Aggregation-based algebraic multigrid

A tutorial

Yvan Notay*

Université Libre de Bruxelles
Service de Métrologie Nucléaire

* Supported by the Belgian FNRS

http://homepages.ulb.ac.be/∼ynotay

Outline

ULB

1. Quick overview of Multigrid
2. Two-grid convergence theory
3. Geometric MG and AMG
4. Aggregation-based AMG
5. The K-cycle
6. Quality aware aggregation
7. Performance
8. Parallelization of AGMG
9. Conclusions

1. Multigrid: quick overview (3)

ULB

A coarse grid correction for the system

$$A\,\mathbf{u} \;=\; \mathbf{b} \quad\Longleftrightarrow\quad A\,(\mathbf{u}-\widetilde{\mathbf{u}}) \;=\; \mathbf{b}-A\,\widetilde{\mathbf{u}} \;=\; \widetilde{\mathbf{r}}$$

(or $L^h(\mathbf{u}^h-\widetilde{\mathbf{u}}^h) \;=\; \mathbf{f}^h - L_h\widetilde{\mathbf{u}}^h$) consists in

- Transfer the problem on the coarse grid

  Restriction R ($n_c \times n$):

  $$\mathbf{r}_c \;=\; R\,(\mathbf{b}-A\,\widetilde{\mathbf{u}})$$

  $$(\,\mathbf{f}^H \;=\; I_h^H(\mathbf{f}^h - L_h\widetilde{\mathbf{u}}^h)\,)$$

1. Multigrid: quick overview (3)

ULB

A coarse grid correction for the system

$$A\,\mathbf{u} \;=\; \mathbf{b} \quad\Longleftrightarrow\quad A\,(\mathbf{u}-\widetilde{\mathbf{u}}) \;=\; \mathbf{b}-A\,\widetilde{\mathbf{u}} \;=\; \widetilde{\mathbf{r}}$$

(or $L^h(\mathbf{u}^h-\widetilde{\mathbf{u}}^h) \;=\; \mathbf{f}^h - L_h\widetilde{\mathbf{u}}^h$) consists in

- Transfer the problem on the coarse grid

  Restriction R ($n_c \times n$):

  $$\mathbf{r}_c \;=\; R\,(\mathbf{b}-A\,\widetilde{\mathbf{u}})$$

  $$(\,\mathbf{f}^H \;=\; I_h^H(\mathbf{f}^h - L_h\widetilde{\mathbf{u}}^h)\,)$$

- Solve the coarse problem

  Coarse grid matrix A_c ($n_c \times n_c$):

  solve $A_c\,\mathbf{u}_c \;=\; \mathbf{r}_c$

  $$(\,A_H\mathbf{v}^H \;=\; \mathbf{f}^H\,)$$

1. Multigrid: quick overview (3)

ULB

A coarse grid correction for the system

$$A\,\mathbf{u} \;=\; \mathbf{b} \quad\Longleftrightarrow\quad A\,(\mathbf{u}-\widetilde{\mathbf{u}}) \;=\; \mathbf{b}-A\,\widetilde{\mathbf{u}} \;=\; \widetilde{\mathbf{r}}$$

(or $L^h(\mathbf{u}^h-\widetilde{\mathbf{u}}^h) \;=\; \mathbf{f}^h - L_h\widetilde{\mathbf{u}}^h$) consists in

- Transfer the problem on the coarse grid

  Restriction R ($n_c \times n$):

  $$\mathbf{r}_c \;=\; R\,(\mathbf{b}-A\,\widetilde{\mathbf{u}})$$
  $$(\,\mathbf{f}^H \;=\; I_h^H(\mathbf{f}^h - L_h\widetilde{\mathbf{u}}^h)\,)$$

- Solve the coarse problem

  Coarse grid matrix A_c ($n_c \times n_c$):

  solve $\quad A_c\,\mathbf{u}_c \;=\; \mathbf{r}_c$
  $$(\,A_H\mathbf{v}^H \;=\; \mathbf{f}^H\,)$$

- Interpolate (prolong) the computed coarse solution

  Prolongation P ($n \times n_c$):

  $$\widetilde{\widetilde{\mathbf{u}}} \;=\; \widetilde{\mathbf{u}} + P\,\mathbf{u}_c$$
  $$(\,\widetilde{\widetilde{\mathbf{u}}}^h \;=\; \widetilde{\mathbf{u}}^h + I_H^h\,\mathbf{v}^H\,)$$

1. Multigrid: quick overview (3)

ULB

A coarse grid correction for the system

$$A\,\mathbf{u} = \mathbf{b} \iff A\,(\mathbf{u} - \widetilde{\mathbf{u}}) = \mathbf{b} - A\,\widetilde{\mathbf{u}} = \widetilde{\mathbf{r}}$$

(or $L^h(\mathbf{u}^h - \widetilde{\mathbf{u}}^h) = \mathbf{f}^h - L_h\widetilde{\mathbf{u}}^h$) consists in

- Transfer the problem on the coarse grid

  Restriction R ($n_c \times n$):

  $$\mathbf{r}_c = R\,(\mathbf{b} - A\,\widetilde{\mathbf{u}})$$
  $$(\,\mathbf{f}^H = I_h^H(\mathbf{f}^h - L_h\widetilde{\mathbf{u}}^h)\,)$$

- Solve the coarse problem

  Coarse grid matrix A_c ($n_c \times n_c$):

  solve $A_c\,\mathbf{u}_c = \mathbf{r}_c$
  $(\,A_H\mathbf{v}^H = \mathbf{f}^H\,)$

- Interpolate (prolong) the computed coarse solution

  Prolongation P ($n \times n_c$):

  $$\widetilde{\widetilde{\mathbf{u}}} = \widetilde{\mathbf{u}} + P\,\mathbf{u}_c$$
  $$(\,\widetilde{\widetilde{\mathbf{u}}}^h = \widetilde{\mathbf{u}}^h + I_H^h\,\mathbf{v}^H\,)$$

Thus: $\widetilde{\widetilde{\mathbf{u}}} = \widetilde{\mathbf{u}} + P\,A_c^{-1}\,R\,(\mathbf{b} - A\,\widetilde{\mathbf{u}})$

1. Multigrid: quick overview (4)

ULB

Basic Two-grid iteration for the system

$$A\,\mathbf{u} \;=\; \mathbf{b} \quad \Longleftrightarrow \quad A\,(\mathbf{u}-\mathbf{u}_k) \;=\; \mathbf{b}-A\,\mathbf{u}_k \;=\; \mathbf{r}_k$$

- Apply ν_1 relaxation iterations to $\mathbf{u}_k$, yielding $\widetilde{\mathbf{u}}_k$

  Error propagation: $\mathbf{u}-\widetilde{\mathbf{u}}_k = S_1^{\nu_1}(\mathbf{u}-\mathbf{u}_k)$

  $S_1 = I - M_1^{-1}A$, where M_1 is the used preconditioner

1. Multigrid: quick overview (4)

ULB

Basic Two-grid iteration for the system

$$A\,\mathbf{u} \;=\; \mathbf{b} \quad \Longleftrightarrow \quad A\,(\mathbf{u}-\mathbf{u}_k) \;=\; \mathbf{b}-A\,\mathbf{u}_k \;=\; \mathbf{r}_k$$

- Apply ν_1 relaxation iterations to $\mathbf{u}_k$, yielding $\widetilde{\mathbf{u}}_k$

  Error propagation: $\mathbf{u}-\widetilde{\mathbf{u}}_k = S_1^{\nu_1}(\mathbf{u}-\mathbf{u}_k)$

  $S_1 = I - M_1^{-1}A$, where M_1 is the used preconditioner

- Apply the coarse grid correction to $\widetilde{\mathbf{u}}_k$, yielding $\widetilde{\widetilde{\mathbf{u}}}_k$

  $\widetilde{\widetilde{\mathbf{u}}} \;=\; \widetilde{\mathbf{u}} + P\,A_c^{-1}\,R\,(\mathbf{b}-A\,\widetilde{\mathbf{u}}) \quad \rightarrow \quad \mathbf{u}-\widetilde{\widetilde{\mathbf{u}}}_k = C(\mathbf{u}-\widetilde{\mathbf{u}}_k)$

  with $C = I - P\,A_c^{-1}\,R\,A$

1. Multigrid: quick overview (4)

ULB

Basic Two-grid iteration for the system

$$A\,\mathbf{u} \;=\; \mathbf{b} \quad \Longleftrightarrow \quad A\,(\mathbf{u}-\mathbf{u}_k) \;=\; \mathbf{b}-A\,\mathbf{u}_k \;=\; \mathbf{r}_k$$

- Apply ν_1 relaxation iterations to $\mathbf{u}_k$, yielding $\widetilde{\mathbf{u}}_k$

  Error propagation: $\mathbf{u}-\widetilde{\mathbf{u}}_k = S_1^{\nu_1}(\mathbf{u}-\mathbf{u}_k)$

  $S_1 = I - M_1^{-1}A$, where M_1 is the used preconditioner

- Apply the coarse grid correction to $\widetilde{\mathbf{u}}_k$, yielding $\widetilde{\widetilde{\mathbf{u}}}_k$

  $\widetilde{\widetilde{\mathbf{u}}} \;=\; \widetilde{\mathbf{u}} + P\,A_c^{-1}\,R\,(\mathbf{b}-A\,\widetilde{\mathbf{u}}) \quad \rightarrow \quad \mathbf{u}-\widetilde{\widetilde{\mathbf{u}}}_k = C(\mathbf{u}-\widetilde{\mathbf{u}}_k)$

  with $C = I - P\,A_c^{-1}\,R\,A$

- Apply ν_2 relaxation iterations to $\widetilde{\widetilde{\mathbf{u}}}_k$, yielding $\mathbf{u}_{k+1}$

  Error propagation: $\mathbf{u}-\mathbf{u}_{k+1} = S_2^{\nu_2}(\mathbf{u}-\widetilde{\widetilde{\mathbf{u}}}_k)$

1. Multigrid: quick overview (4)

ULB

Basic Two-grid iteration for the system

$$A\,\mathbf{u} \;=\; \mathbf{b} \quad \Longleftrightarrow \quad A\,(\mathbf{u}-\mathbf{u}_k) \;=\; \mathbf{b}-A\,\mathbf{u}_k \;=\; \mathbf{r}_k$$

- Apply ν_1 relaxation iterations to $\mathbf{u}_k$, yielding $\widetilde{\mathbf{u}}_k$

  Error propagation: $\mathbf{u}-\widetilde{\mathbf{u}}_k = S_1^{\nu_1}(\mathbf{u}-\mathbf{u}_k)$

  $S_1 = I - M_1^{-1}A$, where M_1 is the used preconditioner

- Apply the coarse grid correction to $\widetilde{\mathbf{u}}_k$, yielding $\widetilde{\widetilde{\mathbf{u}}}_k$

  $\widetilde{\widetilde{\mathbf{u}}} \;=\; \widetilde{\mathbf{u}} + P\,A_c^{-1}\,R\,(\mathbf{b}-A\,\widetilde{\mathbf{u}}) \quad \rightarrow \quad \mathbf{u}-\widetilde{\widetilde{\mathbf{u}}}_k = C(\mathbf{u}-\widetilde{\mathbf{u}}_k)$

  with $C = I - P\,A_c^{-1}\,R\,A$

- Apply ν_2 relaxation iterations to $\widetilde{\widetilde{\mathbf{u}}}_k$, yielding $\mathbf{u}_{k+1}$

  Error propagation: $\mathbf{u}-\mathbf{u}_{k+1} = S_2^{\nu_2}(\mathbf{u}-\widetilde{\widetilde{\mathbf{u}}}_k)$

Globally: $\mathbf{u}-\mathbf{u}_{k+1} \;=\; S_2^{\nu_2}\,C\,S_1^{\nu_1}(\mathbf{u}-\mathbf{u}_k)$

1. Multigrid: quick overview (5)

ULB

From Two- to multi-grid

- solve $A_c \, \mathbf{u}_c = \mathbf{r}_c \rightarrow$ solve approximately $A_c \, \mathbf{u}_c = \mathbf{r}_c$

1. Multigrid: quick overview (5)

ULB

From Two- to multi-grid

- solve $A_c \, \mathbf{u}_c = \mathbf{r}_c \rightarrow$ solve approximately $A_c \, \mathbf{u}_c = \mathbf{r}_c$
- The coarse system is solved with a few iterations, based on the two-grid scheme at that level (referencing thus a further coarser level)
  - 1 iteration: V-cycle
  - 2 iterations: W-cycle

  Thus: the two-grid algorithm is applied recursively

1. Multigrid: quick overview (5)

ULB

From Two- to multi-grid

- solve $A_c \mathbf{u}_c = \mathbf{r}_c \rightarrow$ solve approximately $A_c \mathbf{u}_c = \mathbf{r}_c$
- The coarse system is solved with a few iterations, based on the two-grid scheme at that level (referencing thus a further coarser level)
  - 1 iteration: V-cycle
  - 2 iterations: W-cycle

  Thus: the two-grid algorithm is applied recursively
- Coarsest level: when the number of unknowns is s.t.
  - Solution is trivially obtained ($n = 1$ or so)
  - Relaxation is efficient for all modes
  - A direct solver is cost efficient (w.r.t. the number of fine grid unknowns)

Multigrid as a preconditioner

- The correction $\mathbf{u}_{k+1} - \mathbf{u}_k$ to $\mathbf{u}_k$ is linear w.r.t. the residual $\mathbf{r}_k = \mathbf{b} - A\,\mathbf{u}_k$

1. Multigrid: quick overview (6)

ULB

Multigrid as a preconditioner

- The correction $\mathbf{u}_{k+1} - \mathbf{u}_k$ to $\mathbf{u}_k$ is linear w.r.t. the residual $\mathbf{r}_k = \mathbf{b} - A\,\mathbf{u}_k$
- Hence we may write: $\mathbf{u}_{k+1} = \mathbf{u}_k + \mathcal{B}_{\mathrm{MG}}\mathbf{r}_k$, where $\mathcal{B}_{\mathrm{MG}}$ is a matrix that represents the MG preconditioner

1. Multigrid: quick overview (6)

ULB

Multigrid as a preconditioner

- The correction $\mathbf{u}_{k+1} - \mathbf{u}_k$ to $\mathbf{u}_k$ is linear w.r.t. the residual $\mathbf{r}_k = \mathbf{b} - A\,\mathbf{u}_k$
- Hence we may write: $\mathbf{u}_{k+1} = \mathbf{u}_k + \mathcal{B}_{\mathrm{MG}}\mathbf{r}_k$, where $\mathcal{B}_{\mathrm{MG}}$ is a matrix that represents the MG preconditioner
- MG as a solver: use stationary iterations (seen above)
  MG as a preconditioner: use $\mathcal{B}_{\mathrm{MG}}$ in combination with a Krylov subspace method (CG, GMRES, . . .)

Multigrid as a preconditioner

- The correction $\mathbf{u}_{k+1} - \mathbf{u}_k$ to $\mathbf{u}_k$ is linear w.r.t. the residual $\mathbf{r}_k = \mathbf{b} - A\,\mathbf{u}_k$
- Hence we may write: $\mathbf{u}_{k+1} = \mathbf{u}_k + \mathcal{B}_{\rm MG}\mathbf{r}_k$, where $\mathcal{B}_{\rm MG}$ is a matrix that represents the MG preconditioner
- MG as a solver: use stationary iterations (seen above)
  MG as a preconditioner: use $\mathcal{B}_{\rm MG}$ in combination with a Krylov subspace method (CG, GMRES, . . .)
- Computation of $\mathcal{B}_{\rm MG}\,\mathbf{r}_k$ for given $\mathbf{r}_k$

  Apply the standard algorithm and accumulate in a vector $\mathbf{w}$ all corrections to $\mathbf{u}_k$ (no need to form $\mathbf{u}_{k+1}$)

  For MG as a solver, one would have $\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{w}$

  $\rightarrow \quad \mathbf{w} = \mathcal{B}_{\rm MG}\,\mathbf{r}_k$

1. Multigrid: quick overview (7)

ULB

	MG as a solver	MG prec. for Conjugate Gradients
	$n_{\text{it}} \approx \frac{\ln(\varepsilon^{-1})}{-\ln \rho}$	$n_{\text{it}} \approx \frac{\sqrt{\kappa}\ \ln(2\,\varepsilon^{-1})}{2} = \frac{\ln(2\,\varepsilon^{-1})}{2\sqrt{1-\rho}}$
$(\varepsilon = 10^{-6})$		
$\rho = 0.1$	$n_{\text{it}} \approx 6$	$n_{\text{it}} \approx 8$
$\rho = 0.2$	$n_{\text{it}} \approx 9$	$n_{\text{it}} \approx 9$
$\rho = 0.5$	$n_{\text{it}} \approx 20$	$n_{\text{it}} \approx 10$
$\rho = 0.8$	$n_{\text{it}} \approx 62$	$n_{\text{it}} \approx 16$

- Classical viewpoint: MG as a solver and target $\rho \approx 0.1$

1. Multigrid: quick overview (7)

ULB

	MG as a solver $n_{\text{it}} \approx \frac{\ln(\varepsilon^{-1})}{-\ln \rho}$	MG prec. for Conjugate Gradients $n_{\text{it}} \approx \frac{\sqrt{\kappa}\ \ln(2\,\varepsilon^{-1})}{2} = \frac{\ln(2\,\varepsilon^{-1})}{2\sqrt{1-\rho}}$
($\varepsilon = 10^{-6}$)		
$\rho = 0.1$	$n_{\text{it}} \approx 6$	$n_{\text{it}} \approx 8$
$\rho = 0.2$	$n_{\text{it}} \approx 9$	$n_{\text{it}} \approx 9$
$\rho = 0.5$	$n_{\text{it}} \approx 20$	$n_{\text{it}} \approx 10$
$\rho = 0.8$	$n_{\text{it}} \approx 62$	$n_{\text{it}} \approx 16$

- Classical viewpoint: MG as a solver and target $\rho \approx 0.1$
- Alternative viewpoint: MG as a preconditioner, and ρ up to 0.8 is not an issue, especially if there are only few relaxation steps while the coarsening is sufficiently fast

1. Multigrid: quick overview (8)

Geometric Multigrid

- The hierarchy of grids comes from the discretization

1. Multigrid: quick overview (8)

ULB

Geometric Multigrid

- The hierarchy of grids comes from the discretization
- Prolongation & restriction based on interpolation between successive grids

1. Multigrid: quick overview (8)

ULB

Geometric Multigrid

- The hierarchy of grids comes from the discretization
- Prolongation & restriction based on interpolation between successive grids
- Very efficient, but not so convenient for industrial applications
  - The relaxation has to be adapted to the application
  - Does not work for unstructured discretization meshes
  - Requires interactions between discretization and solution modules

1. Multigrid: quick overview (9)

ULB

Classical Algebraic Multigrid (AMG)

- Work with the information present in A only

1. Multigrid: quick overview (9)

ULB

Classical Algebraic Multigrid (AMG)

- Work with the information present in A only
- Select a subset of variables as coarse variables by inspecting the (strong) connectivity in A

1. Multigrid: quick overview (9)

ULB

Classical Algebraic Multigrid (AMG)

- Work with the information present in A only
- Select a subset of variables as coarse variables by inspecting the (strong) connectivity in A
- Build interpolation rules from matrix coefficients
  $\rightarrow$ prolongation P

1. Multigrid: quick overview (9)

ULB

Classical Algebraic Multigrid (AMG)

- Work with the information present in A only
- Select a subset of variables as coarse variables by inspecting the (strong) connectivity in A
- Build interpolation rules from matrix coefficients
  $\rightarrow$ prolongation P
- Restriction: $R = P^T$ (most often)

1. Multigrid: quick overview (9)

ULB

Classical Algebraic Multigrid (AMG)

- Work with the information present in A only
- Select a subset of variables as coarse variables by inspecting the (strong) connectivity in A
- Build interpolation rules from matrix coefficients
  $\rightarrow$ prolongation P
- Restriction: $R = P^T$ (most often)
- Coarse grid matrix: $A_c = R\,A\,P = P^T\,A\,P$

  (error in the range of P canceled by the coarse grid correction)

1. Multigrid: quick overview (9)

ULB

Classical Algebraic Multigrid (AMG)

- Work with the information present in A only
- Select a subset of variables as coarse variables by inspecting the (strong) connectivity in A
- Build interpolation rules from matrix coefficients
  $\rightarrow$ prolongation P
- Restriction: $R = P^T$ (most often)
- Coarse grid matrix: $A_c = R\,A\,P = P^T\,A\,P$

  (error in the range of P canceled by the coarse grid correction)
- $\rightarrow$ The coarse grid correction is well defined once P has been built

$$\widetilde{\widetilde{\mathbf{u}}} = \widetilde{\mathbf{u}} + P\,A_c^{-1}\,R\,(\mathbf{b} - A\,\widetilde{\mathbf{u}}) = \widetilde{\mathbf{u}} + P\left(P^T\,A\,P\right)^{-1}P^T(\mathbf{b} - A\,\widetilde{\mathbf{u}})$$

2. Two-grid convergence theory (1)

ULB

Case: A SPD

Approximation property constant

$$K(A, P, D_A) = \sup_{\mathbf{v} \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{\mathbf{v}^T D_A \left(I - P \left(P^T D_A P \right)^{-1} P^T D_A \right) \mathbf{v}}{\mathbf{v}^T A \mathbf{v}}$$

Equivalently, smallest constant K such that

$$\forall \mathbf{v} \in \mathbb{R}^n \; \exists \mathbf{v}_c \in \mathbb{R}^{n_c} \text{ such that } \|\mathbf{v} - P \mathbf{v}_c\|_{D_A}^2 \leq K \|\mathbf{v}\|_A^2$$

(Traces back to [Brandt, 1986])

Can be combined with the smoothing property constant

Conclusion:

P is a good prolongation for A (in the AMG sense)

$\Longleftrightarrow$ $K(A, P, D_A)$ is not large

2. Two-grid convergence theory (2)

ULB

More precisely, for a single Jacobi post-relaxation step:

$\nu_1 = 0$, $\nu_2 = 1$ and $M_2 = \omega^{-1} D_A$; that is

$$S_2^{\nu_2}\, C\, S_1^{\nu_1} \;=\; S_{J_\omega} C \;=\; \left(I - \omega\, D_A^{-1} A\right)\left(I - P(P^T A\, P)^{-1} P^T A\right)$$

- The eigenvalues of $S_{J_\omega} C$ are real

2. Two-grid convergence theory (2)

ULB

More precisely, for a single Jacobi post-relaxation step:

$\nu_1 = 0$, $\nu_2 = 1$ and $M_2 = \omega^{-1} D_A$; that is

$$S_2^{\nu_2}\, C\, S_1^{\nu_1} \;=\; S_{J_\omega} C \;=\; \left(I - \omega\, D_A^{-1} A\right)\left(I - P(P^T A\, P)^{-1} P^T A\right)$$

- The eigenvalues of $S_{J_\omega} C$ are real
- $\lambda_{\max}(S_{J_\omega} C) \;=\; 1 - \frac{\omega}{K(A\,,P_A\,,D_A)}$

2. Two-grid convergence theory (2)

ULB

More precisely, for a single Jacobi post-relaxation step:

$\nu_1 = 0$, $\nu_2 = 1$ and $M_2 = \omega^{-1} D_A$; that is

$$S_2^{\nu_2}\, C\, S_1^{\nu_1} \;=\; S_{J_\omega} C \;=\; \left(I - \omega\, D_A^{-1} A\right)\left(I - P(P^T A\, P)^{-1} P^T A\right)$$

- The eigenvalues of $S_{J_\omega} C$ are real
- $\lambda_{\max}(S_{J_\omega} C) \;=\; 1 - \frac{\omega}{K(A\,,P_A\,,D_A)}$
- $\lambda_{\min}(S_{J_\omega} C) \;\geq\; 1 - \omega\, \lambda_{\max}(D_A^{-1} A)$

2. Two-grid convergence theory (2)

ULB

More precisely, for a single Jacobi post-relaxation step:

$\nu_1 = 0$, $\nu_2 = 1$ and $M_2 = \omega^{-1} D_A$; that is

$$S_2^{\nu_2}\, C\, S_1^{\nu_1} \;=\; S_{J_\omega} C \;=\; \big(I - \omega\, D_A^{-1} A\big)\big(I - P(P^T A\, P)^{-1} P^T A\big)$$

- The eigenvalues of $S_{J_\omega} C$ are real
- $\lambda_{\max}(S_{J_\omega} C) \;=\; 1 - \frac{\omega}{K(A\,,P_A\,,D_A)}$
- $\lambda_{\min}(S_{J_\omega} C) \;\geq\; 1 - \omega\, \lambda_{\max}(D_A^{-1} A)$
- $\rightarrow$
  $\rho(S_{J_\omega} C) \;\leq\; \max\left(1 - \frac{\omega}{K(A\,,P\,,D_A)}\;,\; \omega\, \lambda_{\max}(D_A^{-1} A) - 1\right)$

Nontrivial convergence of a very
basic TG scheme with Jacobi relaxation

$\Longleftrightarrow$ $K(A\,,P\,,D_A)$ is not large

2. Two-grid convergence theory (3)

ULB

Case: A nonsymmetric but positive definite in $\mathbb{R}^n$

$$\mathbf{v}^T A\,\mathbf{v} > 0 \quad \forall \mathbf{v} \in \mathbb{R}^n \backslash \{\mathbf{0}\} \quad \Longleftrightarrow \quad A_S = \tfrac{1}{2}(A + A^T) \text{ is SPD}$$

- The eigenvalues of $S_{J_\omega} C$ are in general complex

2. Two-grid convergence theory (3)

ULB

Case: A nonsymmetric but positive definite in $\mathbb{R}^n$

$$\mathbf{v}^T A\,\mathbf{v} > 0 \quad \forall \mathbf{v} \in \mathbb{R}^n \backslash \{\mathbf{0}\} \quad \Longleftrightarrow \quad A_S = \tfrac{1}{2}(A + A^T) \text{ is SPD}$$

- The eigenvalues of $S_{J_\omega} C$ are in general complex
- For any eigenvalue λ of $S_{J_\omega} C$,

$$\Re\mathrm{e}(\lambda) \leq 1 - \frac{\omega}{K(A_S, P, D_A)}$$

No eigenvalue close to 1 if P is good for A_S

$\rightarrow$ near kernel modes properly damped

2. Two-grid convergence theory (3)

ULB

Case: A nonsymmetric but positive definite in $\mathbb{R}^n$

$$\mathbf{v}^T A\,\mathbf{v} > 0 \quad \forall \mathbf{v} \in \mathbb{R}^n \backslash \{\mathbf{0}\} \quad \Longleftrightarrow \quad A_S = \tfrac{1}{2}(A + A^T) \text{ is SPD}$$

- The eigenvalues of $S_{J_\omega} C$ are in general complex
- For any eigenvalue λ of $S_{J_\omega} C$,

$$\Re\mathrm{e}(\lambda) \leq 1 - \frac{\omega}{K(A_S, P, D_A)}$$

No eigenvalue close to 1 if P is good for A_S

$\rightarrow$ near kernel modes properly damped

- For M-matrices: yields a meaningful bound on $\rho(S_{J_\omega} C)$

Nontrivial convergence of a very basic TG scheme with Jacobi relaxation

$\Longleftrightarrow$ $K(A_S, P, D_A)$ is not large

2. Two-grid convergence theory (4)

ULB

$$\forall \mathbf{v} \in \mathbb{R}^n : \quad K \geq \min_{\mathbf{v}_c \in \mathbb{R}^{n_c}} \frac{\|\mathbf{v} - P\,\mathbf{v}_c\|_{D_A}^2}{\|\mathbf{v}\|_A^2}$$

Let $\mathbf{z}_k$ be such that $D_A^{-1} A\,\mathbf{z}_k = \lambda_k\,\mathbf{z}_k$

Then, $\|\mathbf{z}_k\|_A^2 = \mathbf{z}_k^T A\,\mathbf{z}_k = \lambda_k\,\mathbf{z}_k^T D_A\,\mathbf{z}_k = \lambda_k\,\|\mathbf{z}_k\|_{D_A}^2$

Hence, for any such eigenvector:

$$K \geq \min_{\mathbf{v}_c \in \mathbb{R}^{n_c}} \frac{\|\mathbf{z}_k - P\,\mathbf{v}_c\|_{D_A}^2}{\lambda_k\,\|\mathbf{z}_k\|_{D_A}^2}$$

To keep the approximation property constant K moderate, all eigenvectors associated with small eigenvalues should have a close approximation in the range of P

These are called the algebraically smooth vectors

2. Two-grid convergence theory (5)

Consider the 5 point Finite Difference approximation of

$$-\frac{\partial^2 u}{\partial x^2} - \varepsilon \frac{\partial^2 u}{\partial x^2} = f \qquad \text{in } \Omega = (0,1) \times (0,1)$$

with Dirichlet B.C., using a uniform grid of mesh size $h = 1/N$

The eigenvector of $D_A^{-1} A$ are the vectors sampling the functions

$$\sin(k\,\pi\,x)\sin(\ell\,\pi\,y)\ , \quad k,\ell = 1,\ldots,N$$

and the corresponding eigenvalues are

$$\lambda_{k,\ell} = (1+\varepsilon)^{-1}\left(\left(1-\cos\tfrac{k\,\pi}{N}\right) + \varepsilon\left(1-\cos\tfrac{\ell\,\pi}{N}\right)\right)$$

2. Two-grid convergence theory (6)

ULB

$$\lambda_{k,\ell} \;=\; (1+\varepsilon)^{-1}\Big(\big(1-\cos\tfrac{k\,\pi}{N}\big)+\varepsilon\big(1-\cos\tfrac{\ell\,\pi}{N}\big)\Big)$$

- case $\varepsilon \approx 1$: $\lambda_{k,\ell}$ small $\iff$ $k\,,\,\ell \ll N$

  The algebraically smooth vectors coincide with the geometrically smooth vectors

2. Two-grid convergence theory (6)

ULB

$$\lambda_{k,\ell} \;=\; (1+\varepsilon)^{-1}\Big(\big(1-\cos\tfrac{k\,\pi}{N}\big)+\varepsilon\big(1-\cos\tfrac{\ell\,\pi}{N}\big)\Big)$$

- case $\varepsilon \approx 1$: $\lambda_{k,\ell}$ small $\iff$ $k\,,\,\ell \ll N$

  The algebraically smooth vectors coincide with the geometrically smooth vectors

- case $\varepsilon \ll 1$: $\lambda_{k,\ell}$ small for $k \ll N$ and any ℓ

  The algebraically smooth vectors can be arbitrary oscillatory in the y direction

  $\rightarrow$ The approximation property constant $K(A\,,P\,,D_A)$ is large with usual “geometric” prolongations

3. Geometric MG and AMG (1)

ULB

The approximation property constant $K(A, P, D_A)$ can be large with usual “geometric” prolongations

- Geometric MG philosophy

  K is only meaningful for Jacobi and related relaxations. Thus, large K tells us that we have to seek for a relaxation adapted to the problem at hand.

3. Geometric MG and AMG (1)

ULB

The approximation property constant $K(A, P, D_A)$ can be large with usual "geometric" prolongations

- Geometric MG philosophy

  K is only meaningful for Jacobi and related relaxations. Thus, large K tells us that we have to seek for a relaxation adapted to the problem at hand.

- Classical AMG philosophy

  We want to use standard relaxation in a problem independent manner.

  Thus, the algorithms that define P do not just need to mimic geometric MG for model problems.

  Their main task is to keep $K(A, P, D_A)$ bounded in one way or another.

Consequences for Classical AMG

- Initial idea: mimic geometric multigrid

  Usually slower on model problems
  (the copy cannot do better than the original)

Consequences for Classical AMG

- Initial idea: mimic geometric multigrid

  Usually slower on model problems
  (the copy cannot do better than the original)

- But can be quite different for non model problems

Consequences for Classical AMG

- Initial idea: mimic geometric multigrid

  Usually slower on model problems
  (the copy cannot do better than the original)
- But can be quite different for non model problems
- AMG more robust, in the sense that large classes of problems can be handled by a single software code

Consequences for Classical AMG

- Initial idea: mimic geometric multigrid

  Usually slower on model problems
  (the copy cannot do better than the original)
- But can be quite different for non model problems
- AMG more robust, in the sense that large classes of problems can be handled by a single software code
- Need elaborate algorithms, usually with many options and parameters

Consequences for Classical AMG (cont.)

- As a result, the coarse grid matrix $A_c = P^T A P$ is often denser (more nonzero entries per row) than A

Consequences for Classical AMG (cont.)

- As a result, the coarse grid matrix $A_c = P^T A P$ is often denser (more nonzero entries per row) than A
- Main issue: this effect can be cumulative

  Starting from the fine grid matrix $A = A_1$,
  one defines $P = P_1$ and $A_2 = P_1^T A_1 P_1$

  To obtain the next coarser level,
  one defines P_2 from A_2 , and sets $A_3 = P_2^T A_2 P_2$

  And so on . . .

Consequences for Classical AMG (cont.)

- As a result, the coarse grid matrix $A_c = P^T A P$ is often denser (more nonzero entries per row) than A
- Main issue: this effect can be cumulative

  Starting from the fine grid matrix $A = A_1$,
  one defines $P = P_1$ and $A_2 = P_1^T A_1 P_1$

  To obtain the next coarser level,
  one defines P_2 from A_2 , and sets $A_3 = P_2^T A_2 P_2$

  And so on . . .

- The nightmare of classical AMG: Keep control of the Algorithmic Complexity

$$\mathcal{C}_A = \frac{\sum_{k=1}^{L} nnz(A_k)}{nnz(A_1)}$$

4. Aggregation-based AMG (1)

ULB

Philosophy of (plain or unsmoothed) Aggregation AMG

Keep most of the paradigms of classical AMG

- Pattern of the two-grid scheme:
  relaxation – coarse grid correction – relaxation

4. Aggregation-based AMG (1)

ULB

Philosophy of (plain or unsmoothed) Aggregation AMG

Keep most of the paradigms of classical AMG

- Pattern of the two-grid scheme:
  relaxation – coarse grid correction – relaxation
- Work with the information present in A only:
  - Build the prolongation P from A
  - Restriction: $R = P^T$
  - Coarse grid matrix: $A_c = R\,A\,P = P^T\,A\,P$

4. Aggregation-based AMG (1)

ULB

Philosophy of (plain or unsmoothed) Aggregation AMG

Keep most of the paradigms of classical AMG

- Pattern of the two-grid scheme:
  relaxation – coarse grid correction – relaxation
- Work with the information present in A only:
  - Build the prolongation P from A
  - Restriction: $R = P^T$
  - Coarse grid matrix: $A_c = R\,A\,P = P^T\,A\,P$
- Set up P in such a way that $K(A\,,P\,,D_A)$ is bounded

4. Aggregation-based AMG (1)

ULB

Philosophy of (plain or unsmoothed) Aggregation AMG

Keep most of the paradigms of classical AMG

- Pattern of the two-grid scheme:
  relaxation – coarse grid correction – relaxation
- Work with the information present in A only:
  - Build the prolongation P from A
  - Restriction: $R = P^T$
  - Coarse grid matrix: $A_c = R\,A\,P = P^T\,A\,P$
- Set up P in such a way that $K(A\,,P\,,D_A)$ is bounded
- Proceed recursively

  Set up phase: to define coarser and coarser levels

  Solve phase: approximate solution of coarse systems

4. Aggregation-based AMG (2)

ULB

. . . but

do not try to mimic geometric MG
(even for model problems)

Instead, keep P as simple (sparse) as possible
to keep $K(A, P, D_A)$ bounded

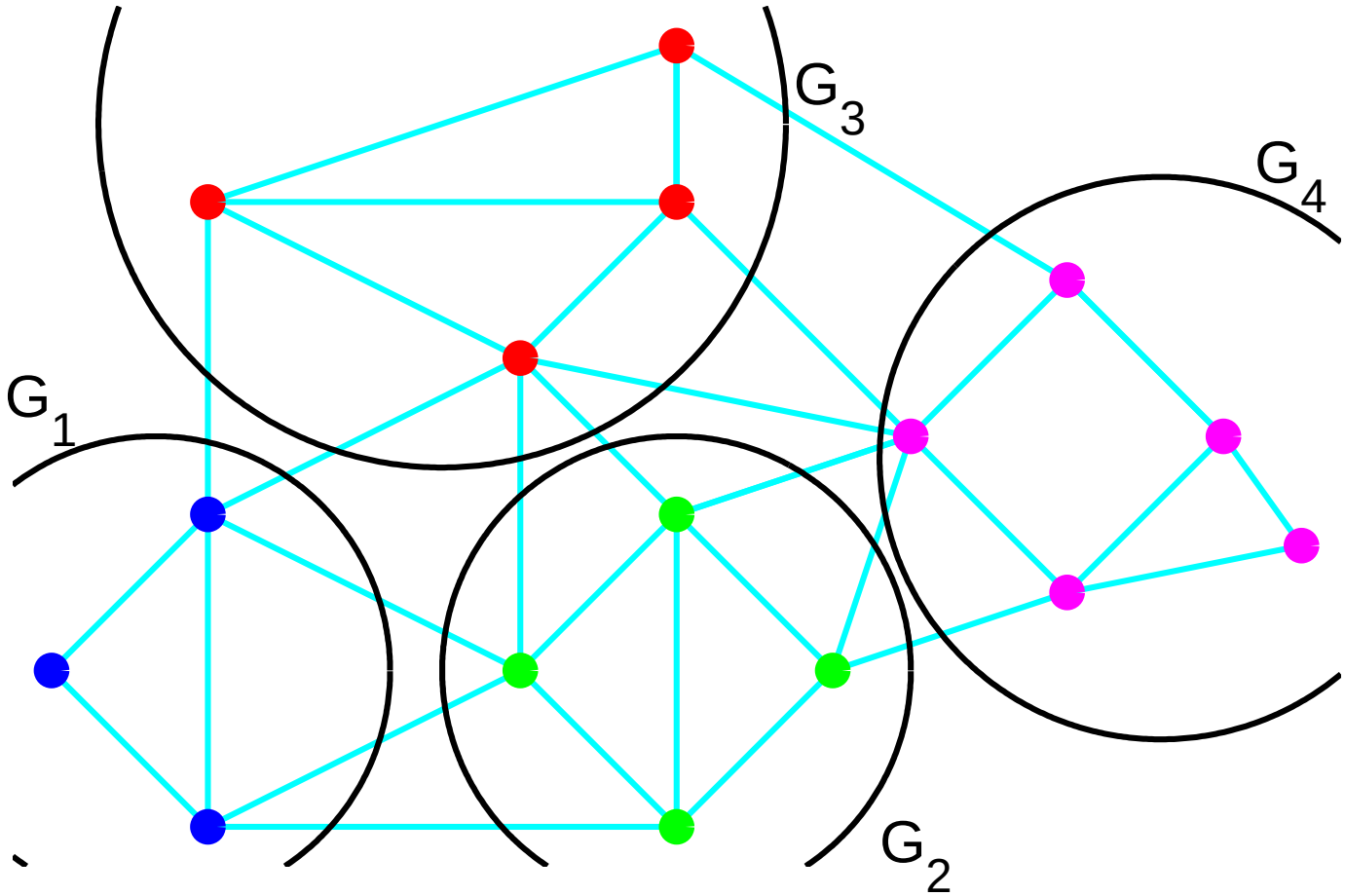
4. Aggregation-based AMG (3)

ULB

Setup phase

To build P , one first group the nodes into aggregates G_J
(The set of G_J form a partitioning of $[1\ ,\ n]$)

Each aggregate G_J corresponds to 1 coarse variable (and vice-versa)



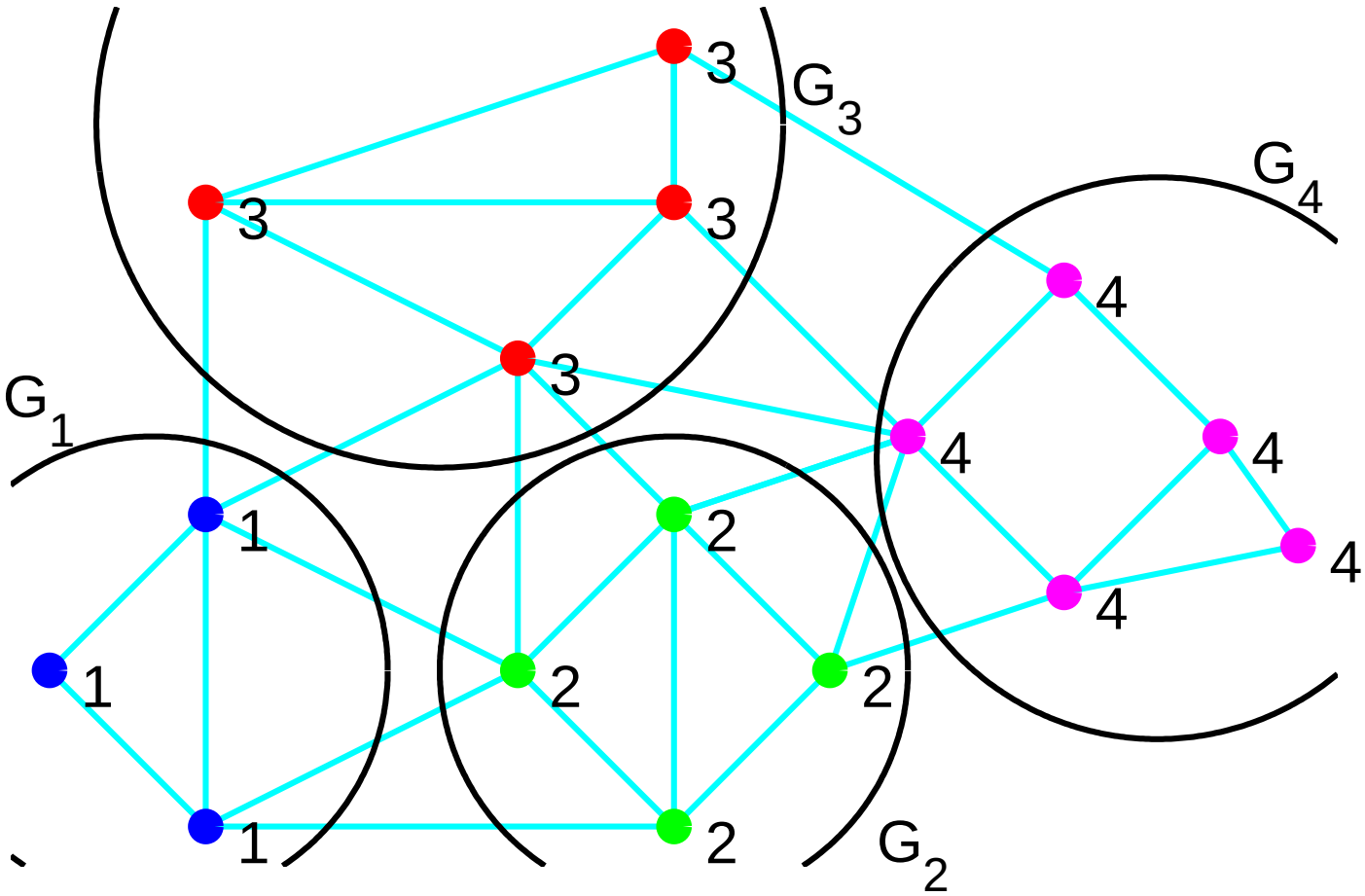
4. Aggregation-based AMG (4)

ULB

Associated prolongation P: $P_{iJ} = \begin{cases} 1 & \text{if } i \in G_J \\ 0 & \text{otherwise} \end{cases}$

Example

$$\mathbf{u}_c = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \rightarrow$$

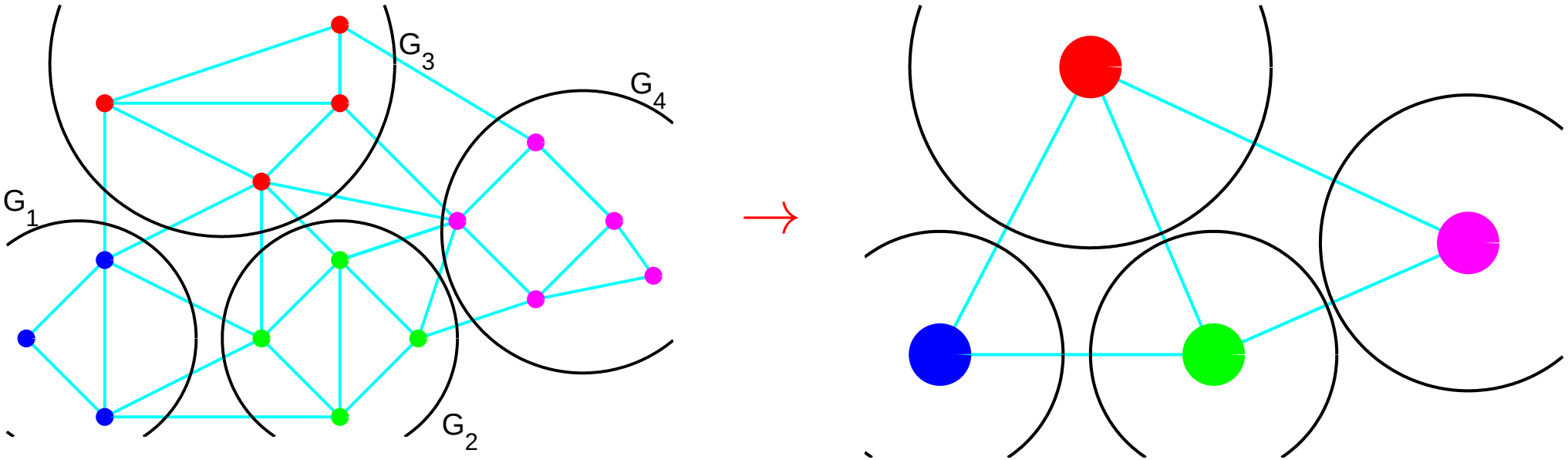


4. Aggregation-based AMG (5)

ULB

Coarse grid matrix: $A_c = P^T A P$ given by

$$(A_c)_{IJ} = \sum_{k \in G_I} \sum_{\ell \in G_J} a_{k\ell}$$



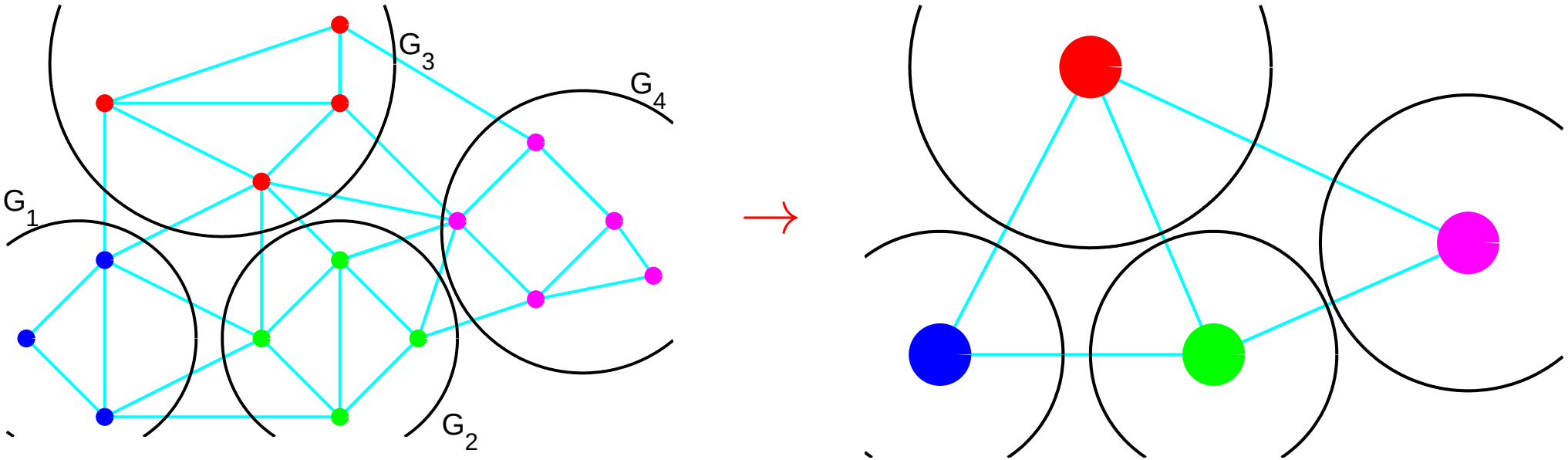
Tends to reproduce the stencil from the fine grid

4. Aggregation-based AMG (5)

ULB

Coarse grid matrix: $A_c = P^T A P$ given by

$$(A_c)_{IJ} = \sum_{k \in G_I} \sum_{\ell \in G_J} a_{k\ell}$$



Tends to reproduce the stencil from the fine grid

- Recursive use raises no difficulties
- Low setup cost & memory requirements

4. Aggregation-based AMG (6)

ULB

No free lunch theorem

We gained something, where are the downsides?

- $\rho \not\to 0$ when the number of relaxation steps $\uparrow$

  Solution: use few relaxation steps

4. Aggregation-based AMG (6)

ULB

No free lunch theorem

We gained something, where are the downsides?

- $\rho \not\to 0$ when the number of relaxation steps $\uparrow$

  Solution: use few relaxation steps

- $\rho \approx 0.1$ out of reach

  Instead: ρ commonly in the range $0.5 - 0.8$

  Solution: use MG as a preconditioner (cf. above)
  (few relaxation steps $\rightarrow$ cheap per iteration)

4. Aggregation-based AMG (6)

ULB

No free lunch theorem

We gained something, where are the downsides?

- $\rho \not\to 0$ when the number of relaxation steps $\uparrow$

  Solution: use few relaxation steps

- $\rho \approx 0.1$ out of reach

  Instead: ρ commonly in the range $0.5 - 0.8$

  Solution: use MG as a preconditioner (cf. above)
  (few relaxation steps $\rightarrow$ cheap per iteration)

- Not optimal with the V-cycle
  (i.e., the number of iter. $\uparrow$ when the number of levels $\uparrow$)

  Solution: enhanced MG cycle – the K-cycle

Remark

Alternatively to the K-cycle, some authors recommend to rescale the coarse grid matrix $A_c = P^T A P$

- Heuristic motivated by the comparison of A_c with coarse grid matrices based on rediscretization, but this comparison makes sense only for regular aggregates and the scaling factor depends on aggregates' size

4. Aggregation-based AMG (7)

ULB

Remark

Alternatively to the K-cycle, some authors recommend to rescale the coarse grid matrix $A_c = P^T A P$

- Heuristic motivated by the comparison of A_c with coarse grid matrices based on rediscretization, but this comparison makes sense only for regular aggregates and the scaling factor depends on aggregates' size
- The two-grid convergence theory is for Galerkin coarse grid matrices only $\rightarrow$ seeking for V-cycle convergence, one may deteriarate two-grid convergence

4. Aggregation-based AMG (7)

ULB

Remark

Alternatively to the K-cycle, some authors recommend to rescale the coarse grid matrix $A_c = P^T A P$

- Heuristic motivated by the comparison of A_c with coarse grid matrices based on rediscretization, but this comparison makes sense only for regular aggregates and the scaling factor depends on aggregates' size
- The two-grid convergence theory is for Galerkin coarse grid matrices only $\rightarrow$ seeking for V-cycle convergence, one may deteriarate two-grid convergence
- In practice: yields improvement (to some extent only)

4. Aggregation-based AMG (7)

ULB

Remark

Alternatively to the K-cycle, some authors recommend to rescale the coarse grid matrix $A_c = P^T A P$

- Heuristic motivated by the comparison of A_c with coarse grid matrices based on rediscretization, but this comparison makes sense only for regular aggregates and the scaling factor depends on aggregates' size
- The two-grid convergence theory is for Galerkin coarse grid matrices only $\rightarrow$ seeking for V-cycle convergence, one may deteriarate two-grid convergence
- In practice: yields improvement (to some extent only)

Some works show that aggregation-based AMG can be good with rescaling, but none show that it is not even better with the K-cyle

4. Aggregation-based AMG (8)

ULB

Remark (paradox)

Geometric MG is heavily based on:

two-grid analysis (for the theoretical assessment)

and the V-cycle (for practical usage),

but, in fact, optimal two-grid convergence is not sufficient to guarantee optimal V-cycle convergence

5. The K-cycle (1)

ULB

Reminder: the coarse grid correction has as main step

- Solve the coarse problem

  Coarse grid matrix A_c ($n_c \times n_c$): solve $A_c \, \mathbf{u}_c = \mathbf{r}_c$

In a multigrid algorithm, "solve" $\rightarrow$ "approximate solve"

5. The K-cycle (1)

ULB

Reminder: the coarse grid correction has as main step

- Solve the coarse problem

  Coarse grid matrix A_c ($n_c \times n_c$): solve $A_c \mathbf{u}_c = \mathbf{r}_c$

In a multigrid algorithm, "solve" $\rightarrow$ "approximate solve"

The coarse system is solved with a few iterations, based on the two-grid scheme at that level (referencing thus a further coarse level)

The used scheme determines the multigrid cycle:

- 1 iteration of MG as a solver: V-cycle
- 2 iterations of MG as a solver: W-cycle

5. The K-cycle (1)

ULB

Reminder: the coarse grid correction has as main step

- Solve the coarse problem

  Coarse grid matrix A_c ($n_c \times n_c$): solve $A_c \mathbf{u}_c = \mathbf{r}_c$

In a multigrid algorithm, "solve" $\rightarrow$ "approximate solve"

The coarse system is solved with a few iterations, based on the two-grid scheme at that level (referencing thus a further coarse level)

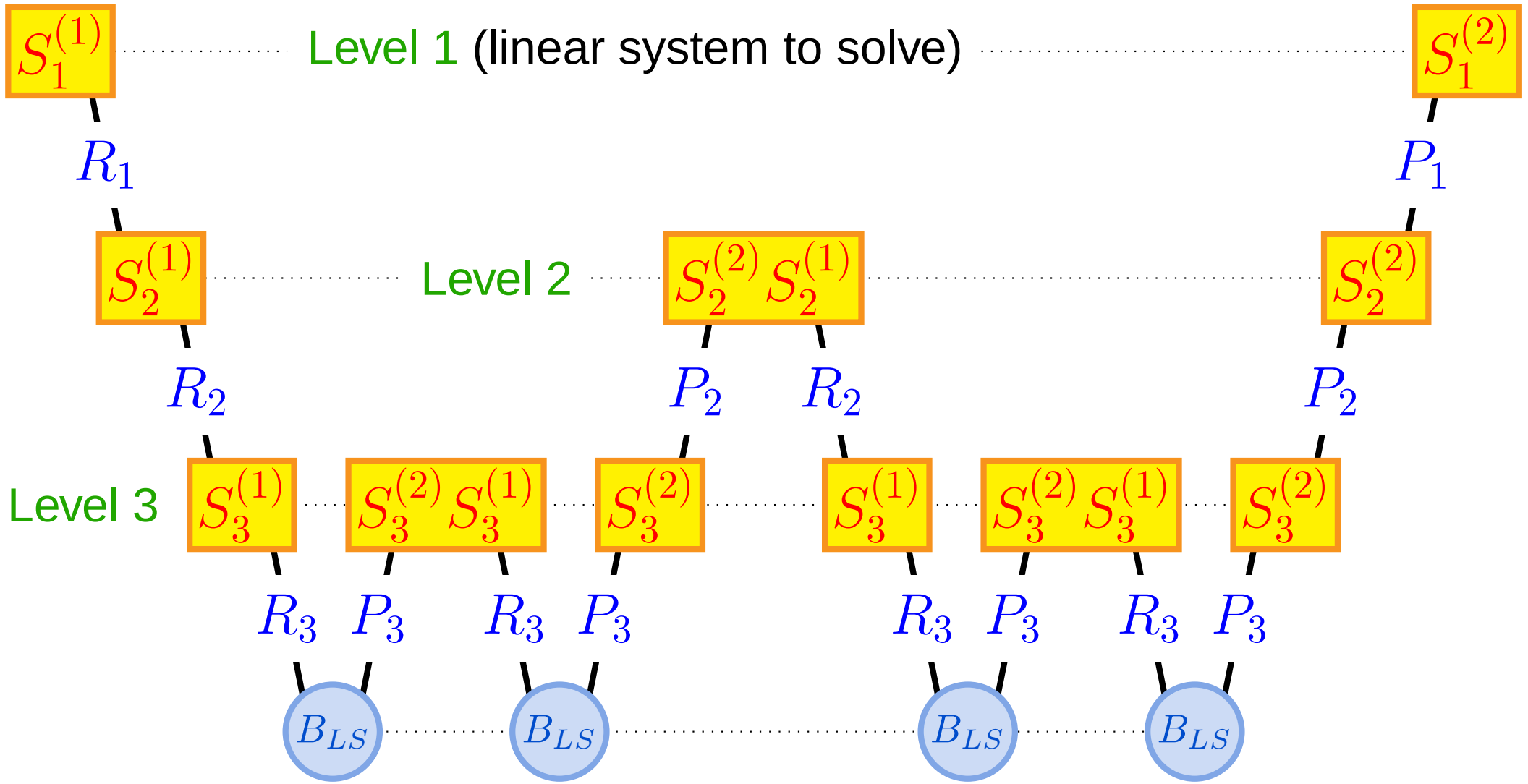
The used scheme determines the multigrid cycle:

- 1 iteration of MG as a solver: V-cycle
- 2 iterations of MG as a solver: W-cycle
- Enhancement:
  2 iterations of MG as a preconditioner: K-cycle

5. The K-cycle (2)

ULB

Workflow for 1 iteration using the K-cycle or the W-cycle ($L = 4$)



Level 4 (B_{LS} : bottom level solver)

5. The K-cycle (3)

ULB

- The workflow is the same for the K-cycle and the W-cycle. The K-cycle is only slightly more costly.

5. The K-cycle (3)

ULB

- The workflow is the same for the K-cycle and the W-cycle. The K-cycle is only slightly more costly.
- Optimal two-grid convergence can be sufficient to guarantee optimal multigrid convergence with the W-cycle and with the K-cycle

5. The K-cycle (3)

ULB

- The workflow is the same for the K-cycle and the W-cycle. The K-cycle is only slightly more costly.
- Optimal two-grid convergence can be sufficient to guarantee optimal multigrid convergence with the W-cycle and with the K-cycle
- In the SPD case, the theoretical limit is, for both cycles,

$$\max_{1 \leq k \leq L-1} \rho_{\mathsf{TG}}^{(k)} < 0.5 \quad ,$$

where $\rho_{\mathsf{TG}}^{(k)}$ is the two-grid convergence factor at level k

5. The K-cycle (3)

ULB

- The workflow is the same for the K-cycle and the W-cycle. The K-cycle is only slightly more costly.
- Optimal two-grid convergence can be sufficient to guarantee optimal multigrid convergence with the W-cycle and with the K-cycle
- In the SPD case, the theoretical limit is, for both cycles,

$$\max_{1 \leq k \leq L-1} \rho_{\mathsf{TG}}^{(k)} < 0.5 \quad ,$$

  where $\rho_{\mathsf{TG}}^{(k)}$ is the two-grid convergence factor at level k
- This upper bound of 0.5 is realistic for the W-cycle, but reflects shortcomings in the analysis regarding the K-cycle

5. The K-cycle (4)

ULB

In practice, the K-cycle allows to stabilize the number of iterations even when $\rho_{\mathsf{TG}}^{(k)}$ is relatively large

Example: Number of iterations to reduce the relative residual error by 10^{-12} as a function of the number of levels

	7 levels	14 levels
	$0.49 < \rho_{\mathsf{TG}}^{(k)} < 0.50$	
	$\left(\frac{\ln(2\,\varepsilon^{-1})}{2\sqrt{1-\rho_{\mathsf{TG}}}} \approx 20\right)$	
V	188	> 999
W	37	50
K	20	20
	$0.79 < \rho_{\mathsf{TG}}^{(k)} < 0.80$	
	$\left(\frac{\ln(2\,\varepsilon^{-1})}{2\sqrt{1-\rho_{\mathsf{TG}}}} \approx 32\right)$	
V	256	> 999
W	108	315
K	42	44

5. The K-cycle (5)

ULB

Computational cost: 1 step of the MG method involves

- ν relaxation steps at level 1
  $+ \; 2$ iterations at level 2

5. The K-cycle (5)

ULB

Computational cost: 1 step of the MG method involves

- ν relaxation steps at level 1
  $+ 2$ iterations at level 2
- that is, ν relaxation steps at level 1
  $+ 2$ times: ν relaxation steps at level 2
  $+ 2$ iterations at level 3

5. The K-cycle (5)

ULB

Computational cost: 1 step of the MG method involves

- ν relaxation steps at level 1
  $+ \; 2$ iterations at level 2
- that is, ν relaxation steps at level 1
  $+ \; 2 \; \nu$ relaxation steps at level 2
  $+ \; 2 \cdot 2 = 2^2$ iterations at level 3

5. The K-cycle (5)

ULB

Computational cost: 1 step of the MG method involves

- ν relaxation steps at level 1
  $+ 2$ iterations at level 2
- that is, ν relaxation steps at level 1
  $+ 2\ \nu$ relaxation steps at level 2
  $+ 2 \cdot 2 = 2^2$ iterations at level 3
- that is, ν relaxation steps at level 1
  $+ 2\ \nu$ relaxation steps at level 2
  $+ 2^2 \nu$ relaxation steps at level 3
  $+ 2^3$ iterations at level 4
- ...

5. The K-cycle (5)

ULB

Computational cost: 1 step of the MG method involves

- ν relaxation steps at level 1
  $+ 2$ iterations at level 2
- that is, ν relaxation steps at level 1
  $+ 2\,\nu$ relaxation steps at level 2
  $+ 2 \cdot 2 = 2^2$ iterations at level 3
- that is, ν relaxation steps at level 1
  $+ 2\,\nu$ relaxation steps at level 2
  $+ 2^2\nu$ relaxation steps at level 3
  $+ 2^3$ iterations at level 4
- ...

Cost of $2^{k-1}\,\nu$ relaxation steps at level k : $\sim \; 2^{k-1}\,\nu\, nnz(A_k)$

5. The K-cycle (6)

ULB

K-cycle or W-cycle

$2^{k-1}\,\nu$ relaxation steps at level k : $\text{Work}_k \sim 2^{k-1}\,\nu\,nnz(A_k)$

V-cycle

ν relaxation steps at level k : $\text{Work}_k \sim \nu\,nnz(A_k)$

- Standard complexity: $\mathcal{C} = \dfrac{\sum_{k=1}^{L} n_k}{n}$

Operator complexity: $\mathcal{C}_A = \dfrac{\sum_{k=1}^{L} nnz(A_k)}{nnz(A)}$

Weighted complexity: $\mathcal{C}_W = \dfrac{\sum_{k=1}^{L} 2^{k-1}\,nnz(A_k)}{nnz(A)}$

5. The K-cycle (6)

ULB

K-cycle or W-cycle

$2^{k-1}\,\nu$ relaxation steps at level k : $\text{Work}_k \sim 2^{k-1}\,\nu\,nnz(A_k)$

V-cycle

ν relaxation steps at level k : $\text{Work}_k \sim \nu\,nnz(A_k)$

- Standard complexity: $\mathcal{C} = \dfrac{\sum_{k=1}^{L} n_k}{n}$

  Operator complexity: $\mathcal{C}_A = \dfrac{\sum_{k=1}^{L} nnz(A_k)}{nnz(A)}$

  Weighted complexity: $\mathcal{C}_W = \dfrac{\sum_{k=1}^{L} 2^{k-1}\, nnz(A_k)}{nnz(A)}$

- $\dfrac{\text{Cost of K-cycle}}{\text{Cost of V-cycle}} \approx \dfrac{\mathcal{C}_W}{\mathcal{C}_A}$

5. The K-cycle (7)

If

$$\frac{nnz(A_j)}{nnz(A_{j+1})} > \tau > 2 ,$$

then $nnz(A_k) < \tau^{-(k-1)}\, nnz(A)$ and

$$c_W = \frac{\sum_{k=1}^{L} 2^{k-1}\, nnz(A_k)}{nnz(A)} < \sum_{k=1}^{L} \left(\frac{2}{\tau}\right)^{k-1} < \frac{1}{1-\frac{2}{\tau}}$$

whereas

$$c_A = \frac{\sum_{k=1}^{L} nnz(A_k)}{nnz(A)} < \sum_{k=1}^{L} \left(\frac{1}{\tau}\right)^{k-1} < \frac{1}{1-\frac{1}{\tau}}$$

5. The K-cycle (7)

ULB

If

$$\frac{nnz(A_j)}{nnz(A_{j+1})} > \tau > 2 ,$$

then $nnz(A_k) < \tau^{-(k-1)}\, nnz(A)$ and

$$\mathcal{C}_W = \frac{\sum_{k=1}^{L} 2^{k-1}\, nnz(A_k)}{nnz(A)} < \sum_{k=1}^{L} \left(\frac{2}{\tau}\right)^{k-1} < \frac{1}{1-\frac{2}{\tau}}$$

whereas

$$\mathcal{C}_A = \frac{\sum_{k=1}^{L} nnz(A_k)}{nnz(A)} < \sum_{k=1}^{L} \left(\frac{1}{\tau}\right)^{k-1} < \frac{1}{1-\frac{1}{\tau}}$$

For $\tau \gtrapprox 4$, the extra cost for the K-cycle is moderate

K-cycle with $\tau \geq 4$: cheaper than the V-cycle with $\mathcal{C}_A \geq 2$

6. Quality aware aggregation (1)

ULB

A SPD : $K(A\,,P\,,D_A) \;=\; \sup_{\mathbf{v}\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \dfrac{\mathbf{v}^T\,D_A\Big(I-P\big(P^T D_A\,P\big)^{-1}P^T D_A\Big)\,\mathbf{v}}{\mathbf{v}^T\,A\,\mathbf{v}}$

With aggregation based P , one has, $\forall i \in [1,n]$:

$P_{iJ} \neq 0\,(i.e.,\, P_{iJ}=1) \iff i \in G_J$ (true for only one J)

Hence, $\left(P^T D_A\,P\right)$ is diagonal:

$$\left(P^T D_A\,P\right)_{IJ} \;=\; \sum_{k=1}^{n} P_{kI}\,a_{kk}\,P_{kJ} \;=\; 0 \quad \text{if } I\neq J$$

Further,

$$\left(P\left(P^T D_A\,P\right)^{-1}P^T\right)_{ik} \;=\; \sum_{J=1}^{n_c} P_{iJ}\left(\left(P^T D_A\,P\right)^{-1}\right)_{JJ} P_{kJ}$$

can be $\neq 0$ only if i and k belong to the same G_J

6. Quality aware aggregation (2)

ULB

Hence

$$Z \;=\; D_A\big(I - P\left(P^T D_A\, P\right)^{-1} P^T D_A\big) \;=\; \begin{pmatrix} Z_1 & & \\ & \ddots & \\ & & Z_{n_c} \end{pmatrix}$$

is block diagonal (using an ordering compliant with the partitioning in aggregates), and

$$\mathbf{v}^t\, Z\, \mathbf{v} \;=\; \begin{pmatrix} \mathbf{v}_{G_1} \\ \vdots \\ \mathbf{v}_{G_{n_c}} \end{pmatrix}^T \begin{pmatrix} Z_1 & & \\ & \ddots & \\ & & Z_{n_c} \end{pmatrix} \begin{pmatrix} \mathbf{v}_{G_1} \\ \vdots \\ \mathbf{v}_{G_{n_c}} \end{pmatrix}$$

$$= \sum_{J=1}^{n_c} \mathbf{v}_{G_J}^T\, Z_J\, \mathbf{v}_{G_J} \quad ,$$

where $\mathbf{v}_{G_J}$ is the restriction of $\mathbf{v}$ to the nodes in G_J

6. Quality aware aggregation (3)

ULB

Suppose now that there exists

$$A_b = \begin{pmatrix} A_1^{(b)} & & \\ & \ddots & \\ & & A_{n_c}^{(b)} \end{pmatrix}$$

such that $\mathbf{v}^T A\,\mathbf{v} \geq \mathbf{v}^T A_b\,\mathbf{v} \qquad \forall \mathbf{v} \in \mathbb{R}^n$

Then:

$$\begin{aligned} K(A, P, D_A) &= \sup_{\mathbf{v}\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \frac{\mathbf{v}^T D_A \left(I - P\left(P^T D_A P\right)^{-1} P^T D_A\right)\mathbf{v}}{\mathbf{v}^T A\,\mathbf{v}} \\ &\leq \sup_{\mathbf{v}\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \frac{\mathbf{v}^T Z\,\mathbf{v}}{\mathbf{v}^T A_b\,\mathbf{v}} \\ &= \sup_{\mathbf{v}\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \frac{\sum_{J=1}^{n_c} \mathbf{v}_{G_J}^T Z_J\,\mathbf{v}_{G_J}}{\sum_{J=1}^{n_c} \mathbf{v}_{G_J}^T A_J^{(b)}\,\mathbf{v}_{G_J}} \\ &\leq \max_{1\leq J\leq n_c} \sup_{\mathbf{v}\in\mathbb{R}^{|G_J|}} \frac{\mathbf{v}^T Z_J\,\mathbf{v}}{\mathbf{v}^T A_J^{(b)}\,\mathbf{v}} \end{aligned}$$

6. Quality aware aggregation (4)

ULB

- The quantity

$$\mu_{G_J} = \sup_{\mathbf{v} \in \mathbb{R}^{|G_J|}} \frac{\mathbf{v}^T Z_J \mathbf{v}}{\mathbf{v}^T A_J^{(b)} \mathbf{v}}$$

measures aggregate's quality (the lower the better)

6. Quality aware aggregation (4)

ULB

- The quantity

$$\mu_{G_J} = \sup_{\mathbf{v} \in \mathbb{R}^{|G_J|}} \frac{\mathbf{v}^T Z_J \mathbf{v}}{\mathbf{v}^T A_J^{(b)} \mathbf{v}}$$

  measures aggregate's quality (the lower the better)

- For μ_J being finite, $A_J^{(b)}$ should be positive definite or have the same null space as Z_J

6. Quality aware aggregation (4)

ULB

- The quantity

$$\mu_{G_J} = \sup_{\mathbf{v} \in \mathbb{R}^{|G_J|}} \frac{\mathbf{v}^T Z_J \mathbf{v}}{\mathbf{v}^T A_J^{(b)} \mathbf{v}}$$

  measures aggregate's quality (the lower the better)

- For μ_J being finite, $A_J^{(b)}$ should be positive definite or have the same null space as Z_J
- Hence there are strict requirements on A_b

  Nevertheless, for (weakly) diagonally dominant matrices, relevant $A_J^{(b)}$ can be easily set up for any G_J

  ($A_J^{(b)}$ equals the restriction of A to the nodes in G_J plus some diagonal correction)

  More general matrices: $\exists$ heuristic extension

6. Quality aware aggregation (5)

ULB

Nonsymmetric matrices

Reminder: if A is nonsymmetric but positive definite in $\mathbb{R}^n$, the meaningful quantity is

$$K(A_S, P, D_A) \quad ,$$

where $A_S = \frac{1}{2}(A + A^T)$

$\rightarrow$ measure of the quality based on A_S

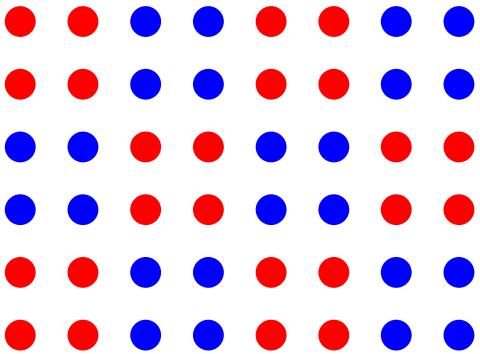
(The approach is rigorous if A_S is diagonally dominant)

6. Quality aware aggregation (6)

ULB

Example: 5 point FD for $-\frac{\partial^2 u}{\partial x^2} - \varepsilon \frac{\partial^2 u}{\partial x^2} = f$,
size 4 aggregates

Boxwise aggregation



$$\mu_G = 1 + \varepsilon^{-1}$$

$$\leq 3 + \sqrt{2}$$

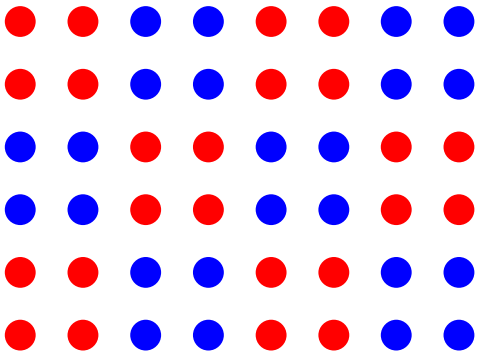
if $\varepsilon \geq (2+\sqrt{2})^{-1}$

6. Quality aware aggregation (6)

ULB

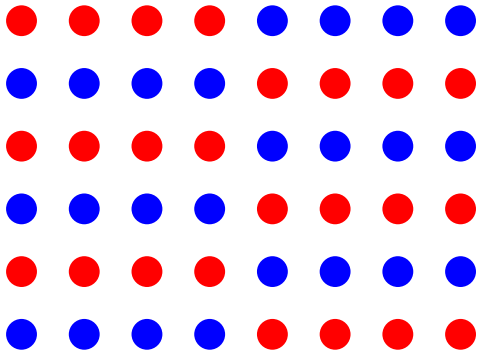
Example: 5 point FD for $-\frac{\partial^2 u}{\partial x^2} - \varepsilon \frac{\partial^2 u}{\partial x^2} = f$,
size 4 aggregates

Boxwise aggregation



$$\mu_G = 1 + \varepsilon^{-1}$$
$$\leq 3 + \sqrt{2}$$
if $\varepsilon \geq (2+\sqrt{2})^{-1}$

Linewise aggregation



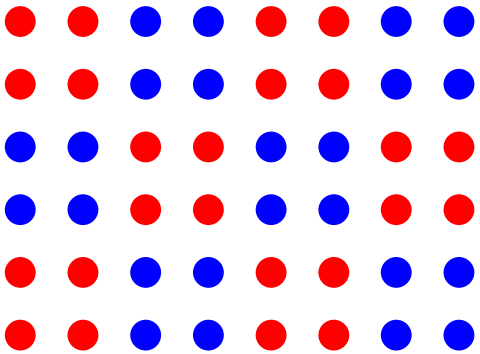
$$\mu_G = (1+\varepsilon)(2+\sqrt{2})$$
$$\leq 3 + \sqrt{2}$$
if $\varepsilon \leq (2+\sqrt{2})^{-1}$

6. Quality aware aggregation (6)

ULB

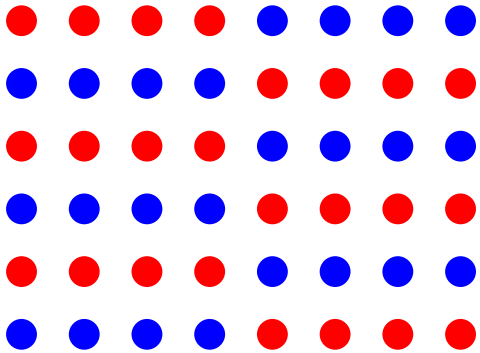
Example: 5 point FD for $-\frac{\partial^2 u}{\partial x^2} - \varepsilon \frac{\partial^2 u}{\partial x^2} = f$,
size 4 aggregates

Boxwise aggregation



$$\mu_G = 1 + \varepsilon^{-1} \leq 3 + \sqrt{2} \quad \text{if } \varepsilon \geq (2+\sqrt{2})^{-1}$$

Linewise aggregation



$$\mu_G = (1+\varepsilon)(2+\sqrt{2}) \leq 3 + \sqrt{2} \quad \text{if } \varepsilon \leq (2+\sqrt{2})^{-1}$$

Thus: $\mu_G \leq 3 + \sqrt{2}$ independently of ε
if one always chooses the good shape & orientation

6. Quality aware aggregation (7)

ULB

The convergence theory via aggregates' quality:

- is compatible with irregular geometries, unstructured grids, jumps in coefficients, etc
  (the bounds are essentially unaffected if the aggregates are chosen properly)

6. Quality aware aggregation (7)

ULB

The convergence theory via aggregates' quality:

- is compatible with irregular geometries, unstructured grids, jumps in coefficients, etc
  (the bounds are essentially unaffected if the aggregates are chosen properly)
- requires diagonally dominant matrices
  (e.g., M-matrices with nonnegative row-sum),
  but has natural heuristic extensions

6. Quality aware aggregation (7)

ULB

The convergence theory via aggregates' quality:

- is compatible with irregular geometries, unstructured grids, jumps in coefficients, etc
  (the bounds are essentially unaffected if the aggregates are chosen properly)
- requires diagonally dominant matrices
  (e.g., M-matrices with nonnegative row-sum),
  but has natural heuristic extensions
- whenever applicable, holds at every level of the hierarchy

6. Quality aware aggregation (7)

ULB

The convergence theory via aggregates' quality:

- is compatible with irregular geometries, unstructured grids, jumps in coefficients, etc
  (the bounds are essentially unaffected if the aggregates are chosen properly)
- requires diagonally dominant matrices
  (e.g., M-matrices with nonnegative row-sum),
  but has natural heuristic extensions
- whenever applicable, holds at every level of the hierarchy
- covers symmetric and nonsymmetric problems in a uniform fashion

6. Quality aware aggregation (8)

ULB

From quality assessment to quality control

The approximation property constant satisfies

$$K(A, P, D_A) \leq \max_J \mu_J$$

- A posteriori control of given aggregation scheme: limited utility (often a few aggregates with large μ_G)

6. Quality aware aggregation (8)

ULB

From quality assessment to quality control

The approximation property constant satisfies

$$K(A, P, D_A) \leq \max_J \mu_J$$

- A posteriori control of given aggregation scheme: limited utility (often a few aggregates with large μ_G)
- $\rightarrow$ Aggregation algorithm based on the control of μ_G

6. Quality aware aggregation (8)

ULB

From quality assessment to quality control

The approximation property constant satisfies

$$K(A, P, D_A) \leq \max_J \mu_J$$

- A posteriori control of given aggregation scheme: limited utility (often a few aggregates with large μ_G)
- $\rightarrow$ Aggregation algorithm based on the control of μ_G
- The repeated assessment of μ_G is costly for aggregates of arbitrary size

  But, for a pair $\{i, j\}$, $\mu_{\{i,j\}}$ is a simple function of a_{ii}, a_{jj}, a_{ij} and $\sum_{k=1}^{n} a_{ik}$

  $\rightarrow$ base the procedure on pairwise aggregation

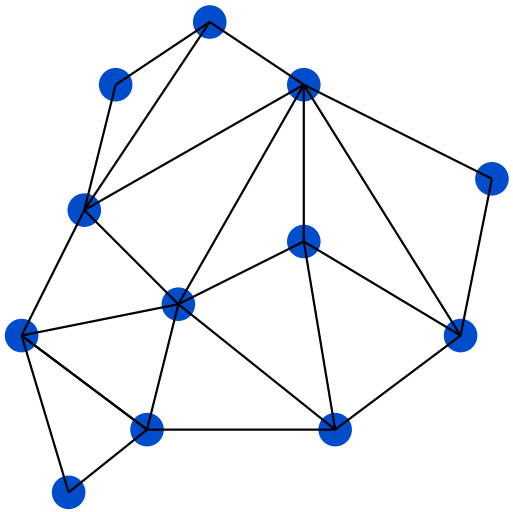
6. Quality aware aggregation (9)

ULB

Pairwise aggregation

1. Pick up a node i
2. For all j s.t. $a_{ij} \neq 0$:
   Compute $\mu_{\{i,j\}}$
3. Select j which minimizes $\mu_{\{i,j\}}$
4. If $\mu_{\{i,j\}}$ is below the given threshold:
   the next aggregate is $\{i, j\}$

   Otherwise:
   the next aggregate is $\{i\}$
5. If some node(s) have not
   been processed yet: GOTO 1.



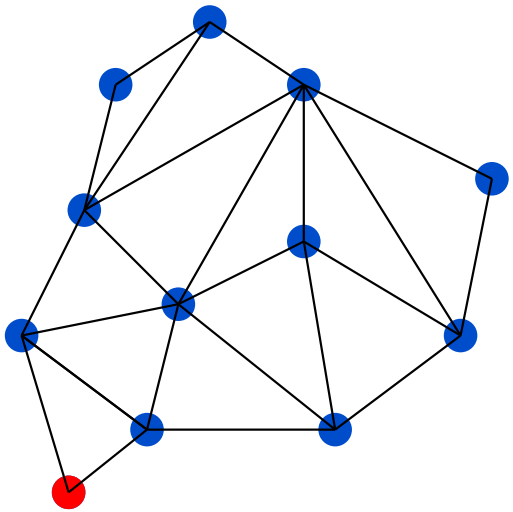
6. Quality aware aggregation (9)

ULB

Pairwise aggregation

1. Pick up a node i
2. For all j s.t. $a_{ij} \neq 0$:
   Compute $\mu_{\{i,j\}}$
3. Select j which minimizes $\mu_{\{i,j\}}$
4. If $\mu_{\{i,j\}}$ is below the given threshold:
   the next aggregate is $\{i, j\}$

   Otherwise:
   the next aggregate is $\{i\}$
5. If some node(s) have not
   been processed yet: GOTO 1.



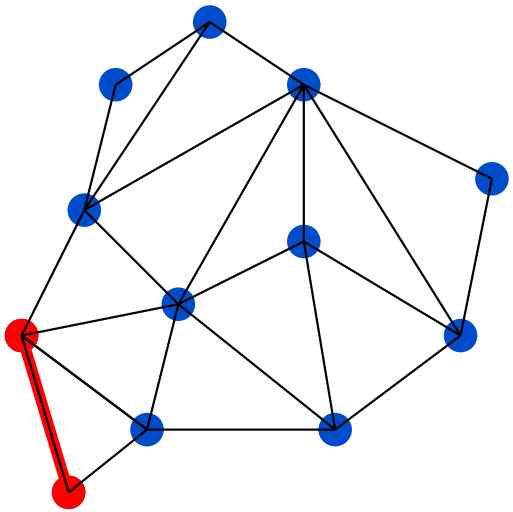
6. Quality aware aggregation (9)

ULB

Pairwise aggregation

1. Pick up a node i
2. For all j s.t. $a_{ij} \neq 0$:
   Compute $\mu_{\{i,j\}}$
3. Select j which minimizes $\mu_{\{i,j\}}$
4. If $\mu_{\{i,j\}}$ is below the given threshold:
   the next aggregate is $\{i, j\}$

   Otherwise:
   the next aggregate is $\{i\}$
5. If some node(s) have not
   been processed yet: GOTO 1.



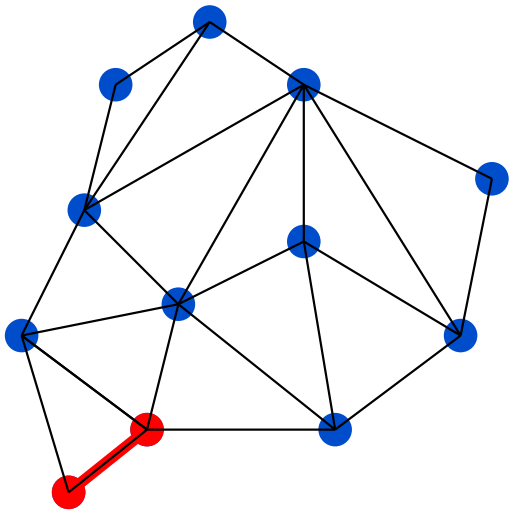
6. Quality aware aggregation (9)

ULB

Pairwise aggregation

1. Pick up a node i
2. For all j s.t. $a_{ij} \neq 0$:
   Compute $\mu_{\{i,j\}}$
3. Select j which minimizes $\mu_{\{i,j\}}$
4. If $\mu_{\{i,j\}}$ is below the given threshold:
   the next aggregate is $\{i, j\}$

   Otherwise:
   the next aggregate is $\{i\}$
5. If some node(s) have not
   been processed yet: GOTO 1.



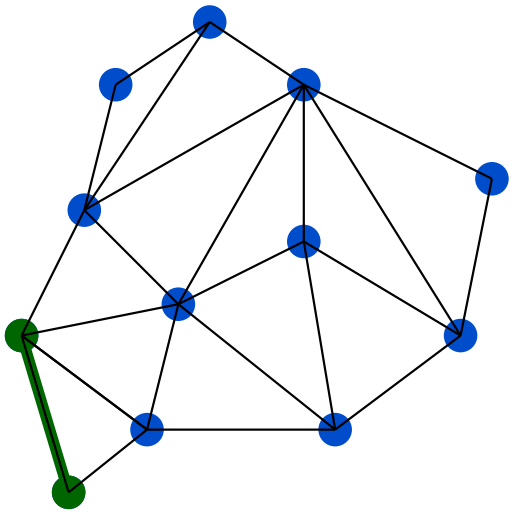
6. Quality aware aggregation (9)

ULB

Pairwise aggregation

1. Pick up a node i
2. For all j s.t. $a_{ij} \neq 0$:
   Compute $\mu_{\{i,j\}}$
3. Select j which minimizes $\mu_{\{i,j\}}$
4. If $\mu_{\{i,j\}}$ is below the given threshold:
   the next aggregate is $\{i, j\}$

   Otherwise:
   the next aggregate is $\{i\}$
5. If some node(s) have not
   been processed yet: GOTO 1.



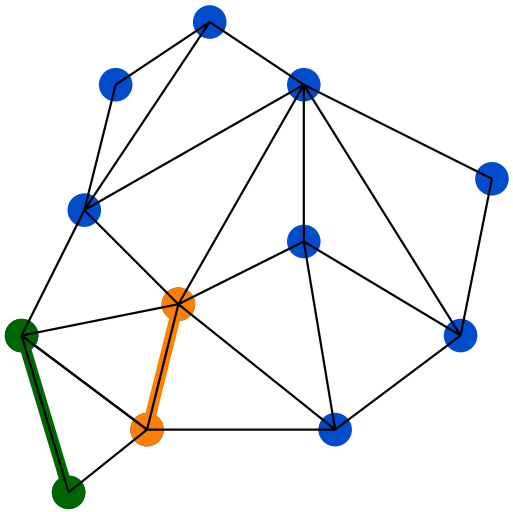
6. Quality aware aggregation (9)

ULB

Pairwise aggregation

1. Pick up a node i
2. For all j s.t. $a_{ij} \neq 0$:
   Compute $\mu_{\{i,j\}}$
3. Select j which minimizes $\mu_{\{i,j\}}$
4. If $\mu_{\{i,j\}}$ is below the given threshold:
   the next aggregate is $\{i, j\}$

   Otherwise:
   the next aggregate is $\{i\}$
5. If some node(s) have not
   been processed yet: GOTO 1.



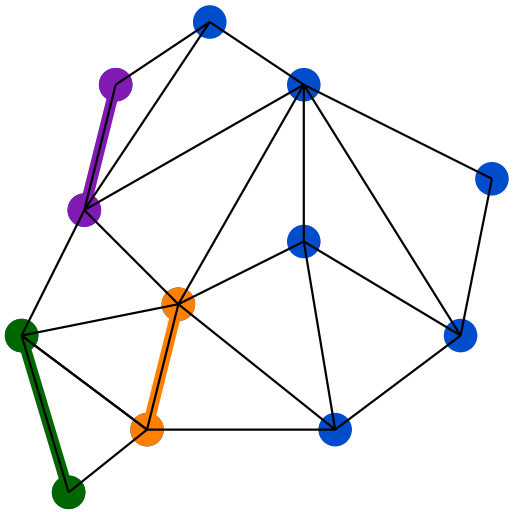
6. Quality aware aggregation (9)

ULB

Pairwise aggregation

1. Pick up a node i
2. For all j s.t. $a_{ij} \neq 0$:
   Compute $\mu_{\{i,j\}}$
3. Select j which minimizes $\mu_{\{i,j\}}$
4. If $\mu_{\{i,j\}}$ is below the given threshold:
   the next aggregate is $\{i, j\}$

   Otherwise:
   the next aggregate is $\{i\}$
5. If some node(s) have not
   been processed yet: GOTO 1.



6. Quality aware aggregation (10)

ULB

- In general,

$$nnz(A_c) \approx \frac{nnz(A)}{\text{Mean aggregate size}}$$

Aggregates of size at most 2 do not yield a sufficiently fast decrease (each MG iteration would be too costly)

With the K-cycle, MG iterations remain dominated by fine grid computations if Mean aggregate size $\gtrapprox 4$

6. Quality aware aggregation (10)

ULB

- In general,

$$nnz(A_c) \approx \frac{nnz(A)}{\text{Mean aggregate size}}$$

  Aggregates of size at most 2 do not yield a sufficiently fast decrease (each MG iteration would be too costly)

  With the K-cycle, MG iterations remain dominated by fine grid computations if Mean aggregate size $\gtrapprox 4$

- One way to achieve this is to form the intermediate coarse grid matrix based on pairwise aggregation, and apply the pairwise algorithm to this latter, forming thus pairs of pairs $\rightarrow$ Repeated Pairwise aggregation

6. Quality aware aggregation (10)

ULB

- In general,

$$nnz(A_c) \approx \frac{nnz(A)}{\text{Mean aggregate size}}$$

  Aggregates of size at most 2 do not yield a sufficiently fast decrease (each MG iteration would be too costly)

  With the K-cycle, MG iterations remain dominated by fine grid computations if Mean aggregate size $\gtrapprox 4$

- One way to achieve this is to form the intermediate coarse grid matrix based on pairwise aggregation, and apply the pairwise algorithm to this latter, forming thus pairs of pairs $\rightarrow$ Repeated Pairwise aggregation

- Doing so, one can check if a tentative pair of pair has an acceptable quality indicator in the fine grid matrix
  Trick: check $\mu_G \leq \overline{\mu}$ without computing $\mu_G \rightarrow$ cheap

6. Quality aware aggregation (11)

ULB

Repeated Pairwise aggregation

$s = 1$; $A^{(s)} = A$

Apply pairwise aggregation to $A^{(s)}$

Form aggregated matrix $A^{(s+1)}$

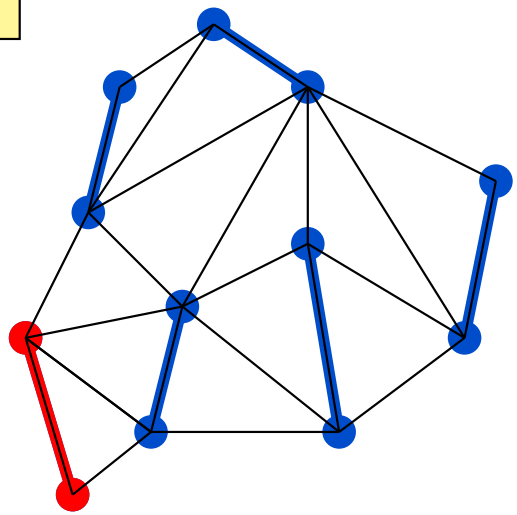
$nnz(A^{(s+1)}) < \frac{nnz(A)}{\tau}$ or s==n_{pass} ?

no

$s \leftarrow s + 1$

yes

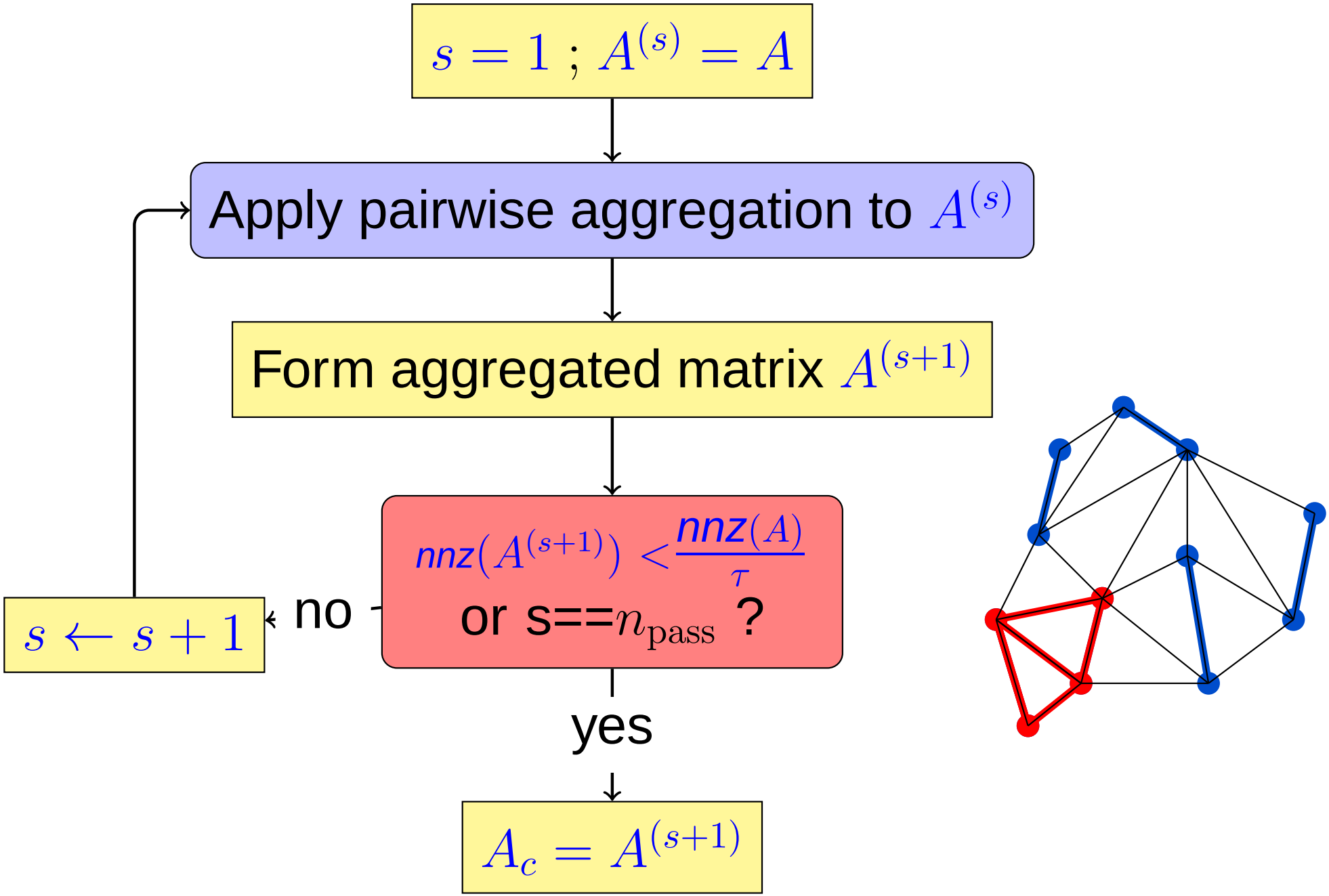
$A_c = A^{(s+1)}$



6. Quality aware aggregation (11)

ULB

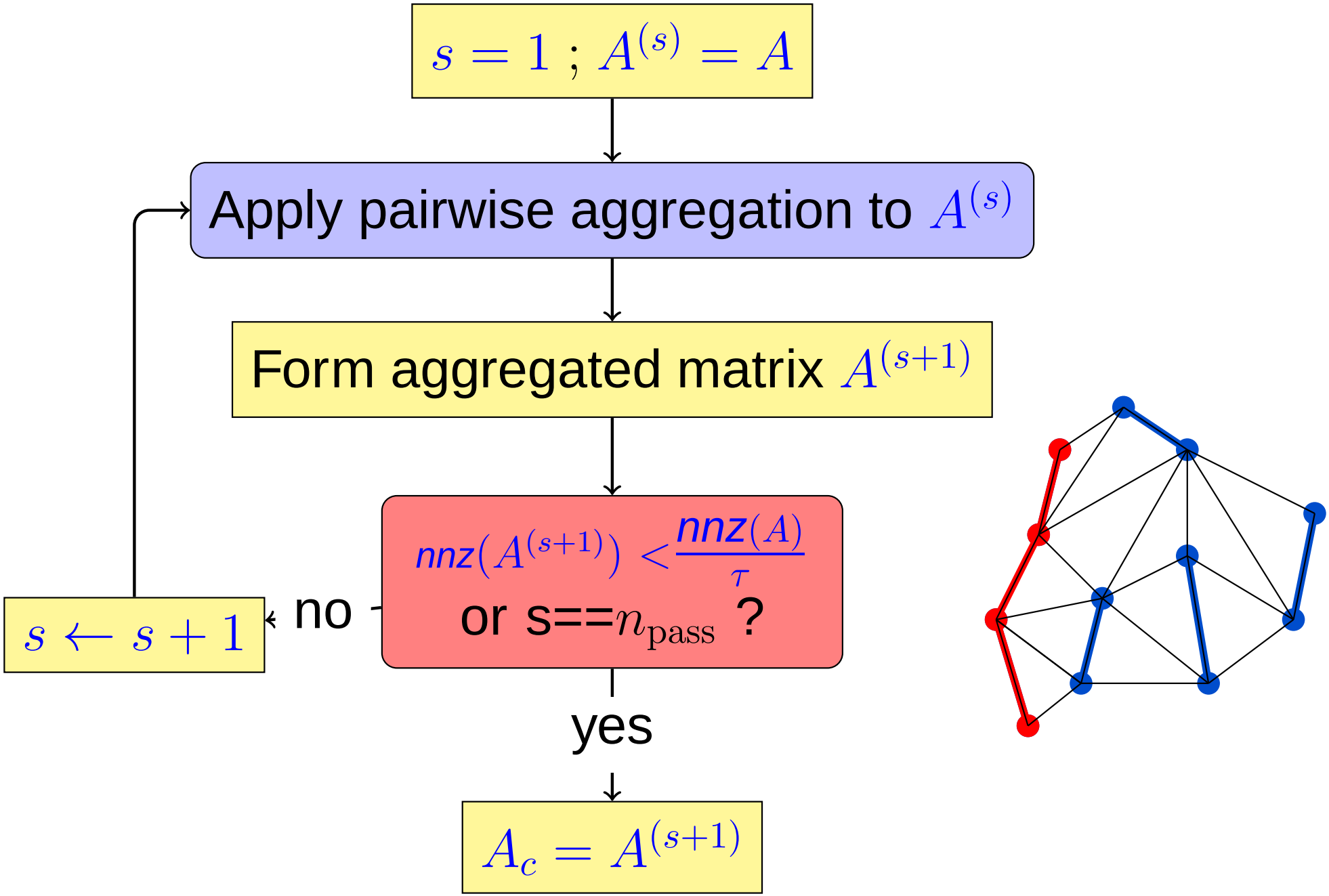
Repeated Pairwise aggregation



6. Quality aware aggregation (11)

ULB

Repeated Pairwise aggregation



6. Quality aware aggregation (11)

ULB

Repeated Pairwise aggregation

$s = 1$; $A^{(s)} = A$

Check $\mu_G < \overline{\mu}$ in A

Apply pairwise aggregation to $A^{(s)}$

Form aggregated matrix $A^{(s+1)}$

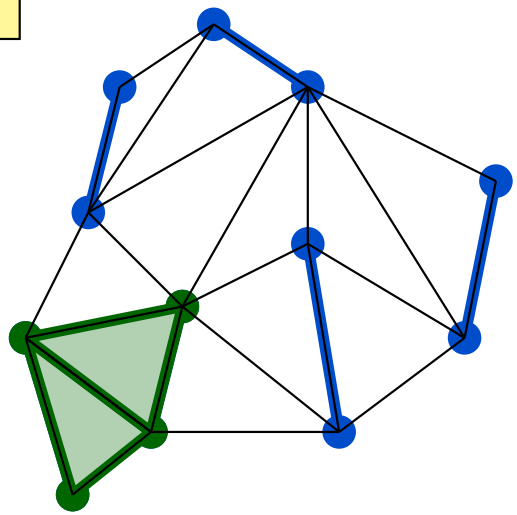
$nnz(A^{(s+1)}) < \frac{nnz(A)}{\tau}$ or s==n_{pass} ?

no

$s \leftarrow s + 1$

yes

$A_c = A^{(s+1)}$



6. Quality aware aggregation (11)

ULB

Repeated Pairwise aggregation

$s = 1$; $A^{(s)} = A$

Check $\mu_G < \overline{\mu}$ in A

Apply pairwise aggregation to $A^{(s)}$

Form aggregated matrix $A^{(s+1)}$

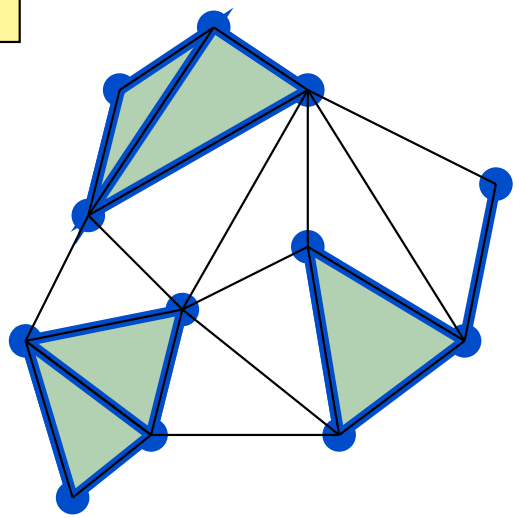
$nnz(A^{(s+1)}) < \frac{nnz(A)}{\tau}$ or s==n_{pass} ?

no

$s \leftarrow s + 1$

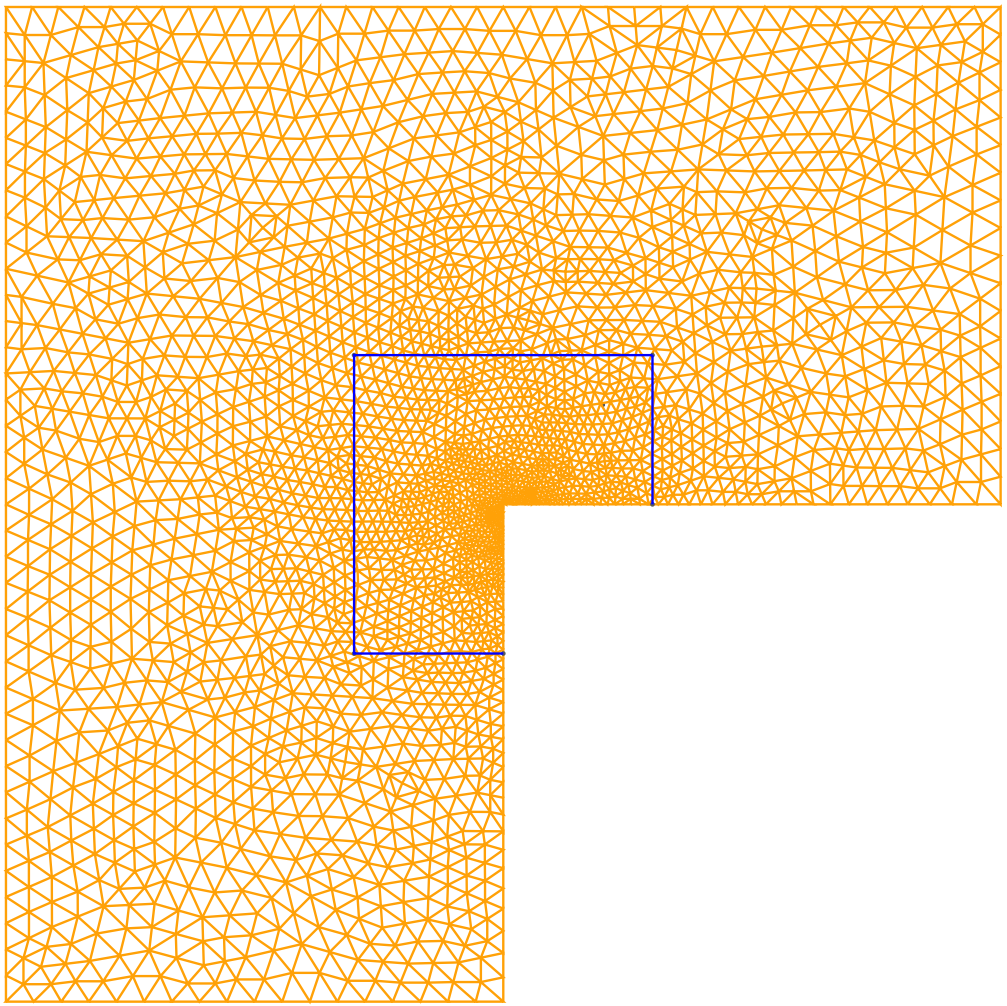
yes

$A_c = A^{(s+1)}$

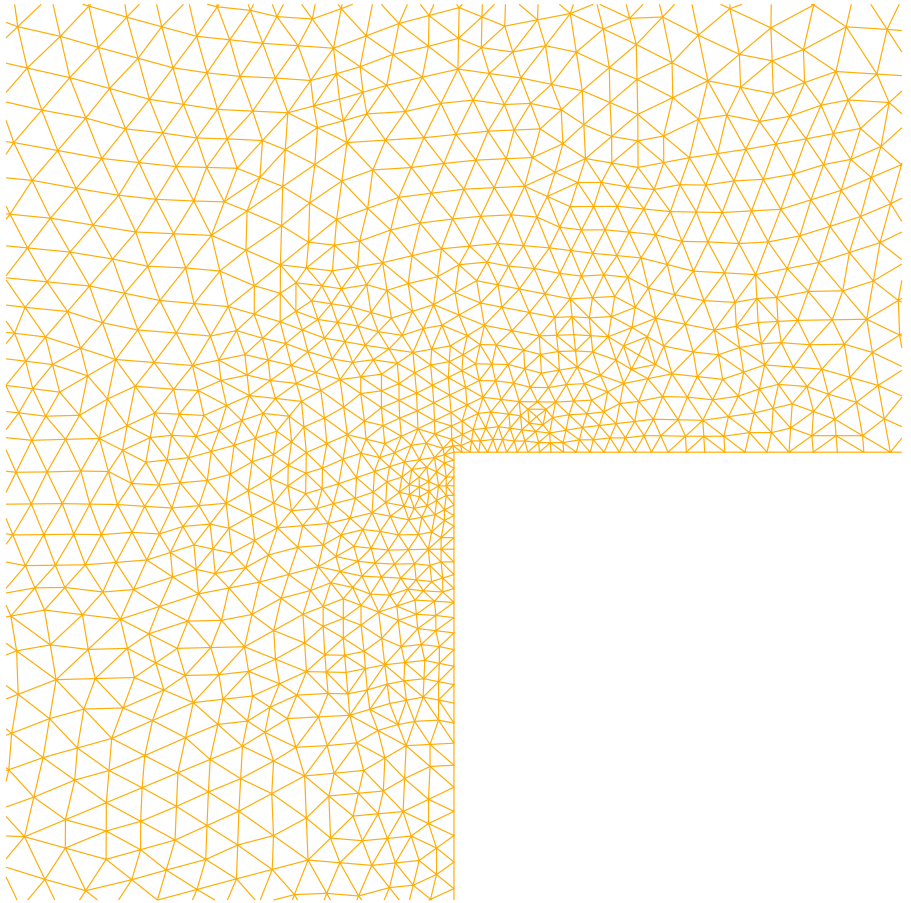


6. Quality aware aggregation (12)

Linear Finite element grid:

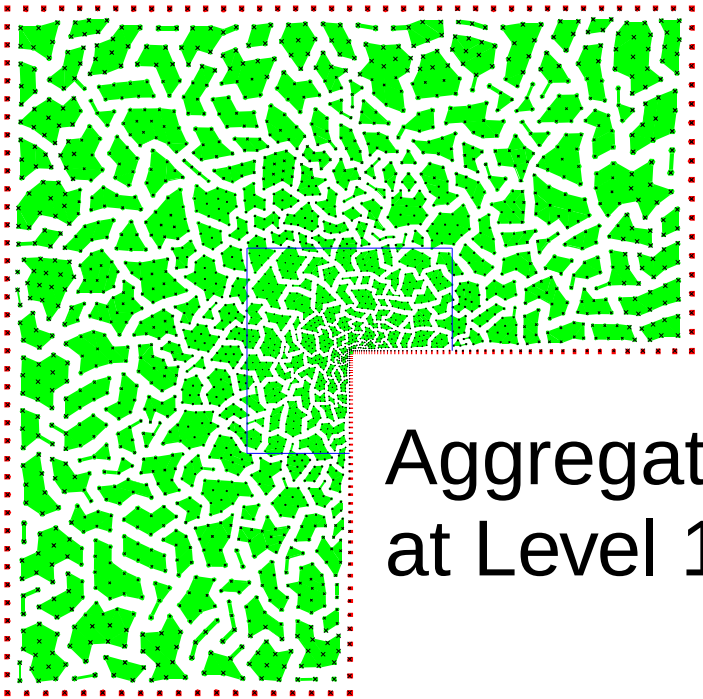


Zoom:

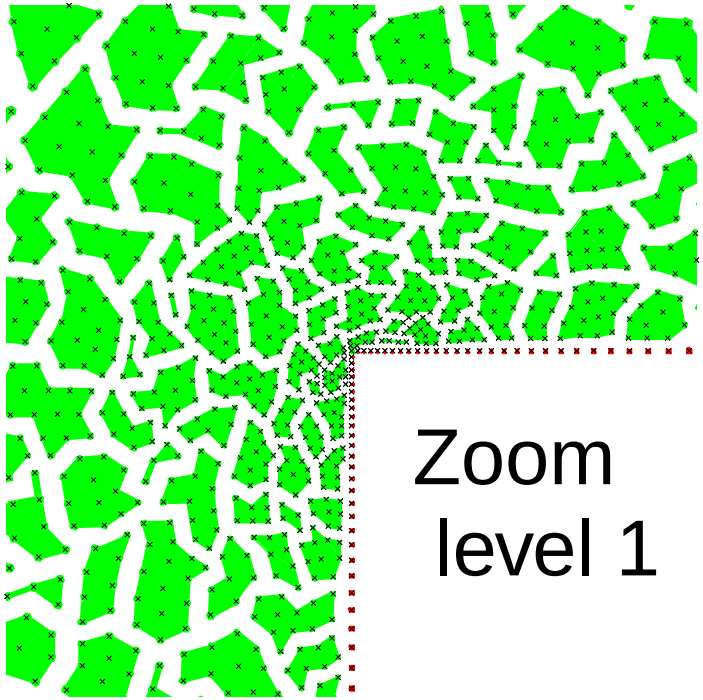


6. Quality aware aggregation (13)

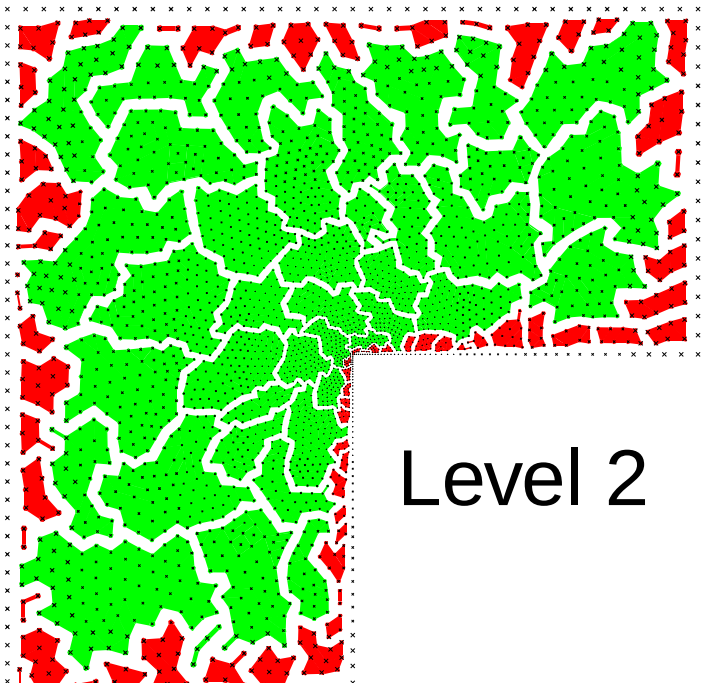
ULB



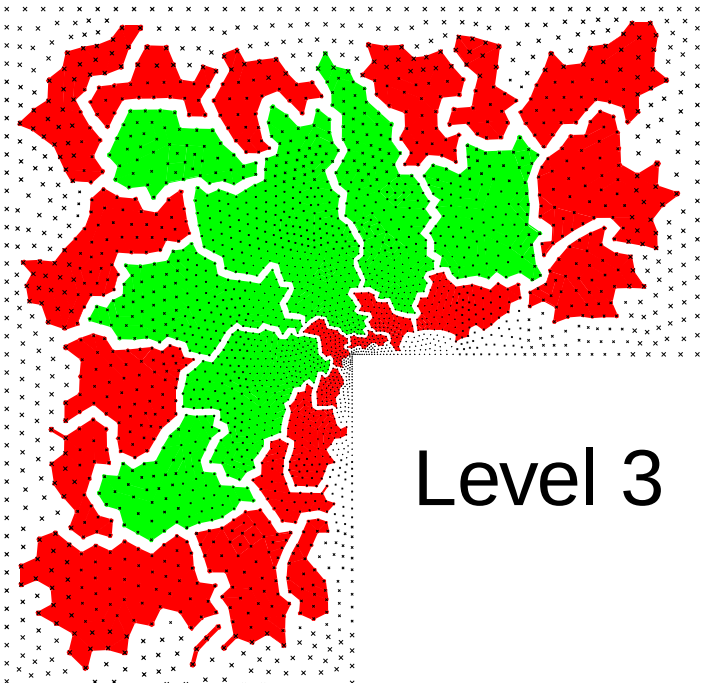
Aggregation at Level 1



Zoom level 1



Level 2



Level 3

6. Quality aware aggregation (14)

ULB

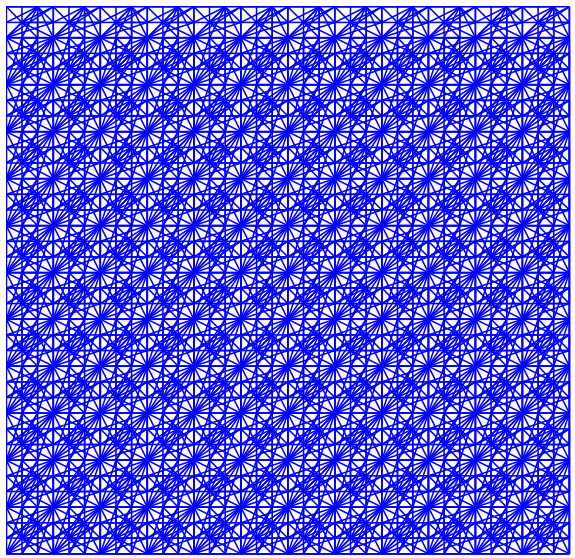
Aggregation works also for higher order FE matrices

Example:

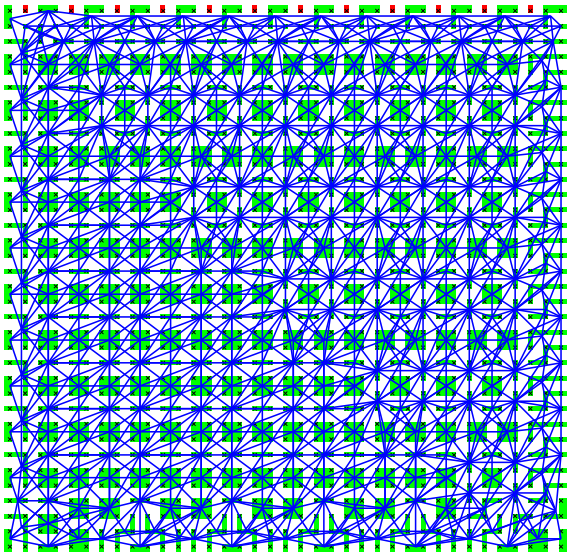
3rd order (P3)

$nnz(A) \approx 16\,n$

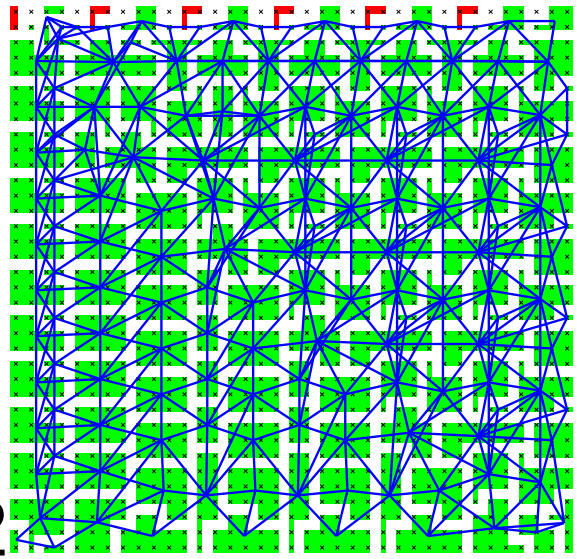
Fine grid



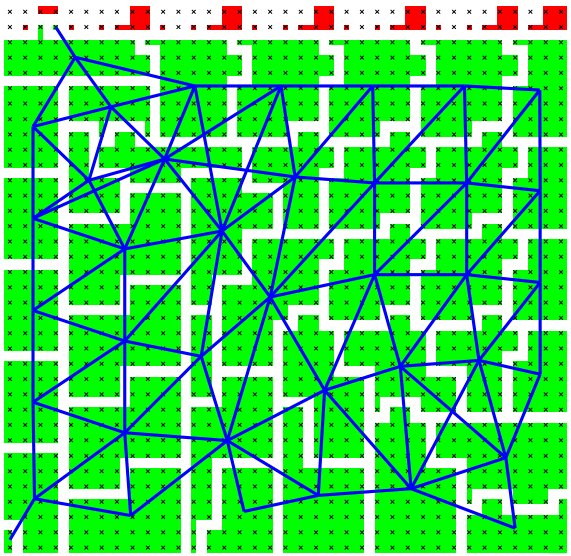
Level 1



Level 2



Level 3



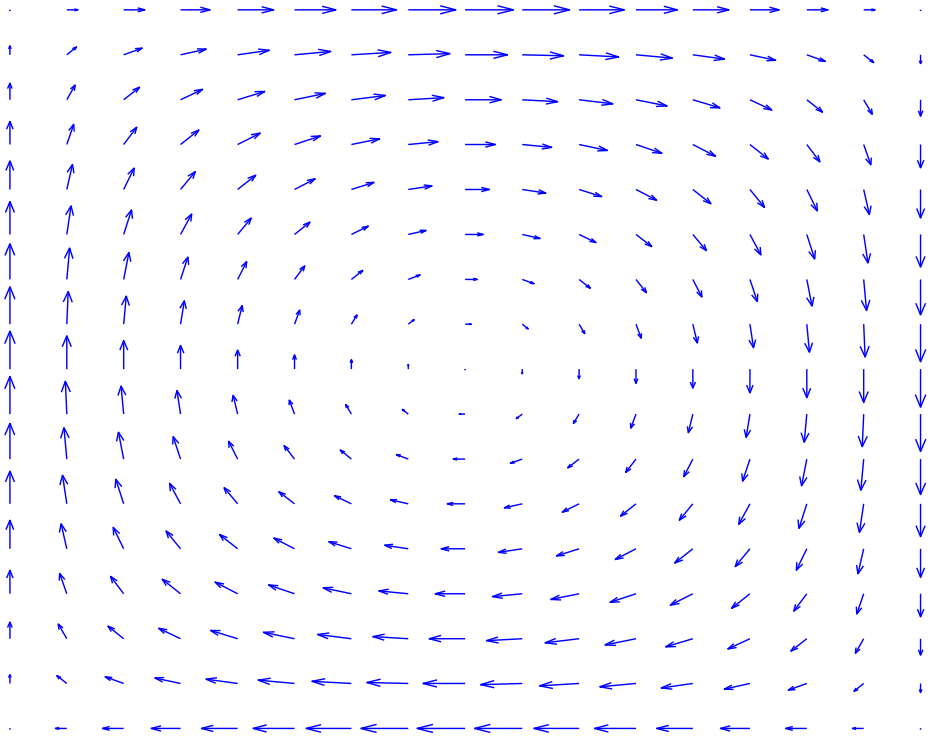
6. Quality aware aggregation (15)

ULB

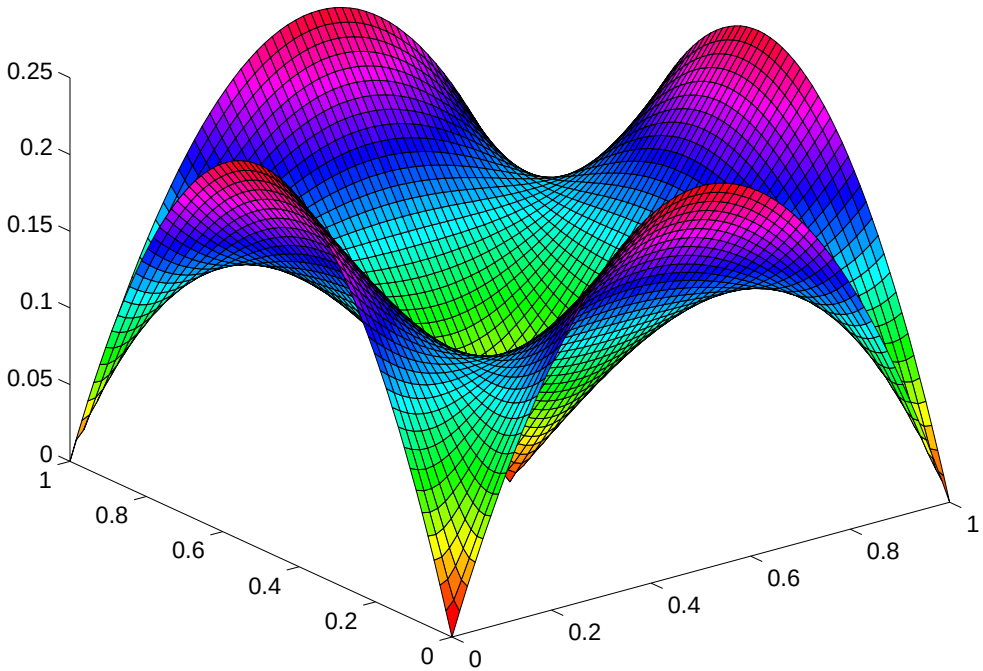
Upwind FD approximation of

$$-\nu\,\Delta u \;+\; \overline{v}\cdot \mathbf{grad}(u) \;=\; f \quad \text{in } \Omega \;=\; \text{unit square}$$

with $u \;=\; g$ on $\partial\Omega$, $\overline{v}(x\,,\,y) = \begin{pmatrix} x(1-x)(2\,y-1) \\ -(2\,x-1)y(1-y) \end{pmatrix}$:



Direction of the flow



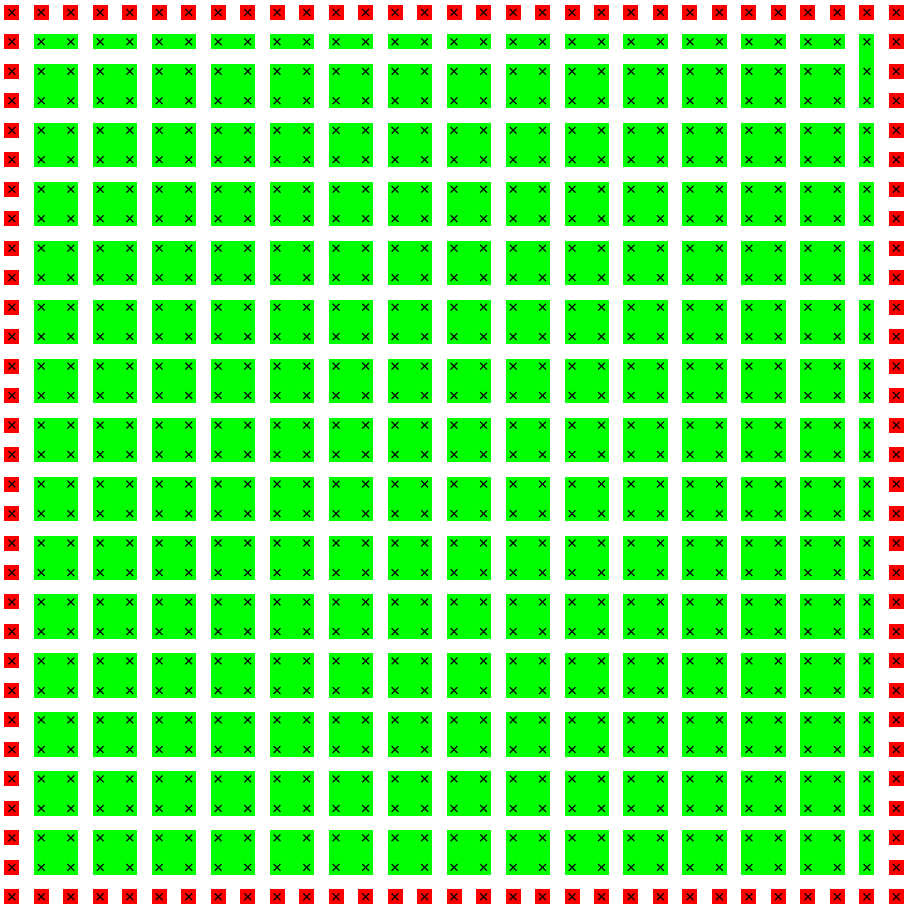
Magnitude

6. Quality aware aggregation (16)

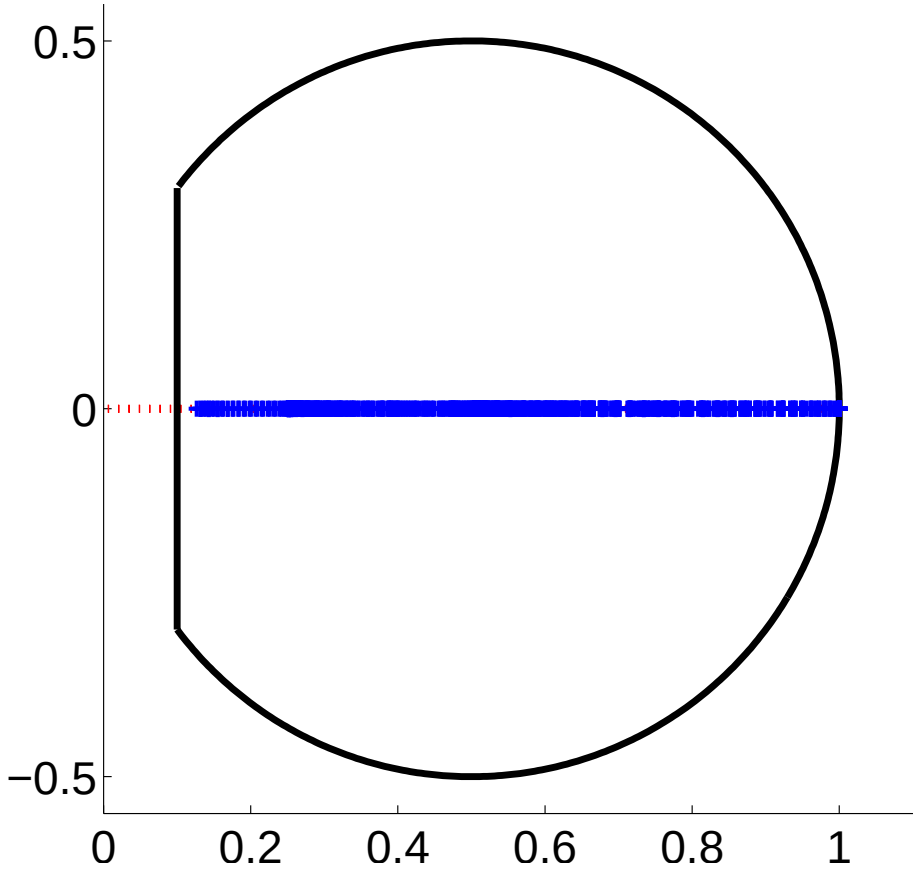
ULB

$\nu = 1$: diffusion dominating (near symmetric)

Aggregation



Spectrum



+ : $\sigma(I - S_{J_\omega} C)$ — : theory

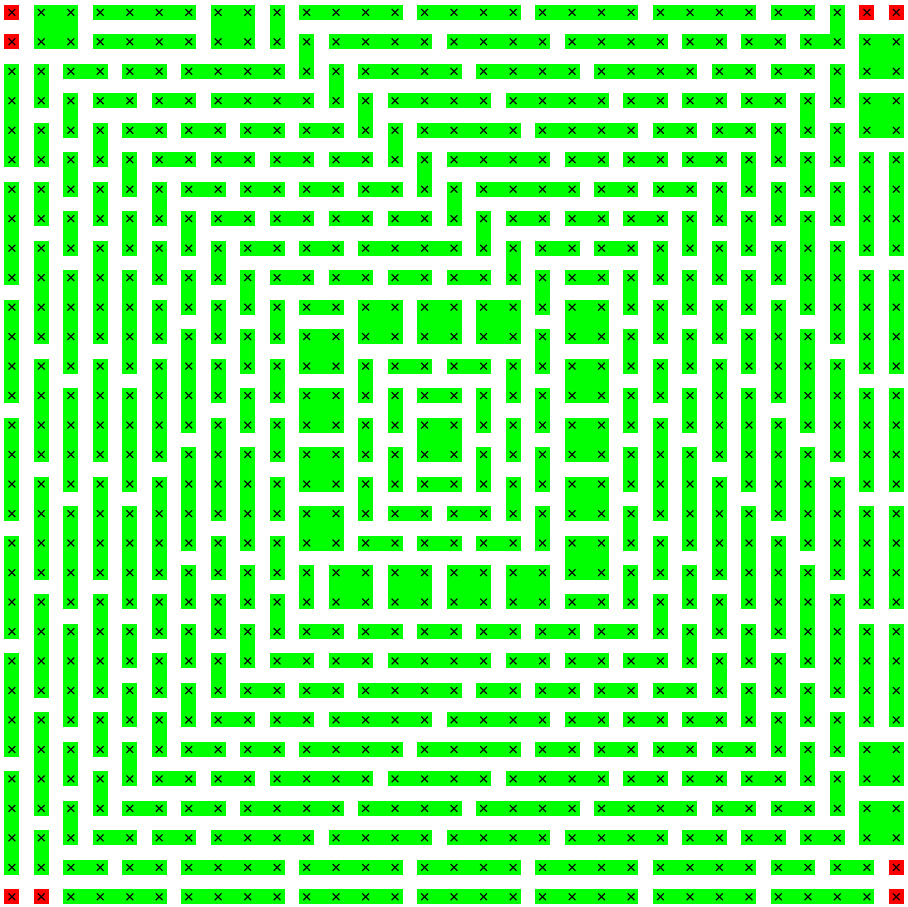
···· : $\sigma(\omega\, D^{-1} A)$ (convex hull)

6. Quality aware aggregation (17)

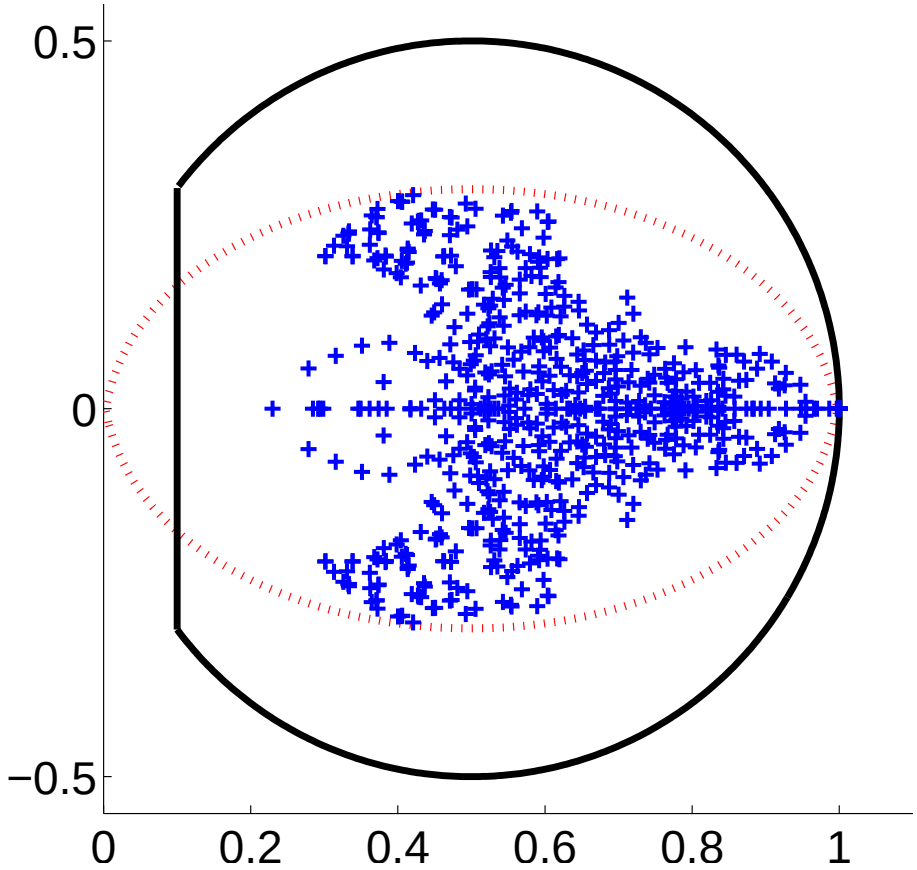
ULB

$\nu = 10{-}3$: convection dominating (strongly nonsymmetric)

Aggregation



Spectrum



$+$: $\sigma(I - S_{J_\omega} C)$ — : theory

···· : $\sigma(\omega\, D^{-1} A)$ (convex hull)

7. Performance (1)

ULB

Classical AMG talk on application

- Description of the application (beautiful pictures)
- Description of the AMG strategy and needed tuning
- Numerical results often not fully informative:
  - no robustness study on a comprehensive test suite;
  - no comparison with state of the art competitors.

Presentation of AGMG performance

- Most applications ran by people downloading the code. Some of those I am aware of: CFD, electrocardiology (we don’t have the beautiful pictures at hand).

7. Performance (2)

ULB

Presentation of AGMG performance

- Most applications ran by people downloading the code. Some of those I am aware of: CFD, electrocardiology (we don’t have the beautiful pictures at hand).
- The code is used black box (adaptation neither sought nor needed)

7. Performance (2)

ULB

Presentation of AGMG performance

- Most applications ran by people downloading the code. Some of those I am aware of: CFD, electrocardiology (we don’t have the beautiful pictures at hand).
- The code is used black box (adaptation neither sought nor needed)
- We think the most important is the robustness on a comprehensive test suite

7. Performance (2)

ULB

Presentation of AGMG performance

- Most applications ran by people downloading the code. Some of those I am aware of: CFD, electrocardiology (we don’t have the beautiful pictures at hand).
- The code is used black box (adaptation neither sought nor needed)
- We think the most important is the robustness on a comprehensive test suite
- We are not afraid of the comparison with state of the art competitors

7. Performance (3)

ULB

Comparison with some other methods

- AMG(Hyp): a classical AMG method as implemented in the Hypre library(Boomer AMG)
- AMG(HSL): a classical AMG method as implemented in the HSL library
- ILUPACK: efficient threshold-based ILU preconditioner
- Matlab \: Matlab sparse direct solver (UMFPACK)

All methods but the last with Krylov subspace acceleration

(Iterations stopped when $\frac{\|\mathbf{r}_k\|}{\|\mathbf{r}_0\|} < 10^{-6}$)

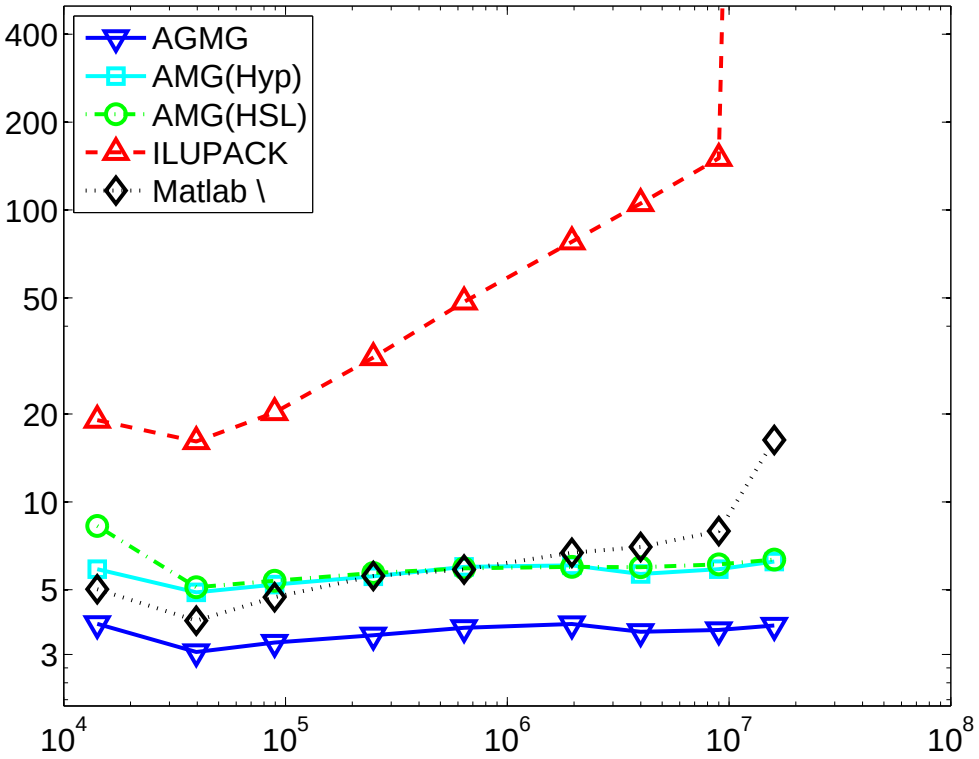
Quantity reported:

Total elapsed times in seconds (including set up) per 10^6 unknowns as a function of the number of unknowns (more unknowns yielded by grid refinement)

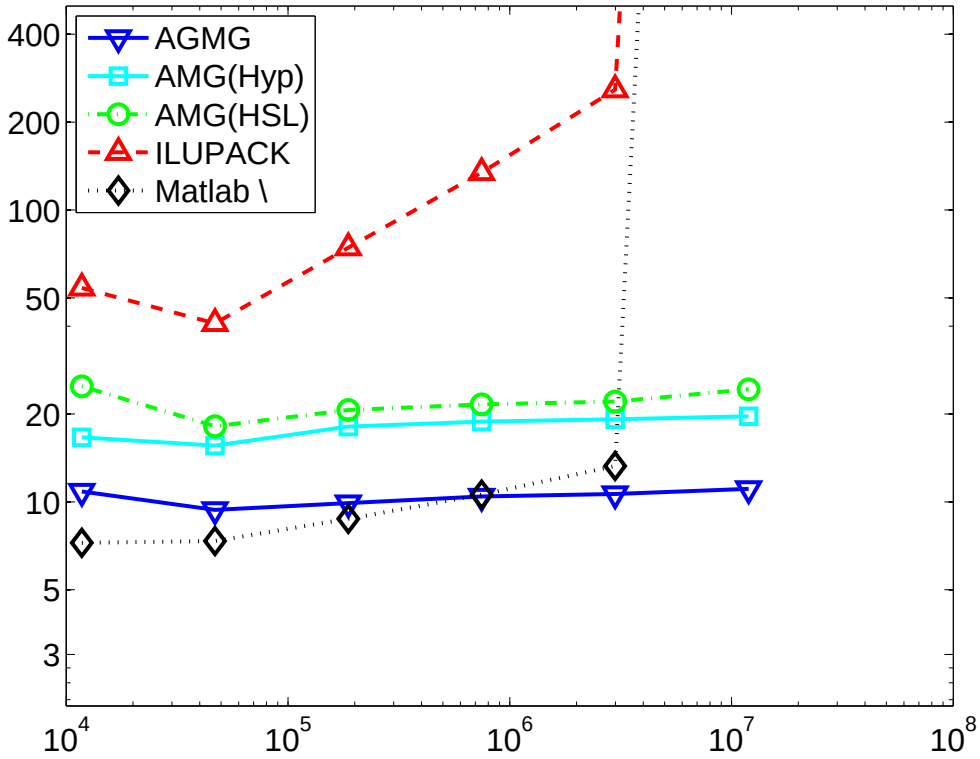
7. Performance (4)

ULB

Poisson 2D, FD



Laplace 2D, FE(P3)



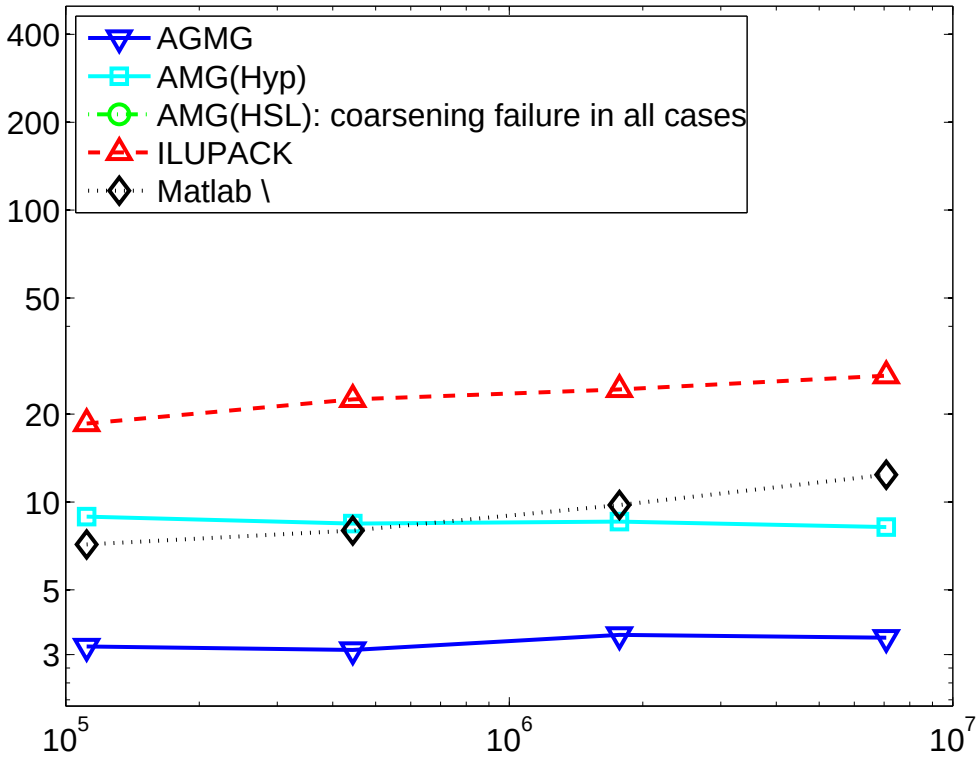
33% of nonzero offdiag > 0

7. Performance (5)

ULB

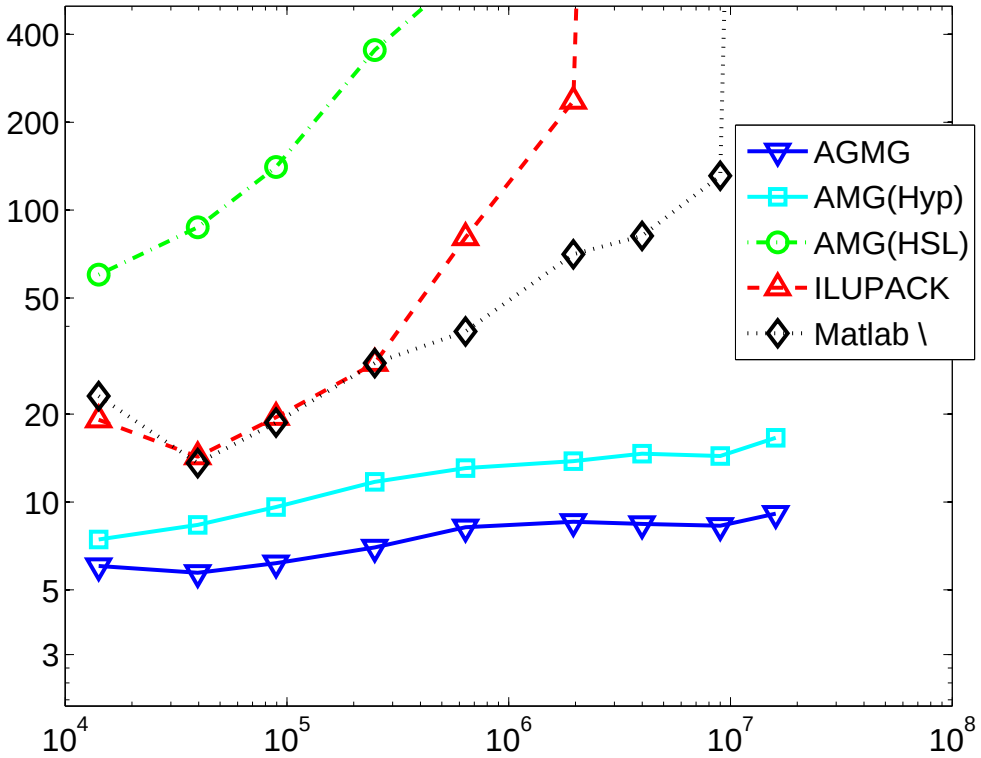
Poisson 2D, L-shaped, FE

Unstructured, Local refin.



Convection-Diffusion 2D, FD

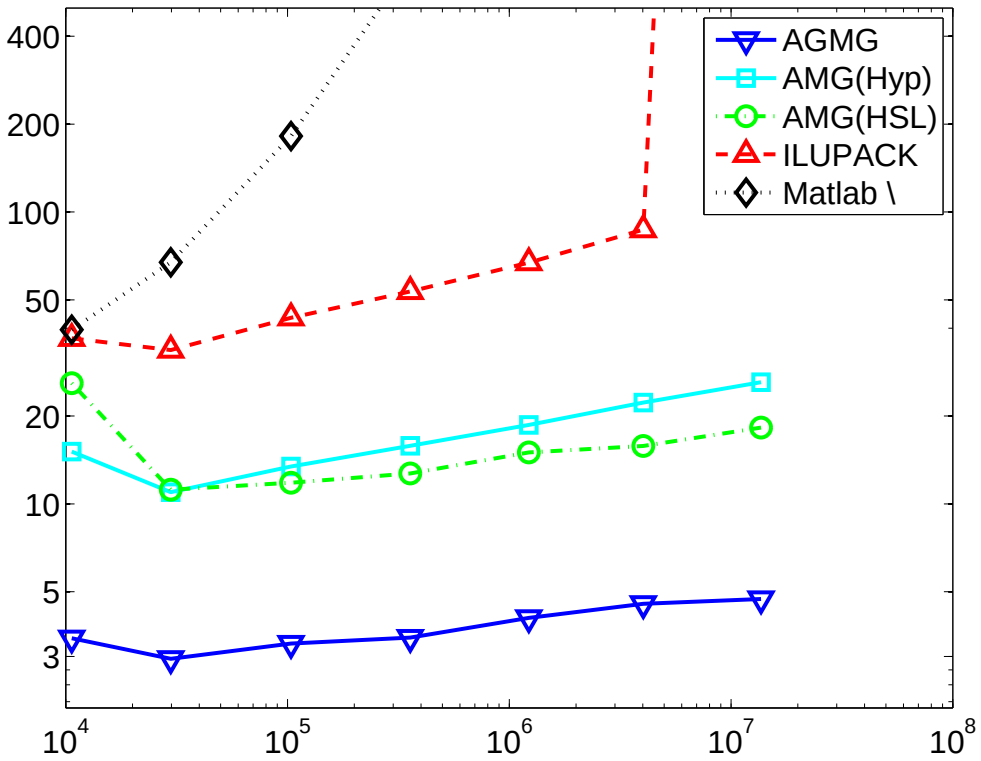
$\nu = 10^{-6}$



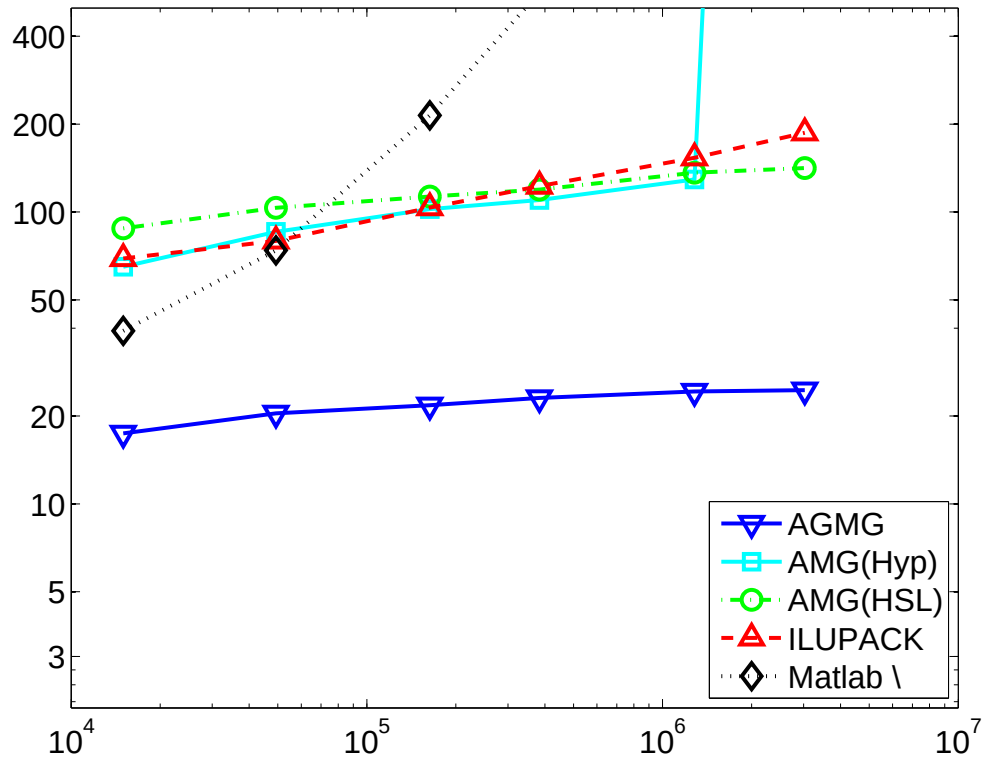
7. Performance (6)

ULB

Poisson 3D, FD



Laplace 3D, FE(P3)



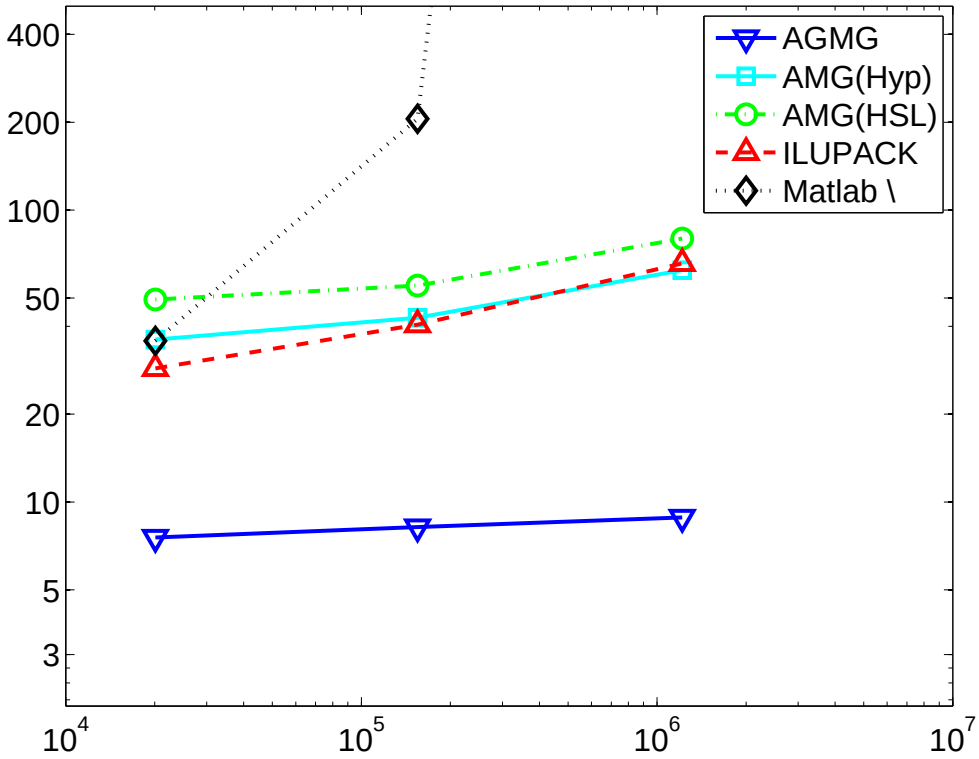
51% of nonzero offdiag > 0

7. Performance (7)

ULB

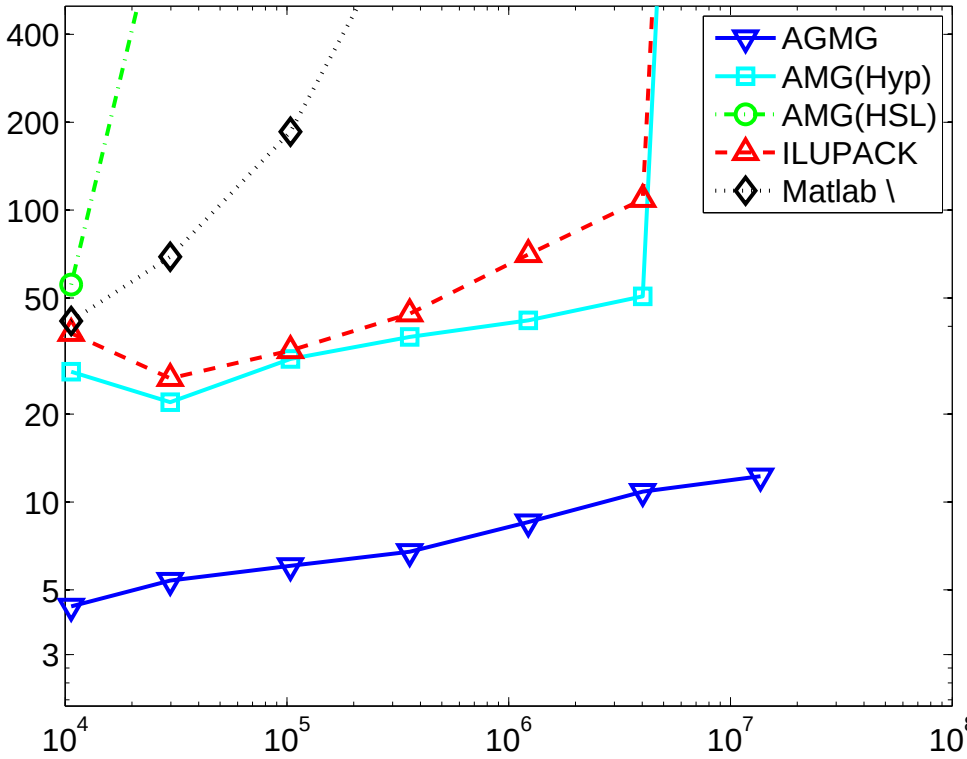
Poisson 3D, FE

Unstructured, Local refin.



Convection-Diffusion 3D, FD

$\nu = 10^{-6}$



7. Performance (8)

ULB

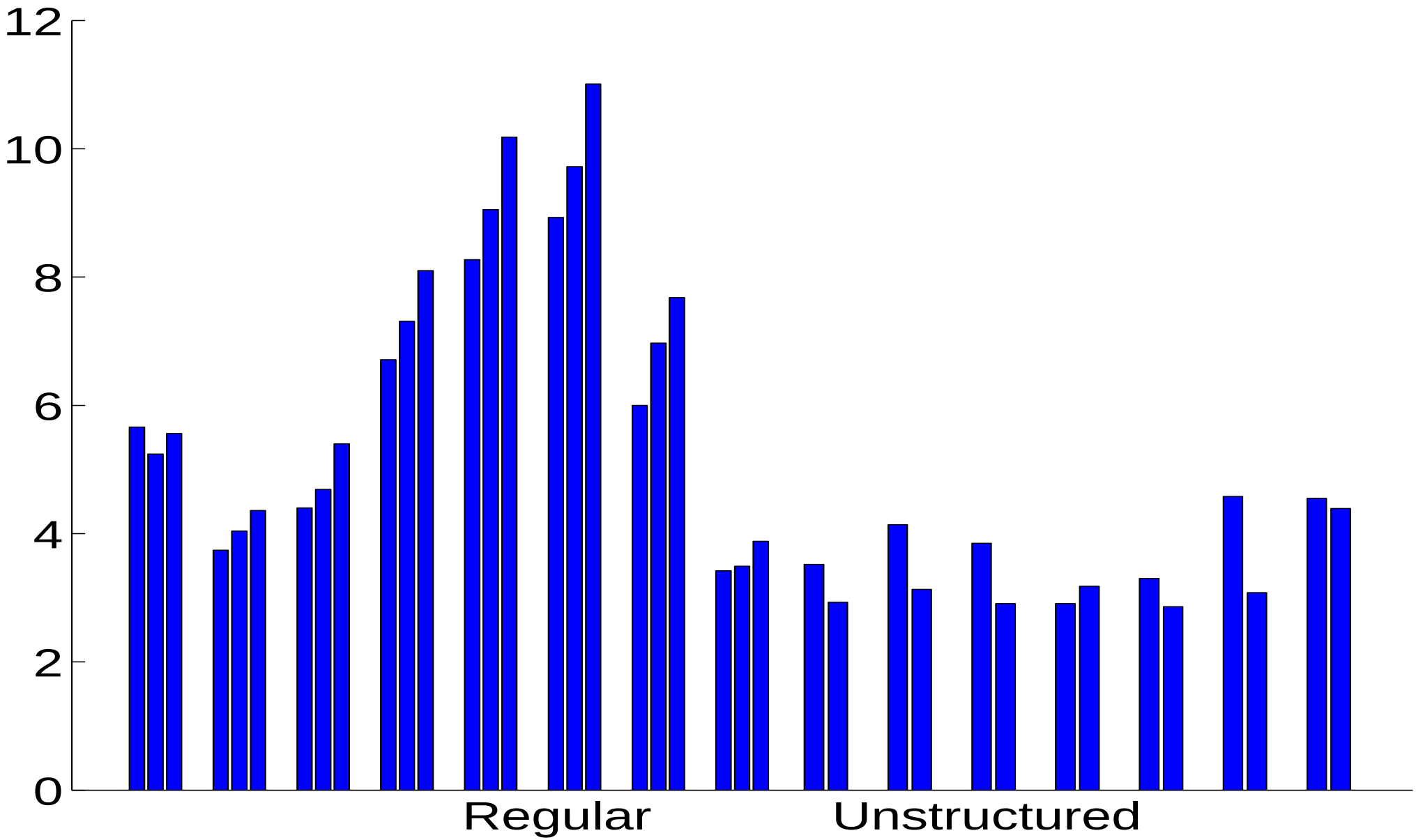
Robustness assessment

- Times reported are total elapsed times in seconds (including set up) per 10^6 unknowns
- Test suite: discrete scalar elliptic PDEs
  - SPD problems with jumps and all kind of anisotropy in the coefficients (some with reentering corner)
  - convection-diffusion problems with viscosity from $1 \rightarrow 10^{-6}$ and highly varying recirculating flow
  - FD on regular grids; 3 sizes:
    2D: $h^{-1} = 600\,,\ 1600\,,\ 5000$
    3D: $h^{-1} = 80\,,\ 160\,,\ 320$
  - FE on (un)structured meshes (with different levels of local refinement); 2 sizes: $n = 0.15e6 \rightarrow n = 7.1e6$

7. Performance (9)

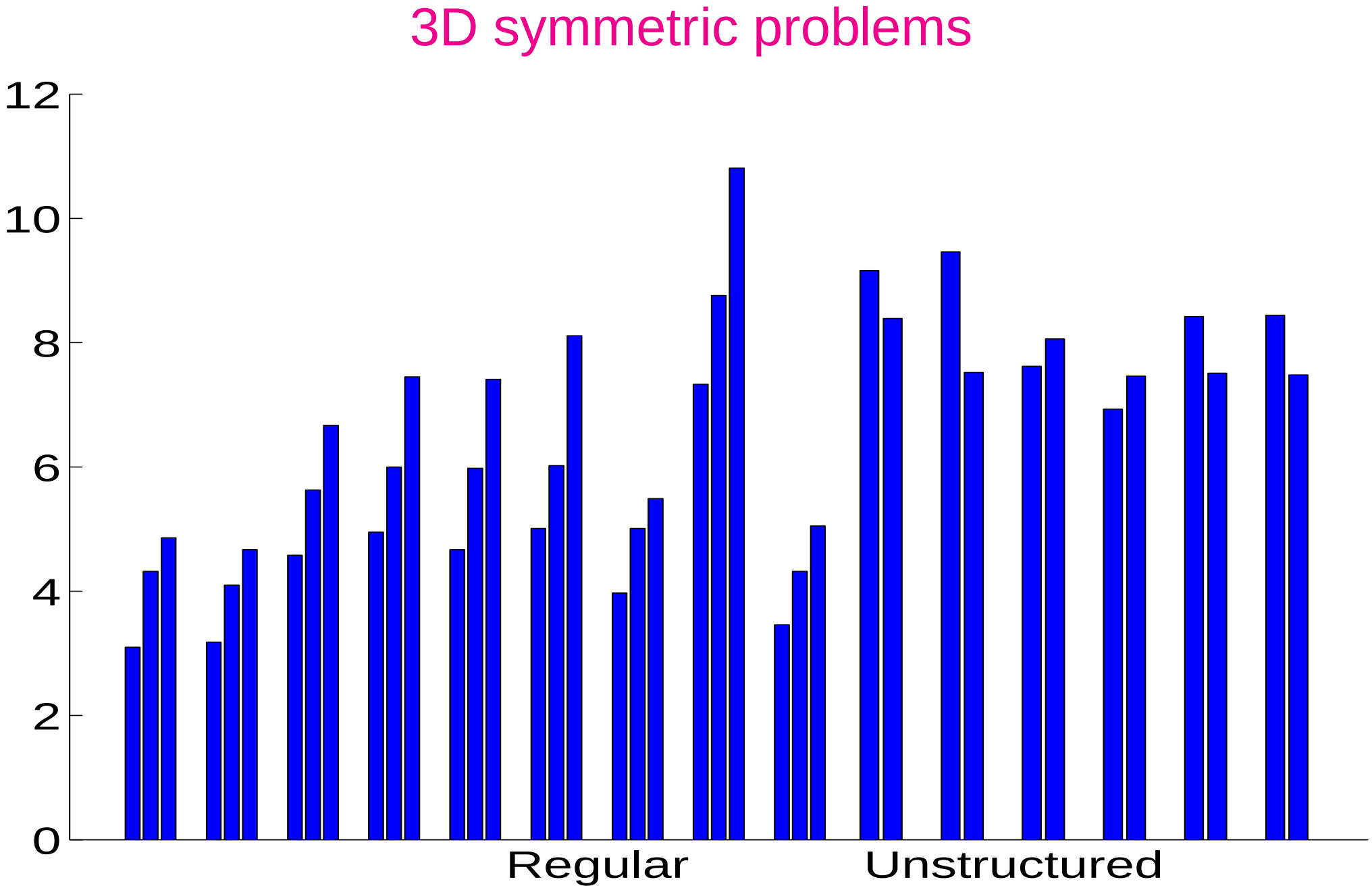
ULB

2D symmetric problems



7. Performance (10)

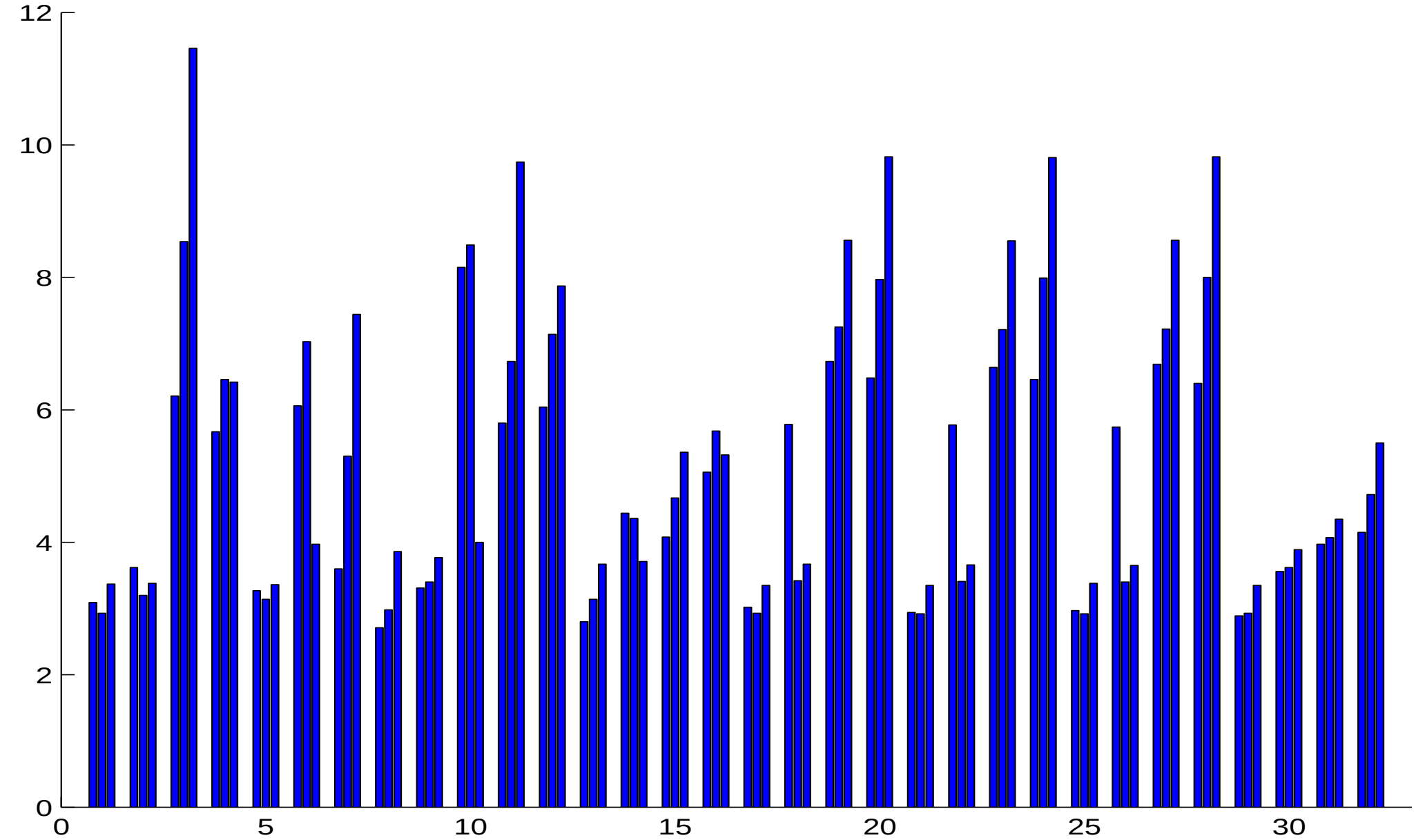
ULB



7. Performance (11)

ULB

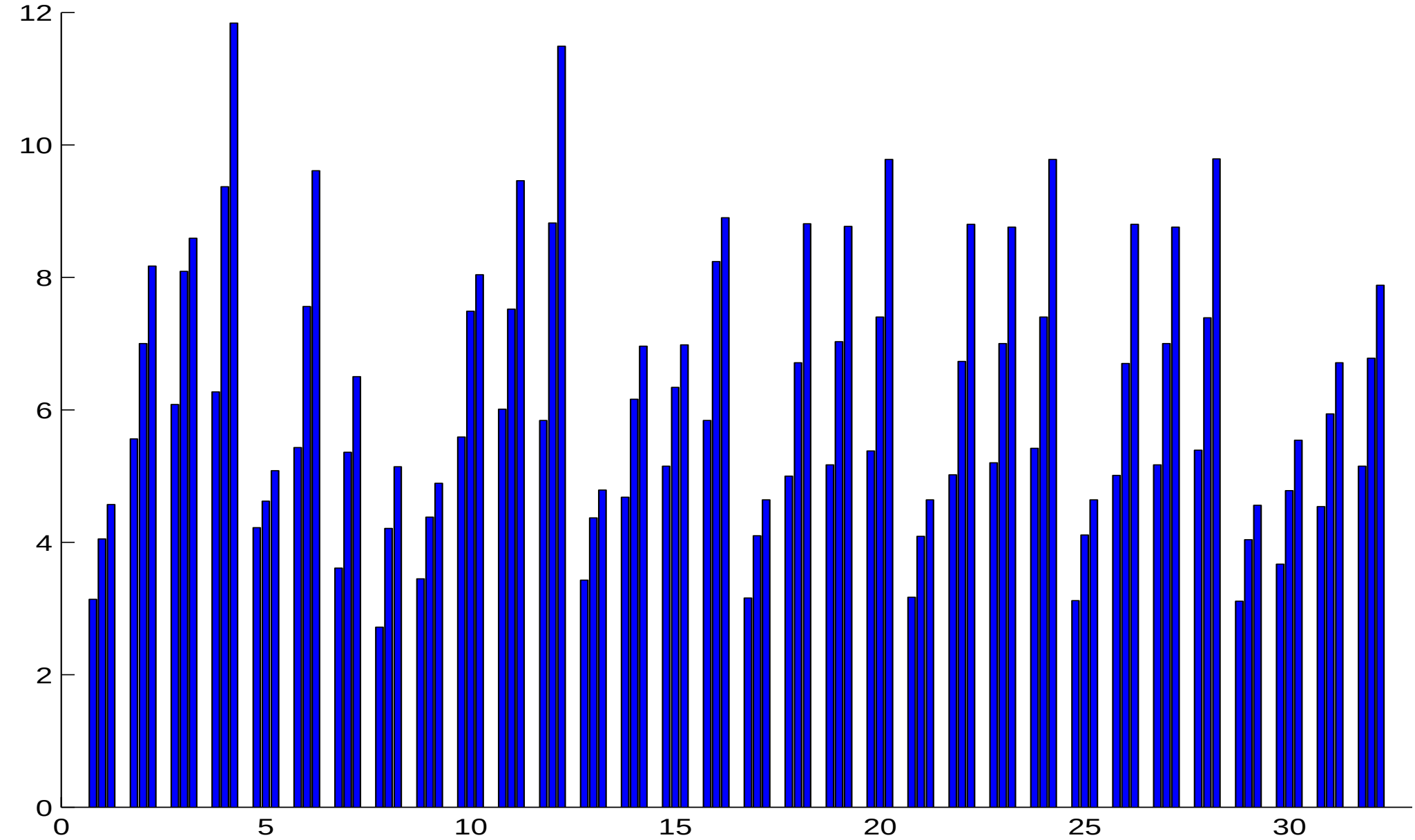
2D nonsymmetric problems



7. Performance (12)

ULB

3D nonsymmetric problems



8. Parallelization of AGMG (1)

ULB

Perspectives

- Good to start from the best sequential method

Any scalability curve should be put in perspective:
how much do we loose with respect the best state-of-the-art method on 1 core?

8. Parallelization of AGMG (1)

ULB

Perspectives

- Good to start from the best sequential method

  Any scalability curve should be put in perspective:
  how much do we loose with respect the best state-of-the-art method on 1 core?

- The faster the method,
  the more challenging its parallelization

  Less computation means less opportunity to overlap communications with computation

8. Parallelization of AGMG (2)

ULB

General parallelization strategy

- Partitioning of the unknowns
  $\rightarrow$ partitioning of matrix rows

8. Parallelization of AGMG (2)

ULB

General parallelization strategy

- Partitioning of the unknowns
  $\rightarrow$ partitioning of matrix rows
- Aggregation algorithm
  Unchanged, except that aggregates are only formed with unknowns in a same partition.
  $\rightarrow$ inherently parallel
  (use the local matrix rows, no communication needed except to form the next coarse grid matrix)

8. Parallelization of AGMG (2)

ULB

General parallelization strategy

- Partitioning of the unknowns
  $\rightarrow$ partitioning of matrix rows
- Aggregation algorithm
  Unchanged, except that aggregates are only formed with unknowns in a same partition.
  $\rightarrow$ inherently parallel
  (use the local matrix rows, no communication needed except to form the next coarse grid matrix)
- $\rightarrow$ Prolongation & Restriction are inherently parallel

8. Parallelization of AGMG (2)

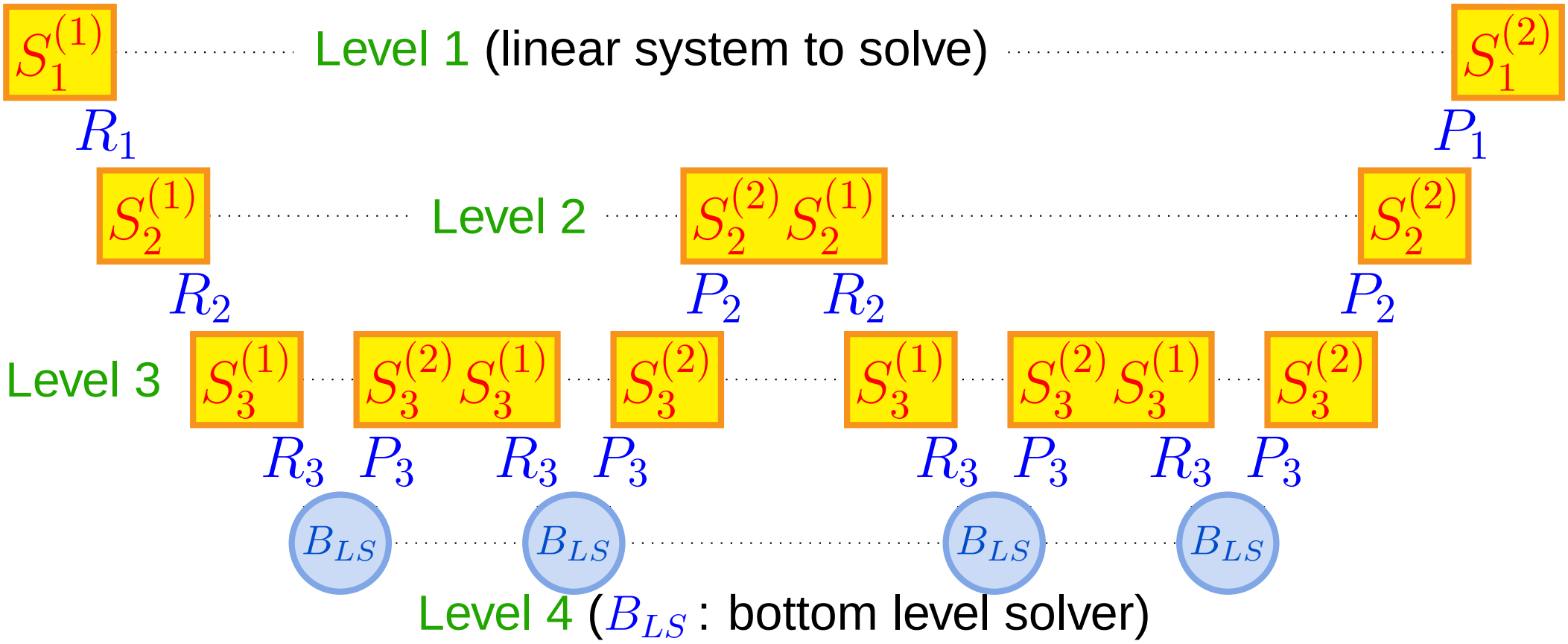
ULB

General parallelization strategy

- Partitioning of the unknowns
  $\rightarrow$ partitioning of matrix rows
- Aggregation algorithm
  Unchanged, except that aggregates are only formed with unknowns in a same partition.
  $\rightarrow$ inherently parallel
  (use the local matrix rows, no communication needed except to form the next coarse grid matrix)
- $\rightarrow$ Prolongation & Restriction are inherently parallel
- Relaxation: Gauss-Seidel, ignoring connections between different partitions $\rightarrow$ inherently parallel

8. Parallelization of AGMG (3)

ULB



$S_i^{(1)}$, $S_i^{(2)}$: Relaxation & Vector updates (|| : no comm.)
Matrix vector product (standard parallelization)
Inner products (global reduce: OK)

R_i , P_i : Grid transfer (|| : no comm.)

B_{LS} : Bottom level solver, initially MUMPS (Tricky)

8. Parallelization of AGMG (4)

ULB

Parallel AGMG: version 3.2.0

Poisson 3D on IBM BG/Q

(16 processes per node)

#p	$\frac{n}{10^6}$	it.	$\mathcal{T}_{\text{su}}$	$\mathcal{T}_{\text{sol}}$ ($\mathcal{T}_{\text{bl}}$)	$\mathcal{T}_{\text{tot}}$
16	43.6	11	27.4	26.6 (1.8)	54.0
64	175.6	11	28.0	29.5 (4.2)	57.5
512	1404.9	11	30.2	59.1 (33.6)	89.3

8. Parallelization of AGMG (5)

ULB

Algorithm redesign

- Only four levels, whatever problem size

Algorithm redesign

- Only four levels, whatever problem size
- Then the bottom level system has about 500 times less unknowns than the initial fine grid system
  $\rightarrow$ still very large

8. Parallelization of AGMG (5)

Algorithm redesign

- Only four levels, whatever problem size
- Then the bottom level system has about 500 times less unknowns than the initial fine grid system
  $\rightarrow$ still very large
- Thus: Iterative bottom level solver

8. Parallelization of AGMG (5)

ULB

Algorithm redesign

- Only four levels, whatever problem size
- Then the bottom level system has about 500 times less unknowns than the initial fine grid system
  $\rightarrow$ still very large
- Thus: Iterative bottom level solver
- 500 times less
  $\rightarrow$ Need not be as fast per unknown as AGMG

8. Parallelization of AGMG (5)

ULB

Algorithm redesign

- Only four levels, whatever problem size
- Then the bottom level system has about 500 times less unknowns than the initial fine grid system
  $\rightarrow$ still very large
- Thus: Iterative bottom level solver
- 500 times less
  $\rightarrow$ Need not be as fast per unknown as AGMG
- But has to scale very well in parallel
  (despite smaller problem size)

8. Parallelization of AGMG (6)

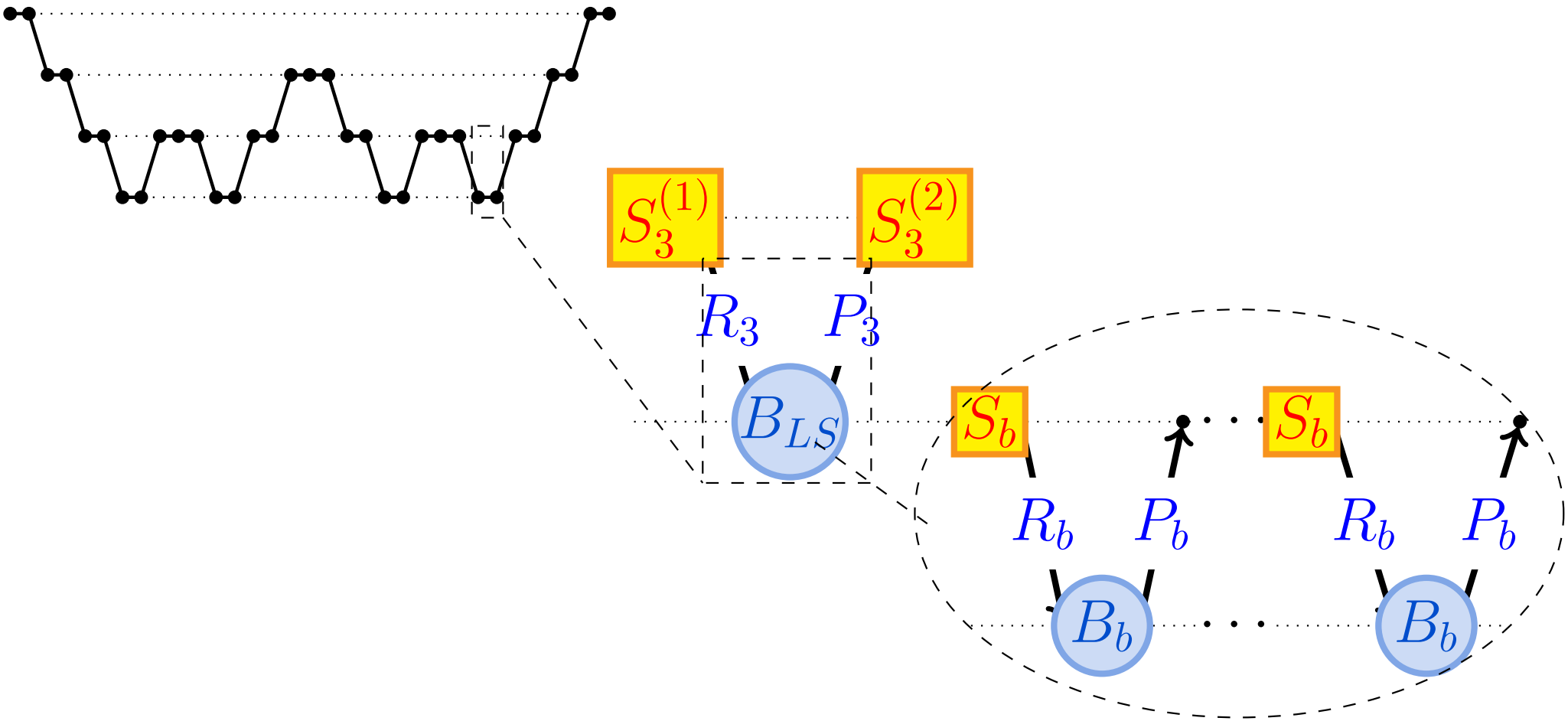
ULB

Iterative bottom level solver

- Aggregation-based two-grid method
  (one further level: very coarse grid)
- All unknowns on a same process form 1 aggregate
  (very coarse grid: size = number of processes (cores))
- Better smoother:
  apply sequential AGMG to the local part of the matrix
- Very coarse grid system
  - if still too large, solved in parallel
    within subgroups of processes
  - the solver is AGMG again
    (either sequential or parallel)

8. Parallelization of AGMG (7)

ULB



S_b : sequential AGMG applied to “local” part of the matrix

B_b : sequential AGMG (512 cores or less) or
parallel AGMG in subgroups (more than 512 cores)

8. Parallelization of AGMG (8)

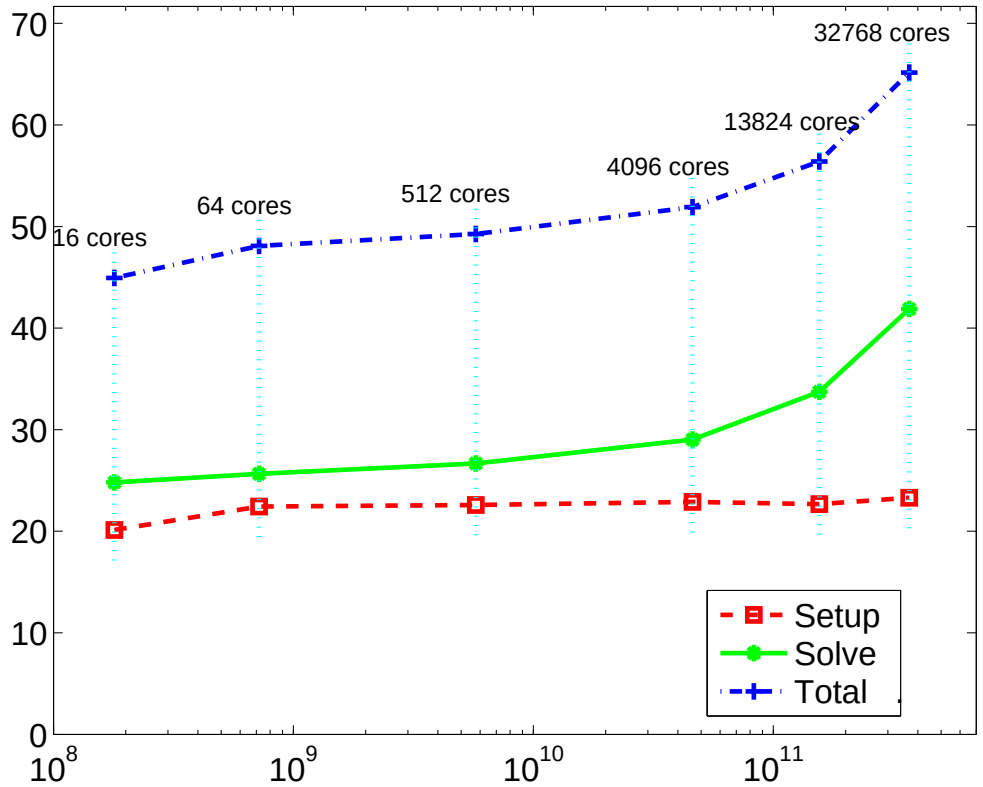
ULB

Results: the magic works

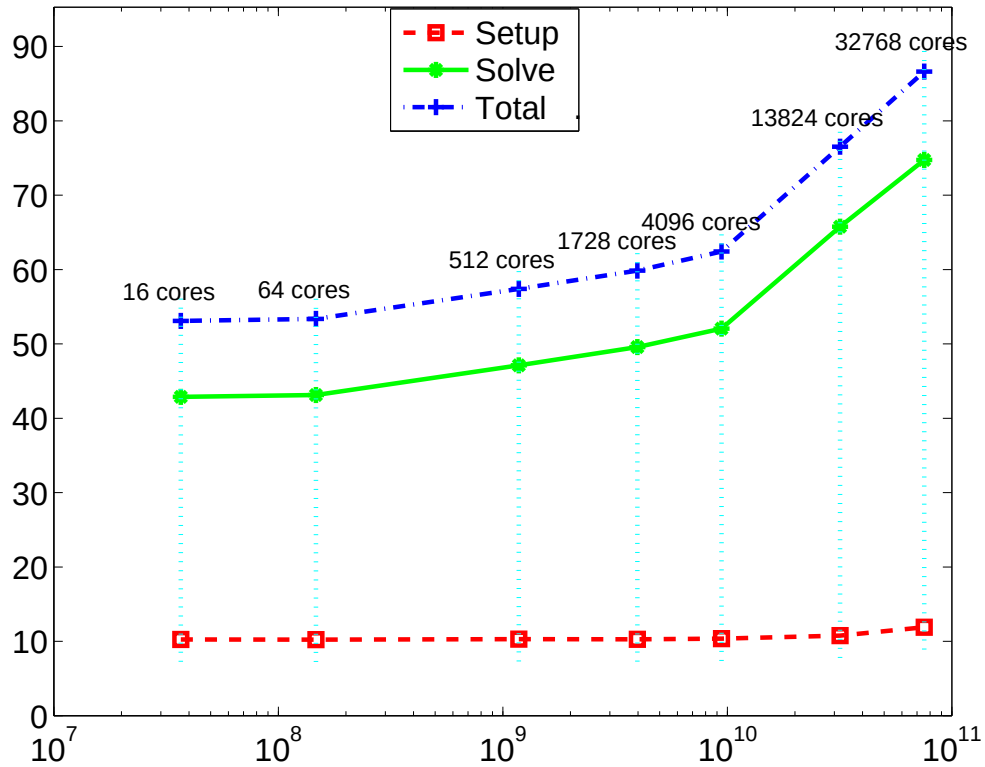
Weak scalability on CURIE (Intel Farm) for 3D Poisson

Elapsed time (seconds) – vs – number of unknowns

Finite Difference



P3 Finite Elements



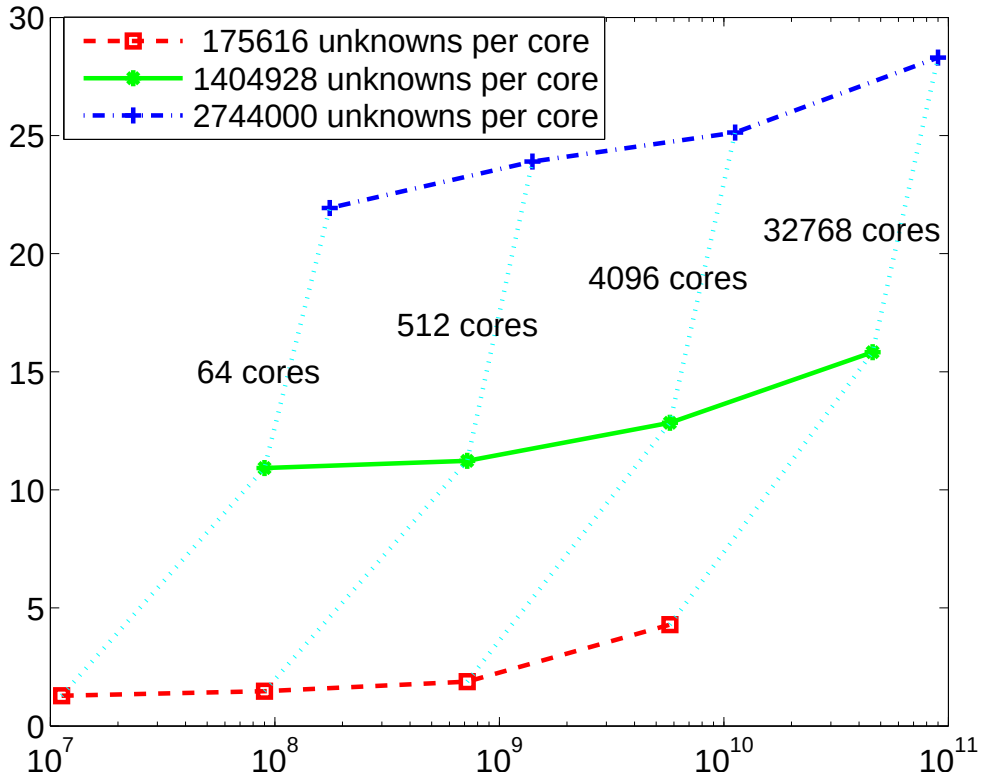
8. Parallelization of AGMG (9)

ULB

3D Poisson (Finite Difference) on HERMIT (Cray XE6)

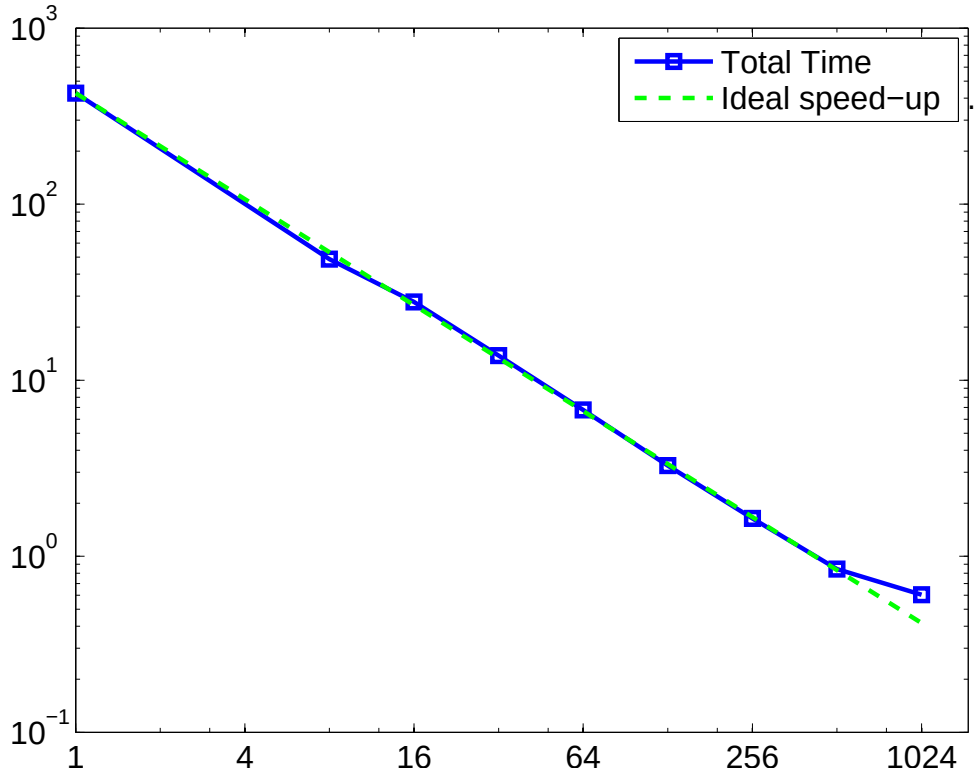
Weak scalability

Time – vs – # unknowns



Strong scalability

Time – vs – number of cores

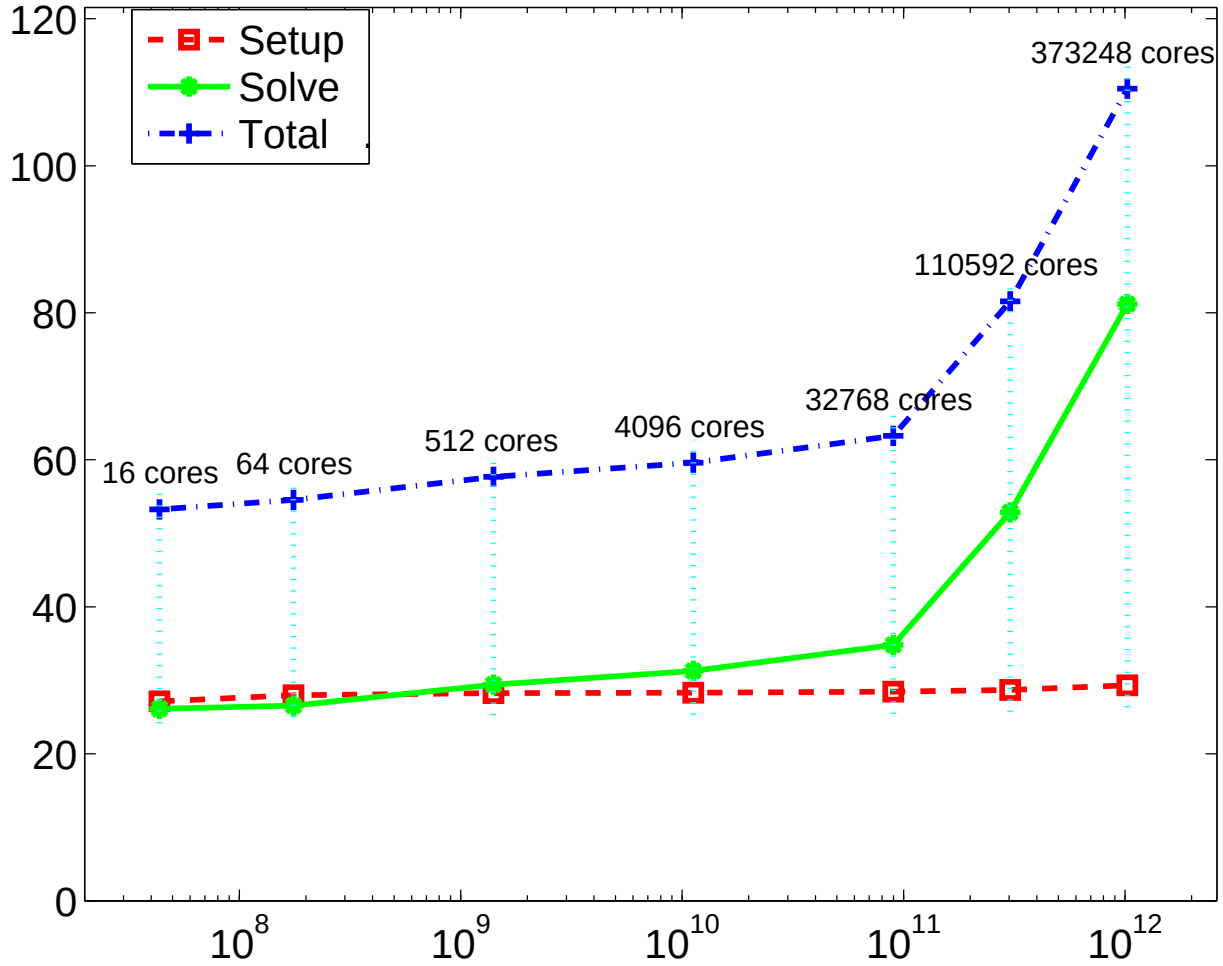


8. Parallelization of AGMG (10)

ULB

Weak scalability on JUQUEEN (IBM BG/Q) for 3D Poisson (Finite Difference)

Elapsed time – vs – number of unknowns



9. Conclusions

ULB

Some key ideas

- Reuse AMG paradigms, but keep P as simple as possible (at most one nonzero per row)

9. Conclusions

ULB

Some key ideas

- Reuse AMG paradigms, but keep P as simple as possible (at most one nonzero per row)
- Possibly slower convergence compensated by
  - low cost per iteration
  - Krylov accelaration everywhere

9. Conclusions

ULB

Some key ideas

- Reuse AMG paradigms, but keep P as simple as possible (at most one nonzero per row)
- Possibly slower convergence compensated by
  - low cost per iteration
  - Krylov accelaration everywhere
- Benefit of “simple” P :

9. Conclusions

ULB

Some key ideas

- Reuse AMG paradigms, but keep P as simple as possible (at most one nonzero per row)
- Possibly slower convergence compensated by
  - low cost per iteration
  - Krylov accelaration everywhere
- Benefit of “simple” P :
  - Explicit control of the two-grid convergence rate

9. Conclusions

ULB

Some key ideas

- Reuse AMG paradigms, but keep P as simple as possible (at most one nonzero per row)
- Possibly slower convergence compensated by
  - low cost per iteration
  - Krylov accelaration everywhere
- Benefit of "simple" P :
  - Explicit control of the two-grid convergence rate
  - Connectivity pattern on the coarse grids:
    similar or sparser than that on the fine grid
    ($\rightarrow$ no complexity issue with the recursive use)

9. Conclusions

ULB

Some key ideas

- Reuse AMG paradigms, but keep P as simple as possible (at most one nonzero per row)
- Possibly slower convergence compensated by
  - low cost per iteration
  - Krylov accelaration everywhere
- Benefit of "simple" P :
  - Explicit control of the two-grid convergence rate
  - Connectivity pattern on the coarse grids:
    similar or sparser than that on the fine grid
    ($\rightarrow$ no complexity issue with the recursive use)
  - Easy parallelization
    (At large scale, need clever coarsest grid solver)

Some references

ULB

Two-grid convergence theory

- Algebraic analysis of two-grid methods: the nonsymmetric case, NLAA (2010)
- Algebraic theory of two-grid methods (NTMA, to appear – review paper)

The K-cycle

- Recursive Krylov-based multigrid cycles (with P. S. Vassilevski), NLAA (2008)

Two-grid analysis of aggregation methods

- Analysis of aggregation–based multigrid (with A. C. Muresan), SISC (2008)
- Algebraic analysis of aggregation-based multigrid, (with A. Napov) NLAA (2011)

AGMG and quality aware aggregation

- An aggregation-based algebraic multigrid method, ETNA (2010).
- An algebraic multigrid method with guaranteed convergence rate (with A. Napov), SISC (2012)
- Aggregation-based algebraic multigrid for convection-diffusion equations, SISC (2012)
- Algebraic multigrid for moderate order finite elements (with A. Napov), SISC (2014)

Parallelization

- A massively parallel solver for discrete Poisson-like problems, Tech. Rep. (2014)

Some references

!! Thank you !!

ULB

Two-grid convergence theory

- Algebraic analysis of two-grid methods: the nonsymmetric case, NLAA (2010)
- Algebraic theory of two-grid methods (NTMA, to appear – review paper)

The K-cycle

- Recursive Krylov-based multigrid cycles (with P. S. Vassilevski), NLAA (2008)

Two-grid analysis of aggregation methods

- Analysis of aggregation–based multigrid (with A. C. Muresan), SISC (2008)
- Algebraic analysis of aggregation-based multigrid, (with A. Napov) NLAA (2011)

AGMG and quality aware aggregation

- An aggregation-based algebraic multigrid method, ETNA (2010).
- An algebraic multigrid method with guaranteed convergence rate (with A. Napov), SISC (2012)
- Aggregation-based algebraic multigrid for convection-diffusion equations, SISC (2012)
- Algebraic multigrid for moderate order finite elements (with A. Napov), SISC (2014)

Parallelization

- A massively parallel solver for discrete Poisson-like problems, Tech. Rep. (2014)