Università degli Studi di Padova

DICEA – Department of Civil, Environmental and Architectural Engineering

PhD SCHOOL – Civil and Environmental Engineering Sciences
XXX Cycle

A robust multilevel approximate inverse preconditioner for symmetric positive definite matrices

Andrea Franceschini, Victor Magri, Massimiliano Ferronato and Carlo Janna

February 16th, 2017

Introduction

Algebraic preconditioner

- Our main goal is solving real world problems arising from the **discretization of PDE** in various fields of application
- **Algebraic preconditioners**: robust tools which can be used knowing the coefficient matrix only, regardless of the specific problem addressed
- Incomplete factorizations, algebraic multigrid, **sparse approximate inverses** (FSAI)

FSAI preconditioner for SPD linear systems

- Factorized Sparse Approximate Inverse (FSAI): an almost perfectly parallel factored preconditioner [Kolotilina and Yeremin, 1993]

$$A^{-1} = F^T F$$

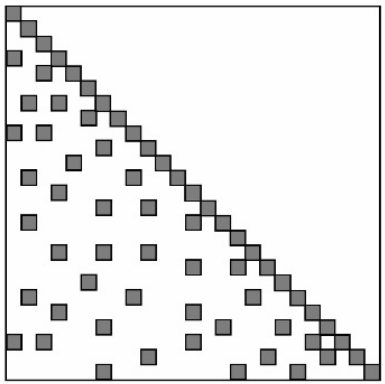
with F a lower triangular matrix such that:

$$\left\| I - FL \right\|_F \to \min$$

over the set of matrices with a prescribed lower triangular sparsity pattern S_L where L is the exact Cholesky factor of A ⟹ L is not actually required for computing F!

- Computed via the solution of n independent small dense systems and applied via matrix-vector products
- Nice features: (1) ideally perfect parallel construction and application of the preconditioner; (2) preservation of the positive definiteness of the native matrix

S_L



MFLR preconditioner

- One of the main difficulties stems from the selection of S_L as an a priori sparsity pattern for F
- Using small powers of A is a popular choice, but for difficult problems high powers may be needed and the preconditioner construction can become quite heavy
- A most efficient option relies on **selecting the pattern dynamically** by an adaptive procedure which uses somewhat the "best" available positions for the non-zero coefficients
- The procedure is based on the minimization of the Kaporin conditioning number **k** of an SPD matrix which is defined as:

$$\kappa(A) = \frac{\mathrm{tr}(A)}{n \det(A)^{1/n}}$$

- It can be shown that this procedure, though expensive, has some optimality properties and gives satisfactorily results for SPD systems

FSAI drawbacks

- The drawback of static FSAI is the choice of the sparsity pattern. The cost increases with the **third power** of the preconditioner density.

$$c \propto n \cdot \overline{m}_G^3 = n \cdot \overline{m}_A^3 \mu_G^3$$

- The adaptive FSAI overcomes the choice of the pattern, but the cost increases with the **fourth power** of the preconditioner density.

$$c = \sum_{i=1}^{n} c_i \propto n \cdot k_{iter}^4 = n \cdot \overline{m}_A^4 \mu_G^4$$

Multilevel approaches for FSAI preconditioning

Basic concepts

- To improve the quality of the FSAI preconditioner we borrow the scheme of the incomplete factorization

… and …

- To reduce the cost of the dynamic selection of the pattern (the set where we look for best coefficients) in the adaptive FSAI algorithm, we developed two new approaches:
  - **Block tridiagonal FSAI**
  - **Domain Decomposition FSAI**: a particular case of the former, with n = 2. Blocks are defined as internal unknowns (1st block) and interface unknowns (2nd block) after a **Domain Decomposition** reordering

Block tridiagonal FSAI

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_1 & & & \\ \mathbf{B}_1^T & \mathbf{A}_2 & \mathbf{B}_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{B}_{n-2}^T & \mathbf{A}_{n-1} & \mathbf{B}_{n-1} \\ & & & \mathbf{B}_{n-1}^T & \mathbf{A}_n \end{bmatrix}$$

Factorization in the LDL^T form (Cholesky-like)

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_1 & & & \\ \mathbf{B}_1^T\mathbf{S}_1^{-1} & \mathbf{I}_2 & & \\ & \ddots & \ddots & \\ & & \mathbf{B}_{n-1}^T\mathbf{S}_{n-1}^{-1} & \mathbf{I}_n \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 & & & \\ & \mathbf{S}_2 & & \\ & & \ddots & \\ & & & \mathbf{S}_n \end{bmatrix} \begin{bmatrix} \mathbf{I}_1 & \mathbf{S}_1^{-1}\mathbf{B}_1 & & \\ & \mathbf{I}_2 & \mathbf{S}_2^{-1}\mathbf{B}_2 & \\ & & \ddots & \ddots \\ & & & \mathbf{I}_n \end{bmatrix}$$

KEY ROLE

Computation of the Schur complement

$$\mathbf{S}_i = \mathbf{A}_i - \mathbf{B}_{i-1}^T\mathbf{S}_{i-1}^{-1}\mathbf{B}_{i-1}, \quad i = 2 \ldots n$$

$$\mathbf{S}_{i-1}^{-1} \approx \mathbf{F}_{i-1}^T\mathbf{F}_{i-1}$$

The inverse of S is factorized with the adaptive FSAI, implemented in FSAIPACK [*Janna et al.*, 2015]

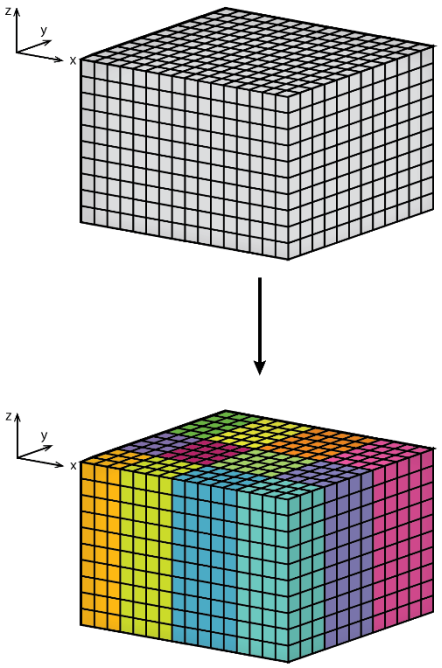
Domain Decomposition FSAI

Following the previous idea, we split the matrix into 2x2 block structure, where the two blocks have different size. Use of domain decomposition technique. Note that this is a particular case of the Block Tridiagonal with 2 blocks.

Internal unknowns Interface unknowns

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_1 \\ \mathbf{B}_1^T & \mathbf{A}_2 \end{bmatrix} = \begin{bmatrix} (\mathbf{A}_1)_1 & & & & (\mathbf{B}_1)_1 \\ & (\mathbf{A}_1)_2 & & & (\mathbf{B}_1)_2 \\ & & \ddots & & \vdots \\ & & & (\mathbf{A}_1)_n & (\mathbf{B}_1)_n \\ (\mathbf{B}_1^T)_1 & (\mathbf{B}_1^T)_2 & \dots & (\mathbf{B}_1^T)_n & \mathbf{A}_2 \end{bmatrix}$$

In this case, we need to compute just one Schur complement.



Domain Decomposition Example

Algorithm to compute and to apply the preconditioner

Algorithm 1 Construction of the BTFSAI preconditioner

procedure CPTBTFSAI($\mathbf{A}, n$)
  Compute $\mathbf{A}_1^{-1} \approx \mathbf{F}_1^T \mathbf{F}_1$
  for $i = 2, \ldots, n$ **do**
    Compute $\mathbf{H}_{i-1} = \mathbf{F}_{i-1}\mathbf{B}_{i-1}$
    Compute $\mathbf{S}_i = \mathbf{A}_i - \mathbf{H}_{i-1}^T \mathbf{H}_{i-1}$
    Compute $\mathbf{S}_i^{-1} = \mathbf{F}_i^T \mathbf{F}_i$
  end for
end procedure

Sequential framework

Parallel phases

S can be indefinite
BREAKDOWN

Algorithm 2 Calculation of $u = \mathbf{M}^{-1} v$

procedure APPLYBTFSAI($\mathbf{F}_i$,$\mathbf{H}_i$,v)
  $w_1 = \mathbf{F} v_1$
  for $i = 2, \ldots, n$ **do**
    $w_i = \mathbf{F}_i \left(v_i - \mathbf{H}_{i-1}^T w_{i-1}\right)$
  end for
  $u_n = \mathbf{F}_n^T w_n$
  for $i = n-1, \ldots, 1$ **do**
    $u_i = \mathbf{F}_i^T \left(w_i - \mathbf{H}_i u_{i+1}\right)$
  end for
end procedure

Sequential framework

Parallel phases

Parallel phases

Results

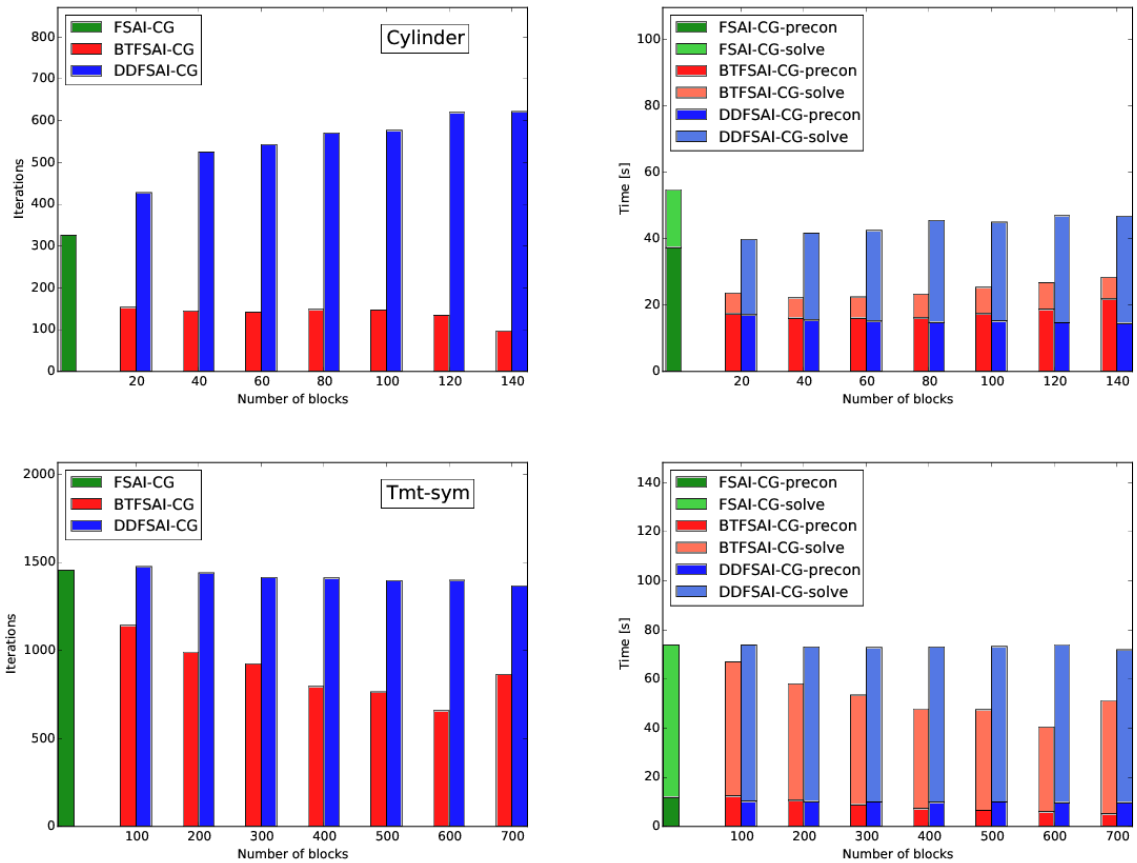
Solver: CG

Exit tolerance: 10^{-8}

CPU: 1 core

Iterations

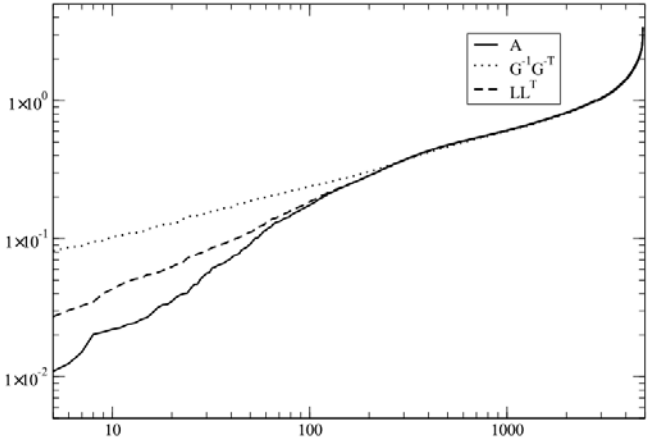
CPU time



Robust Multilevel approaches for FSAI preconditioning

Basic concepts

- The previous implementation of the multilevel is very prone to **breakdowns**: Schur complement approximation is the difference of two SPD matrices and can be **indefinite**. This happens also in relatively well-conditioned problems.
- The reason for this probably resides in the better approximation of the lower eigenvalues usually offered by ILU over FSAI.
- These issues motivate the development of a multilevel preconditioner which is more connected to the FSAI methodology.



Comparison between the spectra of A, LL^T and G^{-1} G^{-T} for the *bcsstk16* matrix from the University of Florida sparse matrix collection

Schur complement computation

- The matrix is subdivided into 4 blocks $\mathbf{A} = \begin{bmatrix} \mathbf{K} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix}$
- The adaptive FSAI matrix **G** of the (1,1) block **K** is computed, so the first phase of preconditioning is done.

$$\left(\mathbf{G}^T\mathbf{G}\right)^{-1} \simeq \mathbf{K} \Longrightarrow \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{K} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix}\begin{bmatrix} \mathbf{G}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{GKG}^T & \mathbf{GB} \\ \mathbf{B}^T\mathbf{G}^T & \mathbf{C} \end{bmatrix} \simeq \begin{bmatrix} \mathbf{I} & \mathbf{GB} \\ \mathbf{B}^T\mathbf{G}^T & \mathbf{C} \end{bmatrix}$$

- The adaptive block FSAI matrix **F** to decouple diagonal blocks is computed. With this also the second phase of preconditioning is done and the Schur complement is defined.

$$\mathbf{F} \simeq -\left(\mathbf{GB}\right)^T\left(\mathbf{GKG}^T\right)^{-1}$$

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{F} & \mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{GKG}^T & \mathbf{GB} \\ \mathbf{B}^T\mathbf{G}^T & \mathbf{C} \end{bmatrix}\begin{bmatrix} \mathbf{I} & \mathbf{F}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{GKG}^T & \mathbf{G}\left(\mathbf{KG}^T\mathbf{F}^T + \mathbf{B}\right) \\ \left(\mathbf{FGK} + \mathbf{B}^T\right)\mathbf{G}^T & \mathbf{C} + \mathbf{FGB} + \mathbf{B}^T\mathbf{G}^T\mathbf{F}^T + \mathbf{FGKG}^T\mathbf{F}^T \end{bmatrix} \simeq \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{S}} \end{bmatrix}$$

Approximate Schur complement that becomes new matrix A (multilevel)

Algorithm to compute the MFLR preconditioner

Multilevel **F**SAI preconditioner with **L**ow-**R**ank corrections

Algorithm 2.2 Multilevel FSAI Set-up

1. Set $A_0 = A$;
2. **for all** $l = 0, \ldots, n_l - 2$ **do**
3. Partition A_l as $\begin{bmatrix} K & B \\ B^T & C \end{bmatrix}$;
4. Compute the adaptive FSAI approximation G of K such that $G^T G \simeq K^{-1}$;
5. Set $P_a = \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix}$;
6. Compute $P_b = \begin{bmatrix} I & 0 \\ F & I \end{bmatrix}$ the adaptive Block FSAI approximation of $P_a A_l P_a^T$;
7. Compute $\widetilde{S} = C + FGB + B^T G^T F^T + FGKG^T F^T$;
8. Form $M_l^{-1} = P_b P_a$;
9. Set $A_{l+1} = \widetilde{S}$;
10. **end for**
11. Compute $M_{n_l-1}^{-1}$ as the adaptive FSAI approximation of A_{n_l-1};
12. Set $M^{-1} = \{M_0^{-1}, M_1^{-1}, \ldots, M_{nl-1}^{-1}\}$;

Sequential framework

Parallel phases

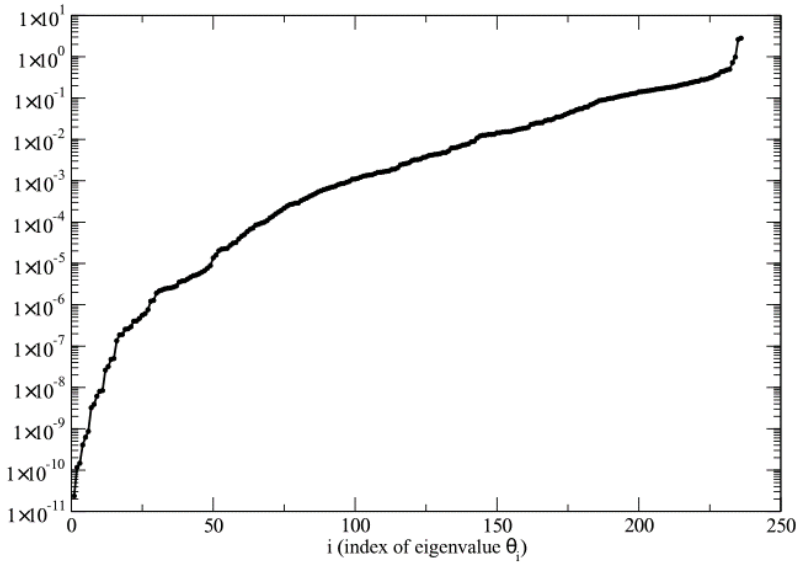
S is SPD
NO BREAKDOWN

Main Schur complement property

The Schur complement is an SPD matrix, indeed, with some algebra:

$$\tilde{\mathbf{S}} = \mathbf{C} + \mathbf{FGB} + \mathbf{B}^T\mathbf{G}^T\mathbf{F}^T + \mathbf{FGKG}^T\mathbf{F}^T = \underbrace{\mathbf{C} - \mathbf{B}^T\mathbf{K}^{-1}\mathbf{B}}_{\text{Real Schur complement (SPD)}} + \mathbf{W}^T\mathbf{KW}$$

where: $\mathbf{W} = \mathbf{K}^{-1}\mathbf{B} + \mathbf{G}^T\mathbf{F}^T$



Spectrum of $\tilde{\mathrm{S}} - \mathrm{S}$ matrix for the *bcsstk38* matrix from the University of Florida sparse matrix collection

Low-Rank corrections

To improve the precondition quality, two different low rank corrections are developed:

- Descending Low-Rank: locally improvement of adaptive FSAI of (1,1) block
- Ascending Low-Rank: global improvement of adaptive FSAI of Schur complement [*Xi et al.*, 2016]

Both ideas are based on follows observations:

- An approximation **G** for the matrix **A**, such that $\mathbf{I}-\mathbf{GAG}^T=\mathbf{H}\rightarrow\mathbf{0}$, is available
- We want to improve $\mathbf{A}^{-1}$ as $\mathbf{A}^{-1}=\mathbf{G}^T\mathbf{G}+\mathbf{G}^T\mathbf{XG}$
- Matrices **X** and **H** are linked by the simple relation: $\mathbf{X}=\mathbf{H}\left[\mathbf{I}-\mathbf{H}\right]^{-1}$
- To reduce ill-conditioning, eigenvalues of **A** that we want to correct are ones close to 0, that are eigenvalues of **H** close to 1. For these values, eigenvalues of **X** tends to infinity
- Due to this fact, just a low-rank representation of matrix **H** is needed. Moreover, computation of eigenvalues of matrix **X** (the correction) is straightforward.

Low-Rank corrections

H matrix is symmetric, so we can write its eigen-decomposition (where Σ is diagonal):

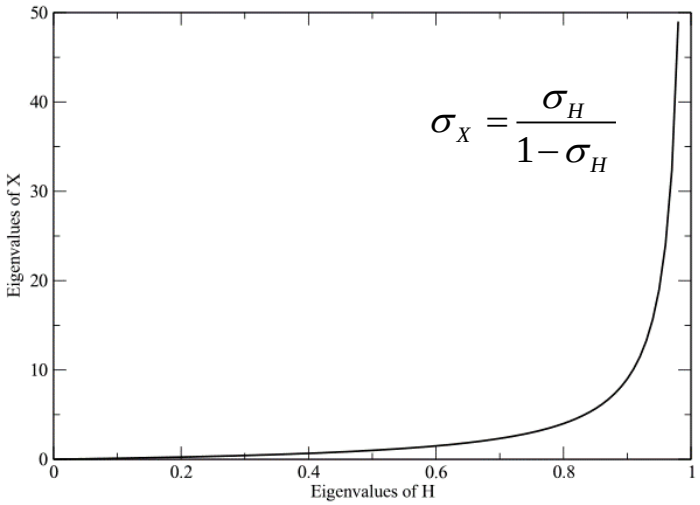
$$\mathbf{H} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T$$

X matrix is linked to **H**, so we have:

$$\mathbf{X} = \mathbf{U}\left[\boldsymbol{\Sigma}\left(\mathbf{I}-\boldsymbol{\Sigma}\right)^{-1}\right]\mathbf{U}^T$$

For each eigenvalue of **H**, eigenvalue of **X** is:

$$\sigma_X = \frac{\sigma_H}{1-\sigma_H}$$



Eigenvalues of correction on the inverse (X matrix) with respect to the eigenvalues of the direct correction (H matrix)

A small number of eigenvalues gives rise to a big correction on the approximate inverse. Moreover, it can be shown that the Low-Rank corrections do not propagate along the levels.

Low-Rank target

Target of Ascending Low Rank is to make the available factorized inverse more similar to the inverse of one of two matrices:

- approximate Schur complement $\tilde{\mathbf{S}} = \mathbf{C} + \mathbf{FGB} + \mathbf{B}^T\mathbf{G}^T\mathbf{F}^T + \mathbf{FGKG}^T\mathbf{F}^T$
- real Schur complement $\mathbf{S} = \mathbf{C} - \mathbf{B}^T\mathbf{K}^{-1}\mathbf{B}$
- Eigenvalue bound with approximate Schur complement as target:

$$|\lambda - 1| \leq \frac{\epsilon_1 + \sqrt{\epsilon_1^2 + 4\tilde{\eta}_1^2}}{2} \qquad (1)$$

- Eigenvalue bound with real Schur complement as target:

$$|\lambda - 1| \leq \frac{\epsilon_1 + \eta_1^2\kappa_n^{-1} + \sqrt{\left(\epsilon_1 - \eta_1^2\kappa_n^{-1}\right)^2 + 4\eta_1^2}}{2} \qquad (2)$$

- It can be shown that bound (1) is narrower or equal to the bound (2).

$$\|E_K\|_2 = \lambda_1(E_K) = \epsilon_1, \quad \|Q\|_2 = \sqrt{\lambda_1(Q^TQ)} = \eta_1, \quad \|\widetilde{Q}\|_2 = \sqrt{\lambda_1(\widetilde{Q}^T\widetilde{Q})} = \tilde{\eta}_1, \quad \|(I + E_K)^{-1}\|_2 = \lambda_n^{-1}(GKG^T) = \kappa_n^{-1}$$

MFLR preconditioner

Algorithm 4.1 MFLR Set-up

1. **Recursive Function** MFLR_SETUP(l,A)
2. **if** $l < (n_l - 1)$ **then**
3. Partition A_l as $\begin{bmatrix} K & B \\ B^T & C \end{bmatrix}$;
4. Compute the adaptive FSAI approximation G of K such that $G^T G \simeq K^{-1}$;
5. Set $P_a = \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix}$;
6. Compute $P_b = \begin{bmatrix} I & 0 \\ F & I \end{bmatrix}$ as the adaptive Block FSAI approximation of $P_a A_l P_a^T$;
7. Compute $\widetilde{S} = C + FGB + B^T G^T F^T + FGKG^T F^T$;
8. Compute a rank k approximation $\overline{U}_k \overline{\Sigma}_k \overline{U}_k^T$ of $\overline{Y} = I - GKG^T$;
9. Set $\widetilde{G} = (I + \overline{U}_k \overline{\Psi}_k \overline{U}_k^T)G$ with $\overline{\Psi}_k = (I - \overline{\Sigma}_k)^{-1/2} - I$;
10. Set $P = \begin{bmatrix} \widetilde{G} & 0 \\ FG & I \end{bmatrix}$;
11. $[Q_{alr}, Q_M] =$ MFLR_SETUP($l+1$,$\widetilde{S}$);
12. Use Q_{alr} and Q_M to compute the rank k correction $\widehat{W}_k$ and $\widehat{\Theta}_k$ as in (3.23);
13. Set $\widetilde{P} = \begin{bmatrix} I & 0 \\ 0 & I + \widehat{W}_k \widehat{\Theta}_k \widehat{W}_k^T \end{bmatrix}$;
14. Push $\widetilde{P}$ in the head of Q_{alr};
15. Push P in the head of Q_M;
16. **return** Q_{alr}, Q_M;
17. **else**
18. Compute the adaptive FSAI approximation G of A such that $G^T G \simeq A$;
19. Set $Q_{alr} = \emptyset$;
20. Set $Q_M = \{G\}$;
21. **return** Q_{alr}, Q_M;
22. **end if**

Algorithm to compute the MFLR preconditioner (with Low Rank Updates)

Descending Low Rank correction

Ascending Low Rank correction

MFLR preconditioner

Algorithm to apply the MFLR preconditioner (with Low Rank Updates)

Algorithm 4.2 MFLR Apply

1. **Recursive Function** MFLR_APPLY(l,Q_{alr},Q_M,x)
2. **if** $l < (n_l - 1)$ **then**
3. Pop P from the head of Q_M;
4. Pop $\widetilde{P}$ from the head of Q_{alr};
5. Compute $y = Px$;
6. Partition $y = \{y_1^T,\ y_2^T\}^T$;
7. Compute $z_2 = \text{MFLR\_APPLY}(l+1, Q_{alr}, Q_M, y_2)$;
8. Compose $z = \{y_1^T,\ z_2^T\}^T$;
9. Compute $y = \widetilde{P}z$;
10. Compute $y = P^T y$;
11. Push P in the head of Q_M;
12. Push $\widetilde{P}$ in the head of Q_{alr};
13. **return** y;
14. **else**
15. Pop G from the head of Q_M;
16. Compute $y = G^T G x$;
17. Push G in the head of Q_M;
18. **return** y;
19. **end if**

Descending Low Rank correction

Ascending Low Rank correction

Descending Low Rank correction

Results: test case

We extensively tested *Cube* matrix, that has 190,581 rows and 7,531,389 non-zeroes

- Increasing the number of levels, number of iterations needed to reach convergence always decreases, but iteration time increases
- With Low-Rank corrections we can reduce both the number of iterations and the iteration time. In particular, the most promising technique is the Ascending Low-Rank

levels	iter	ρ	T_P	T_S	T_T
1	686	0.363	8.49	14.44	22.93
10	640	0.774	34.45	21.87	56.32
20	591	1.173	265.92	29.30	295.22
50	503	2.038	441.67	41.62	483.30
100	464	3.674	612.91	66.96	679.87

Results for *Cube* matrix. All parameters are fixed, expect the number of levels

rank	iter	ρ	T_{DLR}	T_P	T_S	T_T
0	648	0.696	0.00	30.84	22.64	53.48
5	539	0.807	9.44	39.66	23.27	62.93
10	484	0.918	11.73	42.44	21.89	64.34
20	445	1.139	16.76	46.42	21.66	68.09
50	397	1.803	36.20	66.05	24.40	90.46

Results for *Cube* matrix. All parameters are fixed, expect the Descending Low-Rank size

rank	iter	ρ	T_{ALR}	T_P	T_S	T_T
0	648	0.696	0.00	30.84	22.64	53.48
5	329	1.139	32.42	62.31	16.24	78.56
10	241	1.582	53.75	82.94	14.07	97.01
20	222	2.467	93.03	123.62	14.53	138.16
50	206	5.125	372.02	401.15	33.58	434.73

Results for *Cube* matrix. All parameters are fixed, expect the Ascending Low-Rank size

Results: test case

- Combining together Descending and Ascending Low-Rank we reach the best result

Desc. rank	Asc. rank	iter	ρ	T_{DLR}	T_{ALR}	T_P	T_S	T_T
1	1	562	0.807	8.09	21.45	57.88	29.55	87.43
1	10	228	1.604	8.41	45.83	82.60	13.20	95.79
1	20	212	2.490	8.77	92.26	130.01	13.87	143.88
10	1	393	1.006	10.79	23.08	62.35	22.25	84.59
10	10	163	1.803	10.89	48.41	87.75	10.37	98.13
10	20	147	2.689	12.01	93.58	134.93	10.63	145.56
20	1	357	1.227	16.93	26.31	72.42	22.11	94.53
20	10	145	2.025	16.91	54.96	101.03	9.84	110.87
20	20	133	2.910	15.62	91.69	135.93	9.89	145.82

Results for *Cube* matrix. Descending Low-Rank and Ascending Low-Rank are combined together

Results: real world problems

We tested some matrices from the University of Florida Sparse Matrix Collection.

	Size	Number of non-zeroes
af_shell3	504,855	17,588,875
af_shell8	504,855	17,579,155
Emilia_923	923,136	40,373,538
Geo_1438	1,437,960	60,236,322
StocF_1465	1,465,137	21,005,389

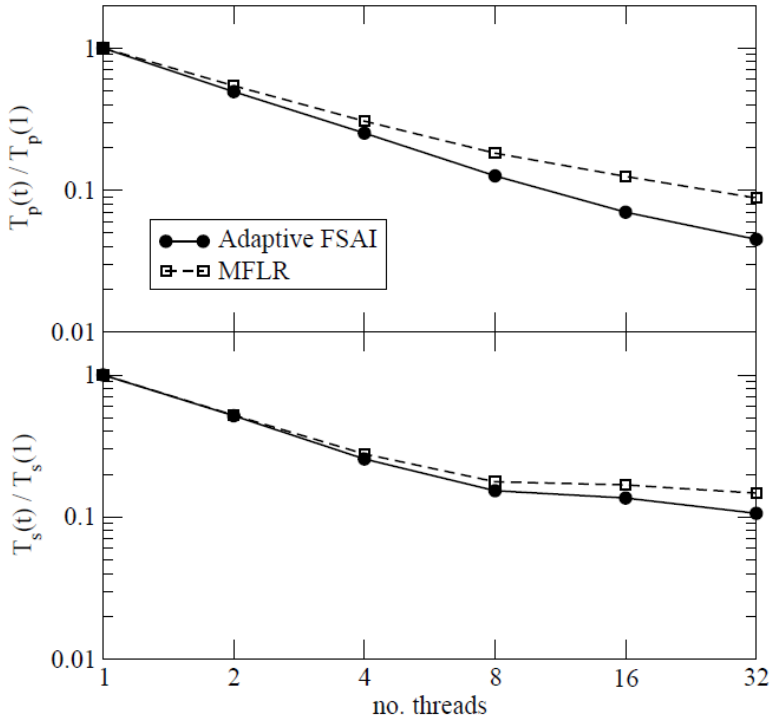
Test matrices (from UF Sparse Matrix Collection)

	Adaptive FSAI				MFLR			
	n_{it}	ρ	T_p [s]	T_s [s]	n_{it}	ρ	T_p [s]	T_s [s]
af_shell3	963	0.89	88.8	63.9	126	4.50	237.8	31.5
af_shell8	1033	0.86	81.2	68.0	131	3.22	174.8	28.6
Emilia_923	1513	0.11	5.0	128.5	575	0.25	82.8	80.4
Geo_1423	491	0.31	59.4	89.6	456	0.49	159.8	110.4
StocF_1465	937	0.93	51.3	133.4	936	1.66	260.4	222.2

Computational performance of the adaptive FSAI and MFLR preconditioners for the test matrices (different configurations)

Results: strong scalability

With a Laplacian 300 x 300 x 300 we tested strong scalability



Strong scalability test for a regular 300^3 Laplacian

We used MARCONI cluster at the CINECA Center of High Performance Computing, Bologna. It has 1,512 nodes. Every node has 128 Gbyte of RAM memory and is equipped with 2 Intel Xeon E5-2697v4 Broadwell processors at 2.3 GHz with 36 cores.

Conclusions

- The use of FSAI in the framework of multilevel preconditioner may arise some difficulties, as indefinite Schur complements. With the MFLR preconditioner, Schur complements are ensured to be **positive definite**.
- The multilevel FSAI preconditioner is further enhanced by introducing Low-Rank corrections at both a local and a global level, namely **Descending** and **Ascending Low-Rank corrections**, respectively.
- Some theoretical properties and bounds on eigenvalues can be computed for the case with just two levels.

Further research

- To explore various degrees of sparsity in matrix products to reduce the cost of preconditioner set-up (matrix sparsification, matrix compression, etc.).

Università degli Studi di Padova

DICEA – Department of Civil, Environmental and Architectural Engineering

Thank you for your attention