$$
Iterative solution of $S z = f$

Recall that $S = Y(I - P)Y^T$ is SPD, therefore the system $S z = f$ can be solved using CG. In the CG iteration, $S$ is used implicitly in the instruction:

$$\alpha_k = \left( r_k^T r_k \right) / \left( p_k^T S p_k \right)$$

The matrix-vector product can be written as:

$$S p_k = Y(I - P)Y^T p_k$$
$$= p_k - YPY^T p_k$$

Where $Y^T p_k = \begin{bmatrix} 0 \\ p_k \end{bmatrix}$ is to be projected by solving augmented systems using MUMPS.
Iterative solution of $Sz = f$ : Test

- Conjugate Gradient acceleration with stopping criteria $1 \times 10^{-8}$.
- A testing (rudimentary) preconditioner (partial build of $S$)
Iterative solution of $Sz = f$ : Test

- Conjugate Gradient acceleration with stopping criteria $1 \times 10^{-8}$.
- A testing (rudimentary) preconditioner (partial build of $S$)

<table>
<thead>
<tr>
<th></th>
<th>size($S$)</th>
<th>CG</th>
<th>PCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>bayer01</td>
<td>752</td>
<td>F</td>
<td>257</td>
</tr>
<tr>
<td>Hamrle3</td>
<td>54,608</td>
<td>1,911</td>
<td>1,238</td>
</tr>
<tr>
<td>R6</td>
<td>8,536</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>ohne2</td>
<td>48,920</td>
<td>32,301</td>
<td>8,066</td>
</tr>
</tbody>
</table>
The R6 case

Smallest eig. = 2.95252e−12; Largest eig. = 1.00000e+00

Eigenvalue distribution of S
The good side:
- Building $S$ is fast (and could be faster), thanks MUMPS!
- ABCD solves block-Cimmino convergence issues

The issues:
- $S$ can be really large and dense (problem dependent)
- Reducing the size of $S$ can perform better than block-Cimmino in the long run.
- Iteratively solving $Sz = f$ is not there yet. Still looking for a preconditioner.