A Hybrid Solver Based on Block-Cimmino Method

Distributed implementation of the Conjugate Gradient accelerated Block Cimmino Method

I.S. Duff   R. Guivarch   D. Ruiz   Mohamed Zenadi,

MUMPS User Days - May 29-30th, 2013

The Hybrid Block Cimmino

Block Cimmino: The basic facts

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

Partitioning the system $Ax = b$

$$\begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{pmatrix} x = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{pmatrix}$$

Block Cimmino: The basic facts

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

The Block Cimmino Iteration

$$
\begin{aligned}
\delta_i^{(k)} &= A_i^{+} b_i - P_{\mathcal{R}(A_i^T)} x^{(k)} \\
x^{(k+1)} &= x^{(k)} + \nu \sum_{i=1}^{p} \delta_i^{(k)}
\end{aligned}
$$

where: $\boxed{A_i^{+} = A_i^T \left(A_i A_i^T\right)^{-1}}$ and $\boxed{P_{\mathcal{R}(A_i^T)} = A_i^{+} A_i}$

Block Cimmino: The basic facts

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

Acceleration

Apply CG to solve the SPD system

$$\sum_{i=1}^{p} A_i^+ A_i \, x = \sum_{i=1}^{p} A_i^+ b_i$$

Block Cimmino: The basic facts

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

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 $\sum_{i=1}^{p} A_i^{+} A_i = \sum_{i=1}^{p} P_{\mathcal{R}(A_i^T)}$

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

Solve independently using `MUMPS` the systems for each partition

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

$$\text{where : } u_i = A_i^{+}(b_i - A_i x) = \delta_i$$

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 exploit up to 3 levels of parallelism (independency, sparsity structure, BLAS3 Kernels)

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

Solve independently using `MUMPS` the systems for each partition

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

$$\text{where : } u_i = A_i^+ (b_i - A_i x) = \delta_i$$

Sequential Case: Computing projections

In sequential:

Sequential Case: Computing projections

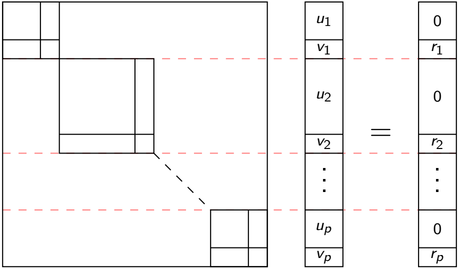
In sequential:

- Computes all the projections at once

Sequential Case: Computing projections

In sequential:

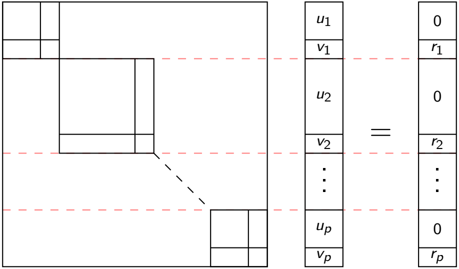
- Computes all the projections at once
- Build a block diagonal system of augmented system



Sequential Case: Computing projections

In sequential:

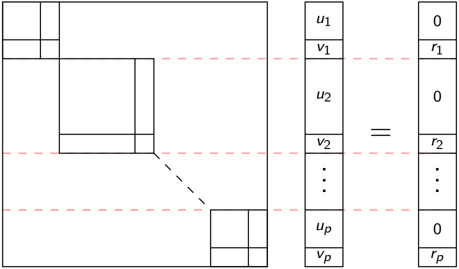
- Computes all the projections at once
- Build a block diagonal system of augmented system
- Analyse + Factorize then solve using `MUMPS`



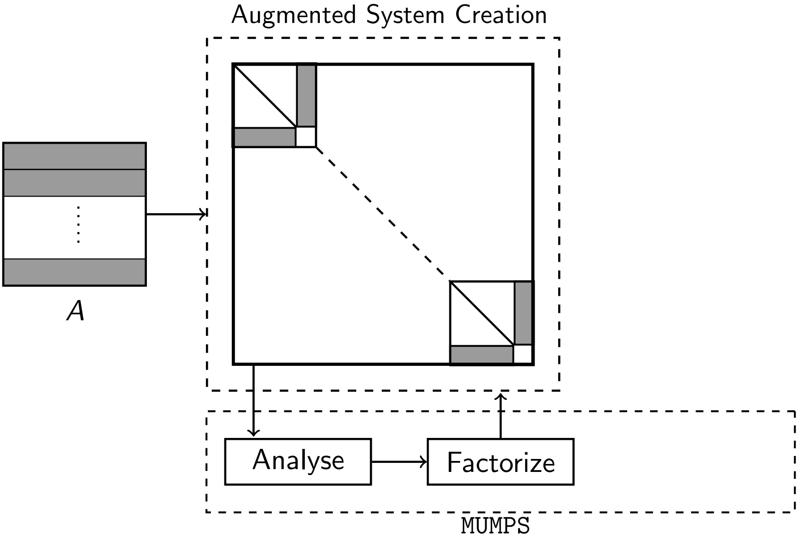
Sequential Case: Computing projections

In sequential:

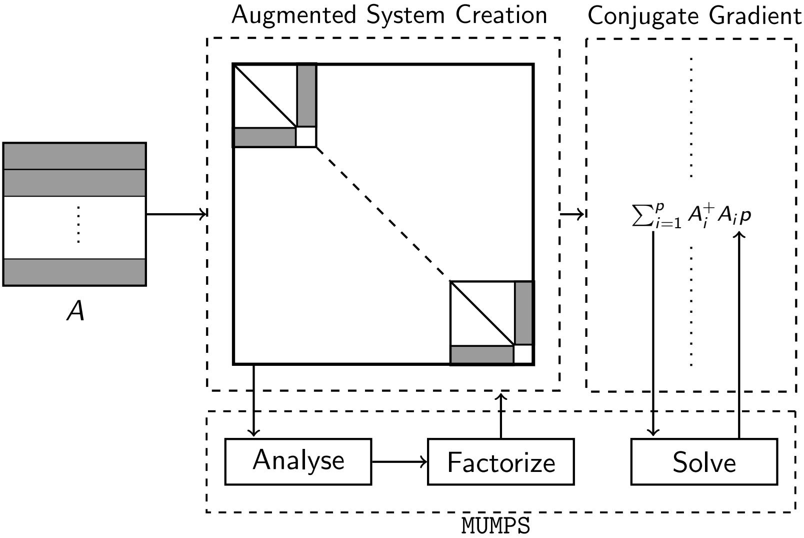
- Computes all the projections at once
- Build a block diagonal system of augmented system
- Analyse + Factorize then solve using `MUMPS`
- Exploits the Forest structure



Sequential workflow

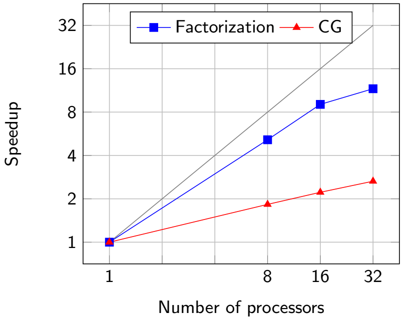


Sequential workflow



`torso3` $259{,}156 \times 259{,}156$ NZ(4,429,042)

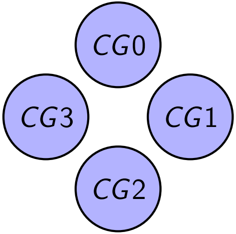
16 partitions 33 iterations



CPU	CG	Fact.
1	46.73s.	21.28.
8	9.09s.	11.66s.
16	5.16s.	9.57s.
32	4.02s.	8.03s.

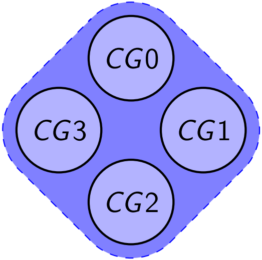
Distributed Scheme

- Distributed Conjugate Gradient Acceleration



Distributed Scheme

- Distributed Conjugate Gradient Acceleration

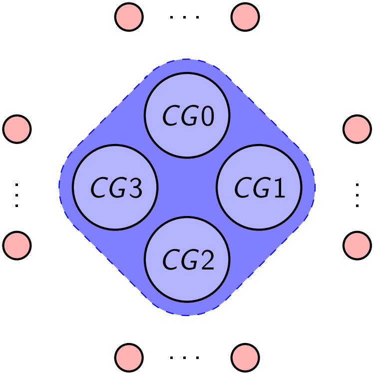


Important Notice

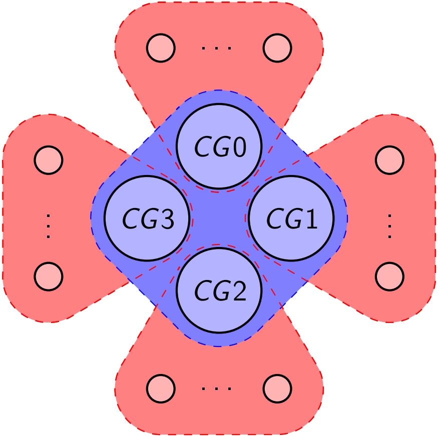
There is only a single Block-CG distributed over these processes.

Distributed Scheme

- Distributed Conjugate Gradient Acceleration

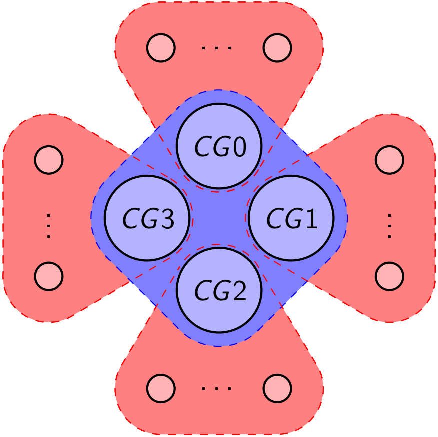


- Distributed Conjugate Gradient Acceleration
- Multiple levels of parallelism

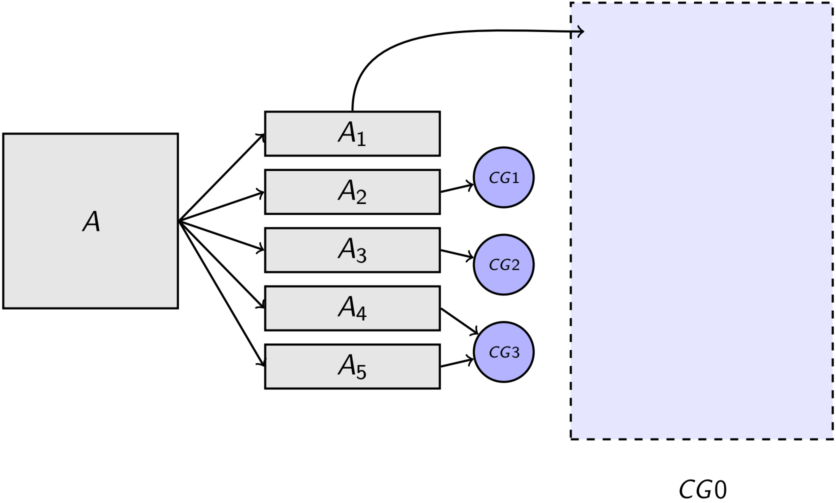


Distributed Scheme

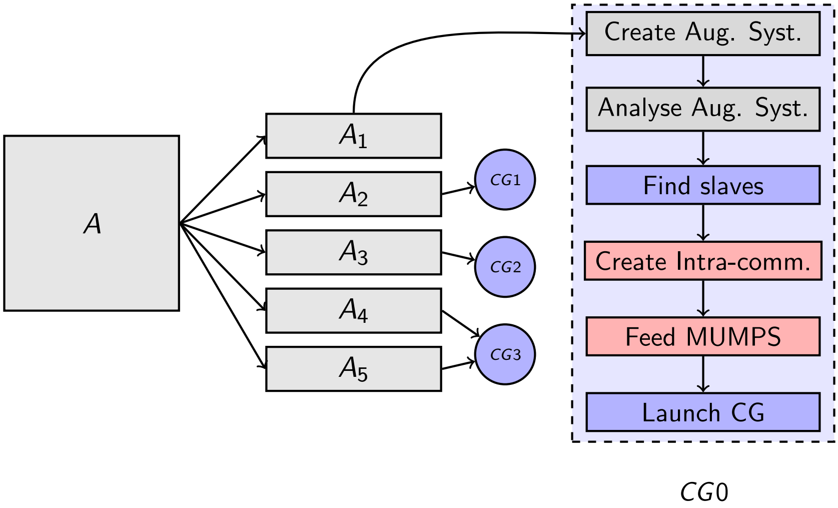
- Distributed Conjugate Gradient Acceleration
- Multiple levels of parallelism
- Forest simulation



Distributed Scheme



Distributed Scheme

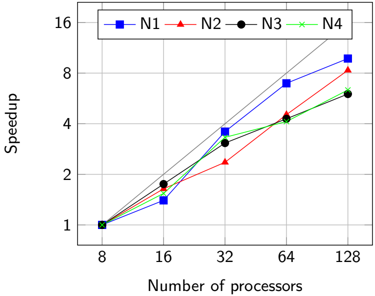


Parallelism results: Distributed B-CG

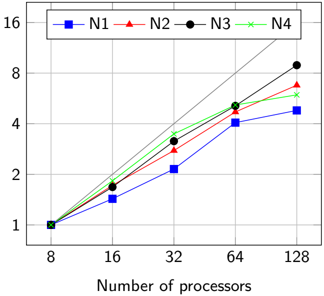
Problem	Size	Nonzeros	Application
N_1: torso3	259,156	4,429,042	3D model of torso
N_2: CoupCons3D	416,800	17,277,420	structural problem
N_3: cage13	445,315	7,479,343	DNA electrophoresis
N_4: Hamrle3	1,447,360	5,514,242	Circuit Simulation

- Runs made on `Hyperion - CICT`
- 3.2Ghz Intel Xeon Quad-Core CPUs (2 per node)
- 36GB per node

Parallelism results: Distributed B-CG



(a) Factorization speedup



(b) B-CG speedup

Problem	Partitions	Factorization at 8 Cores	B-CG at 8 cores
N_1	16	12.11 s.	6.34 s.
N_2	32	14.41 s.	63.49 s.
N_3	256	28.15 s.	13.64 s.
N_4	64	6.67 s.	773 s.

Block Cimmino vs MUMPS

32 cores machine with 500GB of memory [Conan - ENSEEIHT]

Block Cimmino vs MUMPS

32 cores machine with 500GB of memory [Conan - ENSEEIHT]

Problem	[MUMPS] Factorization	[BC] Factorization	B-CG
torso3	5.03 s.	3.21 s.	5.18 s.
Cage13	1452.44 s.	9.31 s.	3.18 s.
Hamrle3	413.21 s.	2.13 s.	282.90 s.

The reality: On most test cases, `MUMPS` does better!
However:

- BC breaks the complexity down so that the factorization goes faster
- BC consumes less memory:
  - `MUMPS + Cage13` : MAX : 6.7GB / AVG : 4.0GB
  - `BC + Cage13` : MAX : 685MB / AVG : 263MG

A path to orthogonality

Issues

- Slow convergence
- Unpredictable convergence behaviour (usually plateaux based)
- Multiple solves requires a re-run of B-CG (too expensive)

A path to orthogonality

Issues

- Slow convergence
- Unpredictable convergence behaviour (usually plateaux based)
- Multiple solves requires a re-run of B-CG (too expensive)

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

- Slow convergence
- Unpredictable convergence behaviour (usually plateaux based)
- Multiple solves requires a re-run of B-CG (too expensive)

Solution

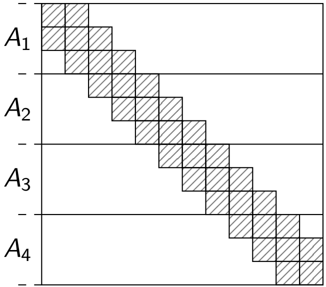
- Enforce numerical orthogonality between partitions by adding extra variables and constraints
- Extract a condensed smaller subsystem (similar to Schur complement techniques) that can be reused for efficient further solves

⇒ **A**ugmented **B**lock **C**immino **D**istributed solver (`ABCD solver`)

Augmented Block Cimmino Distributed solver

The augmentation process

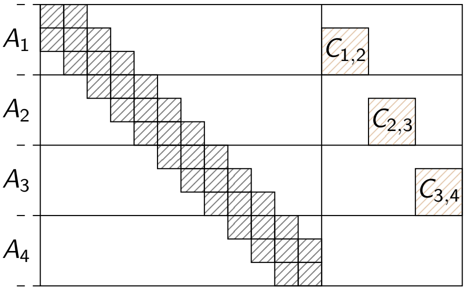
- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)



Illustrative example

The augmentation process

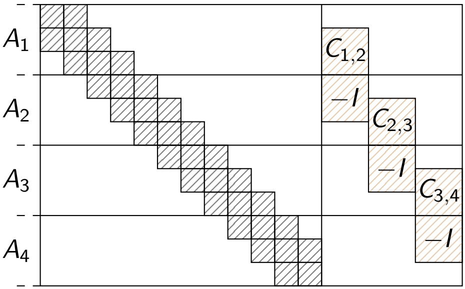
- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions ($j > i$), expand with $C_{i,j} = A_i A_j^T$,



`Illustrative example`

The augmentation process

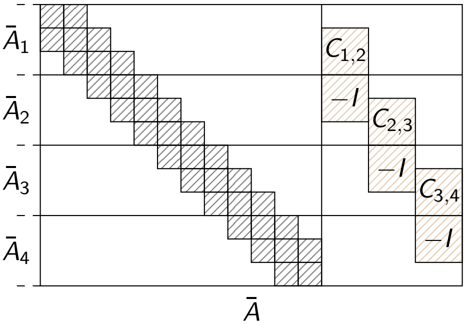
- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions ($j > i$), expand with $C_{i,j} = A_i A_j^T$, and enforce numerical orthogonality



`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 $C_{i,j} = A_i A_j^T$, and enforce numerical orthogonality to obtain $\bar{A} = \begin{bmatrix} A & C \end{bmatrix}$



`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 $C_{i,j} = A_i A_j^T$, and enforce numerical orthogonality to obtain $\bar{A} = \begin{bmatrix} A & C \end{bmatrix}$
- 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 $\bar{A} = \begin{bmatrix} A & C \end{bmatrix}$
- 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 $Y = \begin{bmatrix} 0 & I \end{bmatrix}$, linked to the constraints equations, is not orthogonal to the previous partitions in $\bar{A} = \begin{bmatrix} A & C \end{bmatrix}$.

The augmentation process

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

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

where (as a result of the enforced orthogonality) :

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

The augmentation process

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

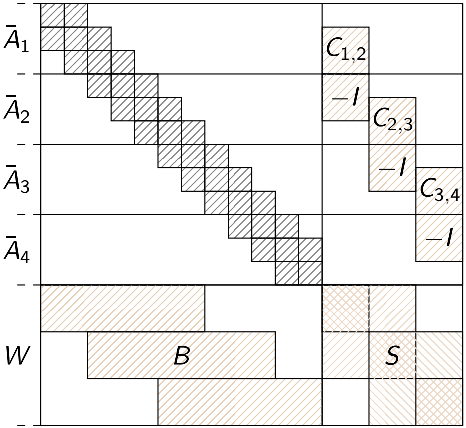
$$W^T = (I - P)\, Y^T,$$

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 $\begin{bmatrix} A & C \\ B & S \end{bmatrix}$, where $\begin{bmatrix} B & S \end{bmatrix} = W$, 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 :

$$f = \begin{bmatrix} B & S \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix} = Y\,(I - P) \begin{bmatrix} x \\ 0 \end{bmatrix}$$

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 :

$$\begin{aligned} f &= \begin{bmatrix} B & S \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix} = Y(I-P) \begin{bmatrix} x \\ 0 \end{bmatrix} \\ &= -YP \begin{bmatrix} x \\ 0 \end{bmatrix} \qquad \left(\text{since } Y = \begin{bmatrix} 0 & I \end{bmatrix}\right) \\ &= -Y\bar{A}^{+}\bar{A} \begin{bmatrix} x \\ 0 \end{bmatrix} \\ f &= -Y\bar{A}^{+}b \end{aligned}$$

Implicit Direct Solver

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 :

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix} &= \bar{A}^{+} b + W^{+} f \\ &= \bar{A}^{+} b - W^{+} Y \bar{A}^{+} b \\ &= \textstyle\sum_{i=1}^{p} \bar{A}_i^{+} b_i - W^{+} Y \sum_{i=1}^{p} \bar{A}_i^{+} b_i \end{aligned}$$

Computational Ingredients

Knowing that $W = \begin{bmatrix} B & S \end{bmatrix} = Y\,(I - P)$, with $Y = \begin{bmatrix} 0 & I \end{bmatrix}$,
we have :

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

Therefore $\boxed{S = Y\,(I-P)\,Y^T}$ and is SPD
and the pseudo inverse $W^+ = W^T(WW^T)^{-1}$ is given by

$$\begin{aligned} W^+ &= W^T S^{-1} \\ W^+ &= (I-P)\,Y^T S^{-1} \end{aligned}$$

Computational Ingredients

The solution is thus given by :

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

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 :

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 $f = -Yw$

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 $\quad Sz = f \quad$ (S should be small enough)

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 $\quad Sz = f \quad$ (S should be small enough)
- Expand z and then project it onto the null space of $\bar{A}$ viz.
$$u = (I - P)\, Y^{T} z$$

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

Current and Future work

- The Augmented Block Cimmino Distributed Method

- The Augmented Block Cimmino Distributed Method

Hamrle3 (1.447M) 64 partitions on 32 cores

	BC (blk. size = 4)	ABCD (size S = 54608)
Fact.	**2.13 s.**	**3.4s.**
CG	(615itr) **282.90 s.**	-
Augmentation	-	**4.6s.**
Build S	-	**145.4s.**
Ana. S	-	**5.6s.**
Fact S	-	**49.1s.**

Current and Future work

- The Augmented Block Cimmino Distributed Method

R6 (132k) 16 partitions on 32 cores

	BC (blk. size = 16)	ABCD (size $S =$ 6506)
Fact.	0.2 s.	0.2s.
CG	(521itr) 107.6 s.	-
Augmentation	-	0.16s.
Build S	-	5.5s.
Ana. S	-	0.3s.
Fact S	-	1.2s.

Current and Future work

- The Augmented Block Cimmino Distributed Method

bmw3_2 (227k) 16 partitions on 32 cores

	BC (blk. size = 1)	ABCD (size $S = 16695$)
Fact.	1.7 s.	1.97s.
CG	(Failed) 176.5 s.	-
Augmentation	-	0.6s.
Build S	-	40.0s.
Ana. S	-	0.2s.
Fact S	-	18.0s.

Current and Future work

- The Augmented Block Cimmino Distributed Method

Current and Future work

- The Augmented Block Cimmino Distributed Method **[on it!]**
- Iteratively and implicitly solve $S^{-1}f$ **[under investigation]**
- More parallelism improvements **[todo]**
  - Distributed hypergraph partitioning (Zoltan)
  - Distributed input matrix
  - Improve parallel scaling
  - etc.

Current and Future work

- The Augmented Block Cimmino Distributed Method **[on it!]**
- Iteratively and implicitly solve $S^{-1}f$ **[under investigation]**
- More parallelism improvements **[todo]**
  - Distributed hypergraph partitioning (Zoltan)
  - Distributed input matrix
  - Improve parallel scaling
  - etc.

Thank you!