Aggregation-based algebraic multigrid

from theory to fast solvers

Yvan Notay*

Université Libre de Bruxelles
Service de Métrologie Nucléaire

CEMRACS, Marseille, July 18, 2012

* Supported by the Belgian FNRS
http://homepages.ulb.ac.be/∼ynotay

Outline

ULB

1. Introduction
2. AMG preconditioning and K-cycle
3. Two-grid analysis
4. Aggregation procedure
   - Repeated pairwise aggregation
5. Multi-level analysis
6. Parallelization
7. Numerical results
8. Conclusions

1. Introduction

ULB

Ubiquitous need:

Efficient methods to solve large sparse linear systems

In many cases, the design of an appropriate iterative linear solver is much easier if one has at hand a library able to efficiently solve linear (sub)systems

$$A\,\mathbf{u} = \mathbf{b}$$

where A corresponds to the discretization of

$$-\text{div}(D\,\textbf{grad}(u)) + \mathbf{v}\,\textbf{grad}(u) + c\,u = f \qquad (+B.C.)$$

(or closely related).

Efficiently:

robustly (stable performances)

in linear time: $\frac{\text{elapsed}}{n \times \#\text{proc}}$ roughly constant

1. Introduction

ULB

- From Martin Gander talk:
  Krylov subspace method needed for robustness

1. Introduction

- From Martin Gander talk:
  Krylov subspace method needed for robustness
- From Ulrich Rüde talk:
  multigrid needed for linear time

1. Introduction

ULB

- From Martin Gander talk:
  Krylov subspace method needed for robustness
- From Ulrich Rüde talk:
  multigrid needed for linear time
- Here: use multigrid as a preconditioner for Krylov
  (combine multigrid with Krylov acceleration)

1. Introduction

ULB

- From Martin Gander talk:
  Krylov subspace method needed for robustness
- From Ulrich Rüde talk:
  multigrid needed for linear time
- Here: use multigrid as a preconditioner for Krylov
  (combine multigrid with Krylov acceleration)

Why algebraic multigrid (AMG)?

- Geometric multigrid: needs a predefined set of grids
- AMG attempts to obtain the same effect using only the information present in the system matrix A

  (Reminder: effect = efficient damping of “smooth” error components, that can be seen only from large scale)

1. Introduction

ULB

Two-grid Algorithm

(as in Ulrich Rüde talk)

(1) Relax several times on grid h, obtaining $\widetilde{u}^h$ with a smooth corresponding error

(2) Calculate the residual:
$r^h = f^h - L^h \widetilde{\mathbf{u}}^h$

(3) Solve approximate error-equation on the coarse grid:
$L^H v^H = f^H \equiv I_h^H r^h$

(4) Interpolate and add correction: $\widetilde{u}^h \leftarrow \widetilde{u}^h + I_H^h v^H$

(5) Relax again on h

1. Introduction

ULB

Two-grid Algorithm

(as in Ulrich Rüde talk)

(1) Relax several times on grid h , obtaining $\widetilde{u}^h$ with a smooth corresponding error

(2) Calculate the residual: $r^h = f^h - L^h \widetilde{\mathbf{u}}^h$

(3) Solve approximate error-equation on the coarse grid: $L^H v^H = f^H \equiv I_h^H r^h$

(4) Interpolate and add correction: $\widetilde{u}^h \leftarrow \widetilde{u}^h + I_H^h v^H$

(5) Relax again on h

(algebraic notation for linear system $A\,\mathbf{u} = \mathbf{b}$ with smoother M ; $\mathbf{u}_k$ is the current approximation)

(1) $\widetilde{\mathbf{u}} = \mathbf{u}_k + M^{-1}(\mathbf{b} - A\,\mathbf{u}_k)$

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

(3) $A_c \mathbf{v}_c = \mathbf{r}_c \equiv R\,\widetilde{\mathbf{r}}$
(R : restriction, $n_c \times n$)

(4) $\widetilde{\mathbf{u}} \leftarrow \widetilde{\mathbf{u}} + P\,\mathbf{v}_c$
(P : prolongation, $n \times n_c$)

(5) $\mathbf{u}_{k+1} = \widetilde{\mathbf{u}} + M^{-1}(\mathbf{b} - A\,\widetilde{\mathbf{u}})$

1. Introduction

ULB

Two-grid Algorithm

(as in Ulrich Rüde talk)

(1) Relax several times on grid h, obtaining $\widetilde{u}^h$ with a smooth corresponding error

(2) Calculate the residual: $r^h = f^h - L^h \widetilde{\mathbf{u}}^h$

(3) Solve approximate error-equation on the coarse grid: $L^H v^H = f^H \equiv I_h^H r^h$

(4) Interpolate and add correction: $\widetilde{u}^h \leftarrow \widetilde{u}^h + I_H^h v^H$

(5) Relax again on h

(algebraic notation for linear system $A\,\mathbf{u} = \mathbf{b}$ with smoother M ; $\mathbf{u}_k$ is the current approximation)

(1) $\widetilde{\mathbf{u}} = \mathbf{u}_k + M^{-1}(\mathbf{b} - A\,\mathbf{u}_k)$

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

(3) $A_c \mathbf{v}_c = \mathbf{r}_c \equiv R\,\widetilde{\mathbf{r}}$
(R : restriction, $n_c \times n$)
$R = P^T$

(4) $\widetilde{\mathbf{u}} \leftarrow \widetilde{\mathbf{u}} + P\,\mathbf{v}_c$
(P : prolongation, $n \times n_c$)

(5) $\mathbf{u}_{k+1} = \widetilde{\mathbf{u}} + M^{-1}(\mathbf{b} - A\,\widetilde{\mathbf{u}})$

Two-grid Algorithm

(as in Ulrich Rüde talk)

(1) Relax several times on grid h, obtaining $\widetilde{u}^h$ with a smooth corresponding error

(2) Calculate the residual:

$r^h = f^h - L^h \widetilde{\mathbf{u}}^h$

(3) Solve approximate error-equation on the coarse grid:

$L^H v^H = f^H \equiv I_h^H r^h$

(4) Interpolate and add correction: $\widetilde{u}^h \leftarrow \widetilde{u}^h + I_H^h v^H$

(5) Relax again on h

(algebraic notation for linear system $A\,\mathbf{u} = \mathbf{b}$ with smoother M; $\mathbf{u}_k$ is the current approximation)

(1) $\widetilde{\mathbf{u}} = \mathbf{u}_k + M^{-1}(\mathbf{b} - A\,\mathbf{u}_k)$

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

(3) $A_c \mathbf{v}_c = \mathbf{r}_c \equiv R\,\widetilde{\mathbf{r}}$

(R: restriction, $n_c \times n$)

$R = P^T$, $A_c = R\,A\,P = P^T A\,P$

(4) $\widetilde{\mathbf{u}} \leftarrow \widetilde{\mathbf{u}} + P\,\mathbf{v}_c$

(P: prolongation, $n \times n_c$)

(5) $\mathbf{u}_{k+1} = \widetilde{\mathbf{u}} + M^{-1}(\mathbf{b} - A\,\widetilde{\mathbf{u}})$

1. Introduction

ULB

Two-grid Algorithm

(as in Ulrich Rüde talk)

(1) Relax several times on grid h, obtaining $\widetilde{u}^h$ with a smooth corresponding error

(2) Calculate the residual:
$r^h = f^h - L^h \widetilde{\mathbf{u}}^h$

(3) Solve approximate error-equation on the coarse grid:
$L^H v^H = f^H \equiv I_h^H r^h$

(4) Interpolate and add correction: $\widetilde{u}^h \leftarrow \widetilde{u}^h + I_H^h v^H$

(5) Relax again on h

(algebraic notation for linear system $A\,\mathbf{u} = \mathbf{b}$ with smoother M; $\mathbf{u}_k$ is the current approximation)

(1) $\widetilde{\mathbf{u}} = \mathbf{u}_k + M^{-1}(\mathbf{b} - A\,\mathbf{u}_k)$

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

(3) $A_c \mathbf{v}_c = \mathbf{r}_c \equiv R\,\widetilde{\mathbf{r}}$
(R: restriction, $n_c \times n$)
$R = P^T$, $A_c = R\,A\,P = P^T A\,P$

(4) $\widetilde{\mathbf{u}} \leftarrow \widetilde{\mathbf{u}} + P\,\mathbf{v}_c$
(P: prolongation, $n \times n_c$)

(5) $\mathbf{u}_{k+1} = \widetilde{\mathbf{u}} + M^{-1}(\mathbf{b} - A\,\widetilde{\mathbf{u}})$

$\rightarrow$ Try to obtain P from A

1. Intro: Why aggregation-based AMG?

ULB

Classical AMG

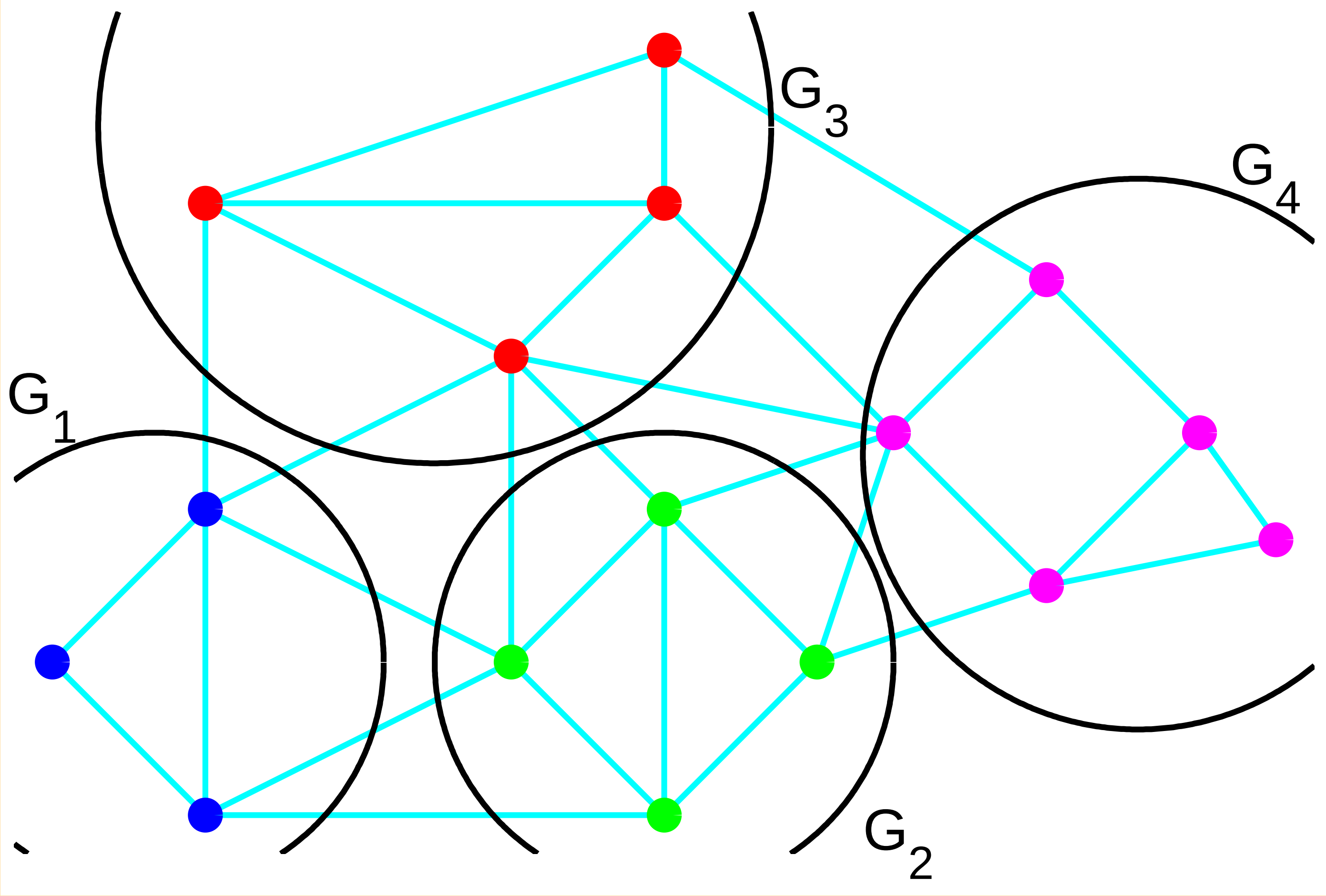
- Heuristic algorithms to mimic geometric multigrid
  (Connectivity $\rightarrow$ set of coarse nodes;
  Matrix entries $\rightarrow$ interpolation rules)
- Need to be used recursively:
  $A_c = P^T A\, P \rightarrow A_{cc} = P_c^T A_c\, P_c$, etc
  Is a good algorithm for A also good for A_c ?
- Several variants and parameters;
  relevant choices depend on applications
- Main difficulty:
  Find a good tradeoff between accuracy and the mastering of “complexity” (i.e., the control of the sparsity in successive coarse grid matrices)

1. Intro: Aggregation-based AMG

ULB

Group nodes into aggregates G_i (partitioning of $[1\,,\,n]$)

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



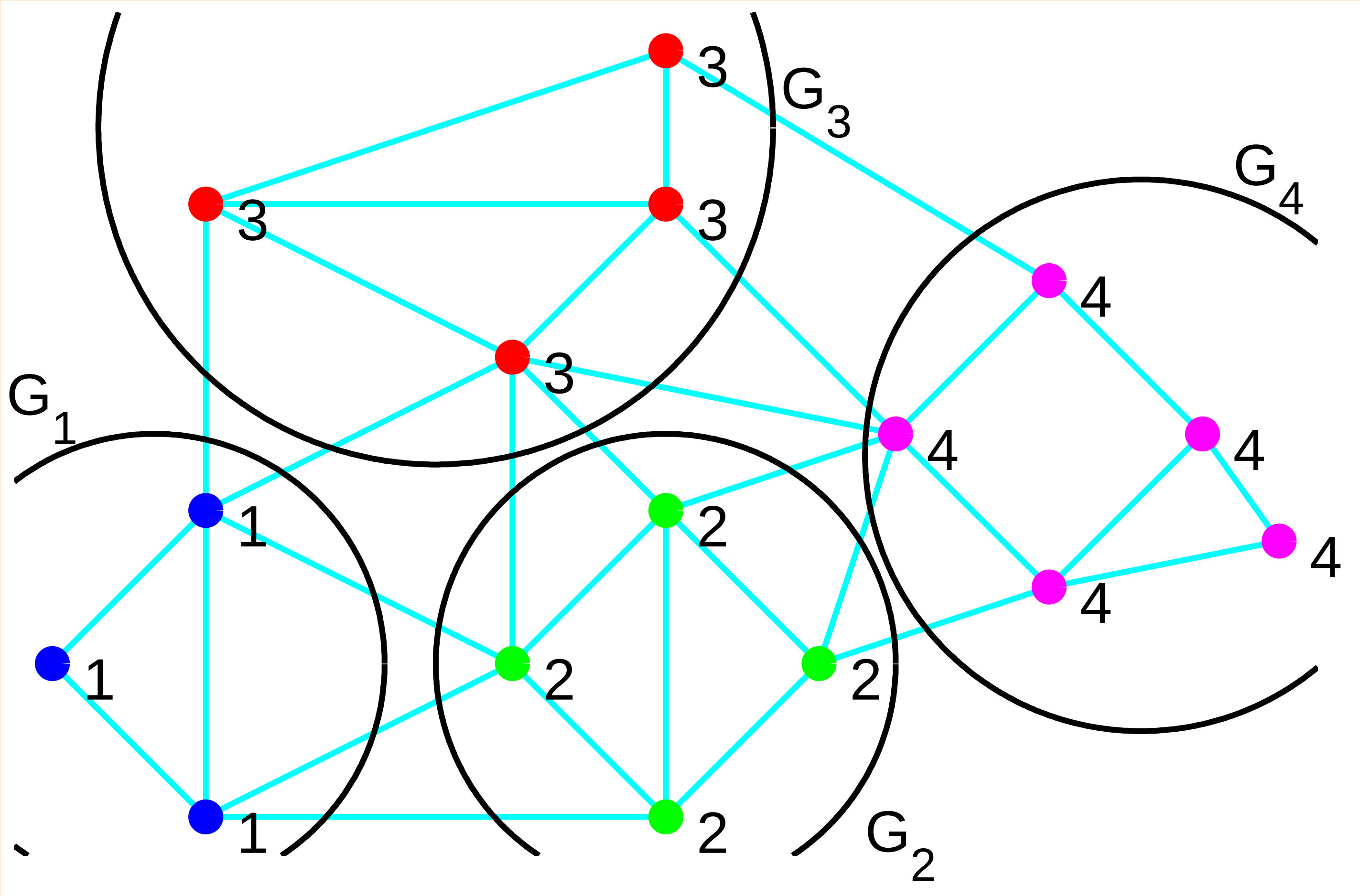
1. Intro: Aggregation-based AMG

ULB

Prolongation P : $P_{ij} = \begin{cases} 1 & \text{if } i \in G_j \\ 0 & \text{otherwise} \end{cases}$

Example

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

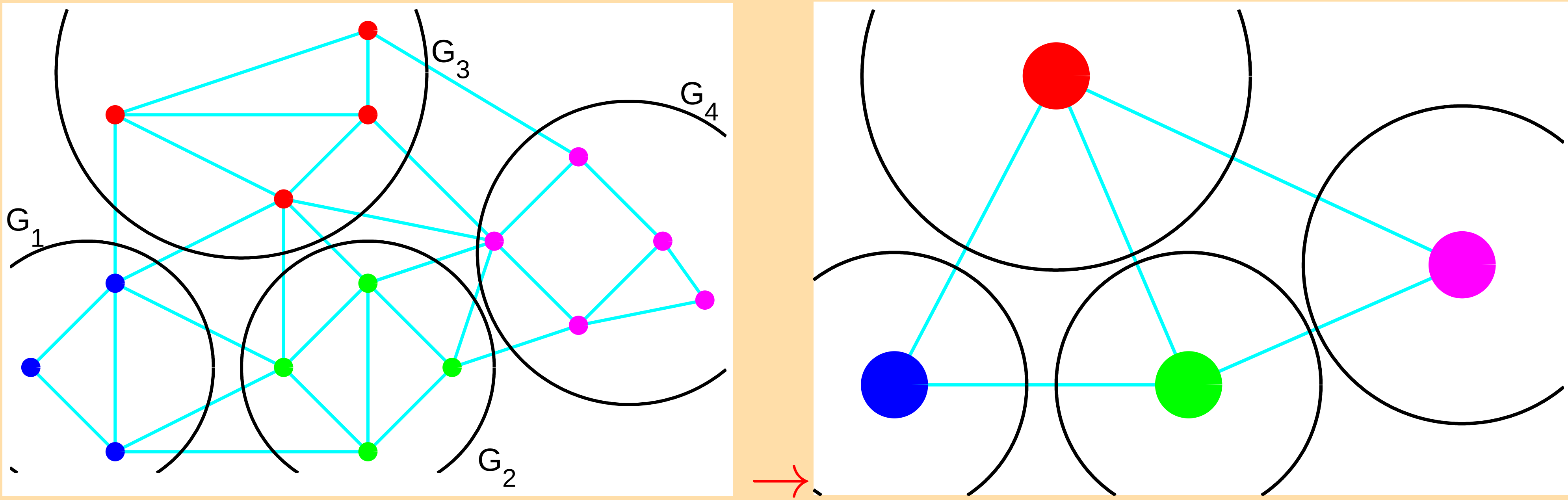


1. Intro: Aggregation-based AMG

ULB

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

$$(A_c)_{ij} = \sum_{k \in G_i} \sum_{\ell \in G_j} a_{k\ell}$$



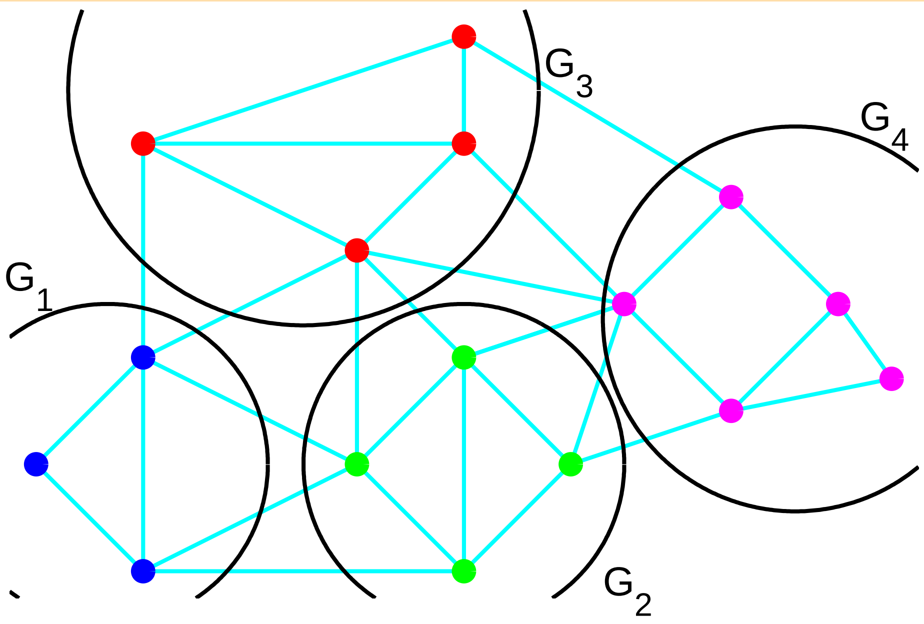
Tends to reproduce the stencil from the fine grid

1. Intro: Aggregation-based AMG

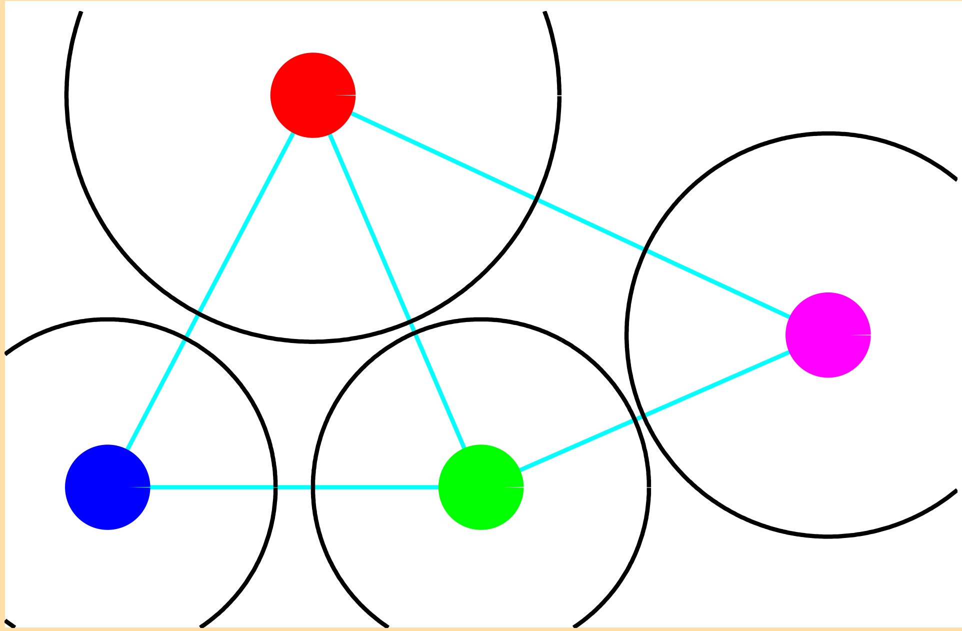
ULB

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

1. Intro: Aggregation-based AMG

ULB

- Does not mimic any classical multigrid method
- Not efficient if the piecewise constant P just substitutes the classical prolongation in a standard multigrid scheme

$\rightarrow$ has been overlooked for a long time

1. Intro: Aggregation-based AMG

ULB

- Does not mimic any classical multigrid method
- Not efficient if the piecewise constant P just substitutes the classical prolongation in a standard multigrid scheme

$\rightarrow$ has been overlooked for a long time

Recent revival:

- Proper convergence theory (mimicry not essential for a good interplay with the smoother)
- Efficient when combined with specific components: preconditioner for a Krylov method, cheap smoother & K-cycle (Krylov for coarse problems – all levels)
- Theory and efficient solver developed hand in hand

Outline

ULB

1. Introduction
2. AMG preconditioning and K-cycle
3. Two-grid analysis
4. Aggregation procedure
   - Repeated pairwise aggregation
5. Multi-level analysis
6. Parallelization
7. Numerical results
8. Conclusions

2. AMG preconditioning and K-cycle

ULB

Reminder:

Stationary iteration: $\mathbf{u}_{k+1} = \mathbf{u}_k + M^{-1}(\mathbf{b} - A\,\mathbf{u}_k)$

Corresponding preconditioning step:

$$\mathbf{v}_k = M^{-1}\mathbf{r}_k \quad (\mathbf{r}_k = \mathbf{b} - A\,\mathbf{u}_k)$$

2. AMG preconditioning and K-cycle

Reminder:

Stationary iteration: $\mathbf{u}_{k+1} = \mathbf{u}_k + M^{-1}(\mathbf{b} - A\,\mathbf{u}_k)$

Corresponding preconditioning step:

$$\mathbf{v}_k = M^{-1}\mathbf{r}_k \quad (\mathbf{r}_k = \mathbf{b} - A\,\mathbf{u}_k)$$

$\rightarrow$ for multigrid, rewrite the algorithm above as

$$\mathbf{u}_{k+1} \;=\; \mathbf{u}_k + B(\mathbf{b} - A\,\mathbf{u}_k)\ ;$$

B is the inverse of the preconditioner and

$$\mathbf{v}_k \;=\; B\,\mathbf{r}_k$$

the corresponding preconditioning step

2. AMG preconditioning and K-cycle

ULB

Benefit of Krylov

- Relaxed convergence conditions
- Scaling-independent convergence, characterized by the condition number (λ_i eig of $B\,A$) :

SPD: $$\kappa = \frac{\max_i \lambda_i}{\min_i \lambda_i} = \frac{\lambda_{\max}(B\,A)}{\lambda_{\min}(B\,A)}$$

General: $$\frac{\max_i |\lambda_i|}{\min_i \Re\mathrm{e}(\lambda_i)} \quad \text{or} \quad \frac{1}{\min_i \Re\mathrm{e}(1/\lambda_i)\ \min_i \Re\mathrm{e}(\lambda_i)}$$

(All eigs with positive real part)

- Accelerated convergence

2. AMG preconditioning and K-cycle

ULB

Benefit of Krylov

- Relaxed convergence conditions
- Scaling-independent convergence,
  characterized by the condition number (λ_i eig of $B\,A$) :

SPD: $$\kappa = \frac{\max_i \lambda_i}{\min_i \lambda_i} = \frac{\lambda_{\max}(B\,A)}{\lambda_{\min}(B\,A)}$$

General: $$\frac{\max_i |\lambda_i|}{\min_i \Re\mathrm{e}(\lambda_i)} \quad \text{or} \quad \frac{1}{\min_i \Re\mathrm{e}(1/\lambda_i)\ \min_i \Re\mathrm{e}(\lambda_i)}$$

(All eigs with positive real part)

- Accelerated convergence

Fast convergence:

if all λ_i bounded & substantially away from 0

2. AMG preconditioning and K-cycle

ULB

K-cycle

- Reminder: recursive use of the two-grid scheme:
  - $A_c \mathbf{v}_c = \mathbf{r}_c$ not solved exactly
  - $\mathbf{v}_c \leftarrow$ approximate solution from multigrid step(s) to solve the coarse system
  - 1 step $\rightarrow$ V-cycle
    2 steps $\rightarrow$ W-cycle
- K-cycle: solve $A_c \mathbf{v}_c = \mathbf{r}_c$ with 2 steps of a Krylov method with multigrid preconditioner at coarser level

  (essentially: W-cycle with Krylov acceleration)

2. AMG preconditioning and K-cycle

ULB

K-cycle -vs- V- & W-cycles

Number of iterations to reduce relative residual error by 10^{-12} as a function of the number of levels and of the convergence factor of the two grid method at each level

	7 levels	14 levels
$0.49 < \rho_{TG} < 0.50$ $(1.99 < \kappa_{TG} < 2.00)$		
V	188	> 999
W	37	50
K	20	20
$0.79 < \rho_{TG} < 0.80$ $(4.86 < \kappa_{TG} < 4.92)$		
V	256	> 999
W	108	315
K	42	44

2. AMG preconditioning and K-cycle

ULB

- Performances remain stable for a wide range of κ : the number of iterations is (near) independent of the number of levels

2. AMG preconditioning and K-cycle

- Performances remain stable for a wide range of κ : the number of iterations is (near) independent of the number of levels
- Hence analyzing the two-grid method is enough

2. AMG preconditioning and K-cycle

ULB

- Performances remain stable for a wide range of κ : the number of iterations is (near) independent of the number of levels
- Hence analyzing the two-grid method is enough
- $\neq$ from classical multigrid theory, based on a global view of all levels (or scales)

2. AMG preconditioning and K-cycle

ULB

- Performances remain stable for a wide range of κ : the number of iterations is (near) independent of the number of levels
- Hence analyzing the two-grid method is enough
- $\neq$ from classical multigrid theory, based on a global view of all levels (or scales)
- Classical multigrid: use “enough” smoothing steps to have spectral radius as small as desired

  Aggregation-based AMG:
  compensate for the larger condition number with Krylov, but also cheap smoothing stage (typically: one Gauss-Seidel sweep for pre- and post-smoothing)

Computational complexity

$$\text{Work} \sim C_W = \frac{\sum_{k=0}^{\ell} 2^k \, nnz(A_k)}{nnz(A)}$$

($A_0 = A$, $A_1 = A_c$, etc; $\ell =$ number of levels)

$\rightarrow$ ensure $\dfrac{nnz(A_k)}{nnz(A_{k-1})} \lesssim \dfrac{1}{4}$

(then $2^k nnz(A_k) \lesssim 2^{-k} nnz(A)$ and $C_W \lesssim 2$)

2. AMG preconditioning and K-cycle

ULB

Computational complexity

$$\text{Work} \sim C_W = \frac{\sum_{k=0}^{\ell} 2^k \, nnz(A_k)}{nnz(A)}$$

($A_0 = A$, $A_1 = A_c$, etc; $\ell =$ number of levels)

$\rightarrow$ ensure $\dfrac{nnz(A_k)}{nnz(A_{k-1})} \lesssim \dfrac{1}{4}$

(then $2^k nnz(A_k) \lesssim 2^{-k} nnz(A)$ and $C_W \lesssim 2$)

With aggregation-based methods:

$$\frac{nnz(A_k)}{nnz(A_{k-1})} \approx \frac{1}{\text{Mean aggregates' size}}$$

Outline

ULB

1. Introduction
2. AMG preconditioning and K-cycle
3. Two-grid analysis
4. Aggregation procedure
   Repeated pairwise aggregation
5. Multi-level analysis
6. Parallelization
7. Numerical results
8. Conclusions

3. Two-grid analysis

The algebraic convergence theory:

- yields meaningful bounds
  (actual convergence as proved often OK)

3. Two-grid analysis

ULB

The algebraic convergence theory:

- yields meaningful bounds
  (actual convergence as proved often OK)
- is compatible with irregular geometries, unstructured grids, jumps in coefficients, etc
  (guarantees essentially the same bound)

3. Two-grid analysis

The algebraic convergence theory:

- yields meaningful bounds
  (actual convergence as proved often OK)
- is compatible with irregular geometries, unstructured grids, jumps in coefficients, etc
  (guarantees essentially the same bound)
- requires M-matrix, but has natural heuristic extensions

3. Two-grid analysis

ULB

The algebraic convergence theory:

- yields meaningful bounds
  (actual convergence as proved often OK)
- is compatible with irregular geometries, unstructured grids, jumps in coefficients, etc
  (guarantees essentially the same bound)
- requires M-matrix, but has natural heuristic extensions
- whenever applicable, holds at every level of the hierarchy

3. Two-grid analysis

The algebraic convergence theory:

- yields meaningful bounds
  (actual convergence as proved often OK)
- is compatible with irregular geometries, unstructured grids, jumps in coefficients, etc
  (guarantees essentially the same bound)
- requires M-matrix, but has natural heuristic extensions
- whenever applicable, holds at every level of the hierarchy
- covers symmetric and nonsymmetric problems in a uniform fashion

3. Two-grid analysis

The algebraic convergence theory:

- yields meaningful bounds
  (actual convergence as proved often OK)
- is compatible with irregular geometries, unstructured grids, jumps in coefficients, etc
  (guarantees essentially the same bound)
- requires M-matrix, but has natural heuristic extensions
- whenever applicable, holds at every level of the hierarchy
- covers symmetric and nonsymmetric problems in a uniform fashion

The aggregation algorithm we use is entirely based on the theory and its heuristic extensions

3. Two-grid analysis

ULB

- Method used as a preconditioner for CG or GCR

  $\rightarrow$ Fast convergence if the eigenvalues λ_i of the preconditioned matrix are:

  - bounded
  - substantially away from 0

- Using a standard smoother (e.g., Gauss-Seidel), the eigenvalues are bounded independently of P
- If $P = 0$ the eigenvalues associated with "smooth" modes are in general very small
  - $\rightarrow$ Main difficulty: λ_i substantially away from 0
  - Role of the coarse grid correction: move the small eigenvalues enough to the right (Guideline for the choice of P)

3. Two-grid analysis: λ_i away from 0

ULB

SPD case

Main identity [Falgout, Vassilevski & Zikatanov (2005)]:

$$\lambda_{\min} = \frac{1}{\kappa(A\,,\,P)}$$

with

$$\kappa(A,P) = \omega^{-1} \sup_{\mathbf{v}\neq 0} \frac{\mathbf{v}^T D\left(I - P(P^T D\, P)^{-1} P^T D\right)\mathbf{v}}{\mathbf{v}^T A \mathbf{v}}$$

3. Two-grid analysis: λ_i away from 0

ULB

SPD case

Main identity [Falgout, Vassilevski & Zikatanov (2005)]:

$$\lambda_{\min} = \frac{1}{\kappa(A\,,\,P)}$$

with

$$\kappa(A,P) = \omega^{-1} \sup_{\mathbf{v}\neq 0} \frac{\mathbf{v}^T D \left(I - P(P^T D\, P)^{-1} P^T D\right) \mathbf{v}}{\mathbf{v}^T A \mathbf{v}}$$

General case [YN (2010)]

For any λ_i :

$$\Re\mathrm{e}(\lambda_i) \geq \frac{1}{\kappa(A_S\,,\,P)}$$

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

The analysis of the SPD case can be sufficient

3. Two-grid analysis: λ_i away from 0

$$\kappa(A_S, P) = \omega^{-1} \sup_{\mathbf{v}\neq 0} \frac{\mathbf{v}^T D\left(I - P(P^T D\, P)^{-1} P^T D\right)\mathbf{v}}{\mathbf{v}^T A_S \mathbf{v}}$$

Aggregation-based methods

$$P = \begin{pmatrix} \mathbf{1}_{n^{(1)}} & & \\ & \ddots & \\ & & \mathbf{1}_{n^{(n_c)}} \end{pmatrix} \quad , \quad D = \text{diag}(A) = \begin{pmatrix} D_1 & & \\ & \ddots & \\ & & D_{n_c} \end{pmatrix}$$

$$\rightarrow D\left(I - P(P^T D\, P)^{-1} P^T D\right)$$
$$= \text{blockdiag}\left(D_i\left(I - \mathbf{1}_{n^{(i)}}(\mathbf{1}_{n^{(i)}}^T D_i\, \mathbf{1}_{n^{(i)}})^{-1} \mathbf{1}_{n^{(i)}}^T D_i\right)\right)$$

3. Two-grid analysis: λ_i away from 0

ULB

$$\kappa(A_S, P) = \omega^{-1} \sup_{\mathbf{v}\neq 0} \frac{\mathbf{v}^T D \left(I - P(P^T D\, P)^{-1} P^T D\right) \mathbf{v}}{\mathbf{v}^T A_S \mathbf{v}}$$

Aggregation-based methods

$$P = \begin{pmatrix} \mathbf{1}_{n^{(1)}} & & \\ & \ddots & \\ & & \mathbf{1}_{n^{(n_c)}} \end{pmatrix}, \quad D = \text{diag}(A) = \begin{pmatrix} D_1 & & \\ & \ddots & \\ & & D_{n_c} \end{pmatrix}$$

$$\rightarrow D\left(I - P(P^T D\, P)^{-1} P^T D\right) = \text{blockdiag}\left(D_i\left(I - \mathbf{1}_{n^{(i)}}(\mathbf{1}_{n^{(i)}}^T D_i\, \mathbf{1}_{n^{(i)}})^{-1} \mathbf{1}_{n^{(i)}}^T D_i\right)\right)$$

$\rightarrow$ find A_b, A_r nonnegative definite s.t. $A_S = A_b + A_r$ with

$$A_b = \begin{pmatrix} A_{G_1}^{(S)} & & \\ & \ddots & \\ & & A_{G_{n_c}^{(S)}} \end{pmatrix}$$

3. Two-grid analysis: λ_i away from 0

$$\kappa(A_S, P) \leq \omega^{-1} \sup_{\mathbf{v}\neq 0} \frac{\mathbf{v}^T D \left(I - P(P^T D\, P)^{-1} P^T D\right) \mathbf{v}}{\mathbf{v}^T A_b \mathbf{v}}$$

Aggregation-based methods

$$P = \begin{pmatrix} \mathbf{1}_{n^{(1)}} & & \\ & \ddots & \\ & & \mathbf{1}_{n^{(n_c)}} \end{pmatrix}, \quad D = \text{diag}(A) = \begin{pmatrix} D_1 & & \\ & \ddots & \\ & & D_{n_c} \end{pmatrix}$$

$$\rightarrow D \left(I - P(P^T D\, P)^{-1} P^T D\right)$$
$$= \text{blockdiag}\left(D_i \left(I - \mathbf{1}_{n^{(i)}} (\mathbf{1}_{n^{(i)}}^T D_i\, \mathbf{1}_{n^{(i)}})^{-1} \mathbf{1}_{n^{(i)}}^T D_i\right)\right)$$

$\rightarrow$ find A_b, A_r nonnegative definite s.t. $A_S = A_b + A_r$ with

$$A_b = \begin{pmatrix} A_{G_1}^{(S)} & & \\ & \ddots & \\ & & A_{G_{n_c}^{(S)}} \end{pmatrix}$$

3. Two-grid analysis: λ_i away from 0

ULB

Aggregate Quality

$$\mu_G = \omega^{-1} \sup_{\mathbf{v}\notin\mathcal{N}(A_G^{(S)})} \frac{\mathbf{v}^T D_G(I - \mathbf{1}_G(\mathbf{1}_G^T D_G \mathbf{1}_G)^{-1}\mathbf{1}_G^T D_G)\mathbf{v}}{\mathbf{v}^T A_G^{(S)} \mathbf{v}} ,$$

Then: $\kappa(A_S\,,\,P) \;\leq\; \max_i \mu_{G_i}$

Controlling μ_{G_i} ensures that eigenvalues are away from 0

3. Two-grid analysis: λ_i away from 0

ULB

Aggregate Quality

$$\mu_G = \omega^{-1} \sup_{\mathbf{v}\notin\mathcal{N}(A_G^{(S)})} \frac{\mathbf{v}^T D_G (I - \mathbf{1}_G(\mathbf{1}_G^T D_G \mathbf{1}_G)^{-1}\mathbf{1}_G^T D_G)\mathbf{v}}{\mathbf{v}^T A_G^{(S)} \mathbf{v}},$$

Then: $\kappa(A_S\,,\,P) \leq \max_i \mu_{G_i}$

Controlling μ_{G_i} ensures that eigenvalues are away from 0

$A_G^{(S)}$: Computed from $A_S = A_b + A_r$ with $A_r\mathbf{1} = 0$

- Rigorous for M-matrices s.t. $A_S\mathbf{1} \geq 0$
(then A_b , A_r guaranteed nonnegative definite)
- Heuristic in other cases
(A_r could have negative eigenvalue(s))

Outline

ULB

1. Introduction
2. AMG preconditioning and K-cycle
3. Two-grid analysis
4. Aggregation procedure
   Repeated pairwise aggregation
5. Multi-level analysis
6. Parallelization
7. Numerical results
8. Conclusions

4. Aggregation procedure

$\kappa(A_S\,,\,P)\;\leq\;\max_i \mu_{G_i}$

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

4. Aggregation procedure

$\kappa(A_S\,,\,P)\;\leq\;\max_i \mu_{G_i}$

- 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_i}

4. Aggregation procedure

$\kappa(A_S\,,\,P) \;\leq\; \max_i \mu_{G_i}$

- 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_i}
- Problem: repeated assessment of μ_G is costly

4. Aggregation procedure

ULB

$\kappa(A_S\,,\,P)\;\leq\;\max_i \mu_{G_i}$

- 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_i}
- Problem: repeated assessment of μ_G is costly
- For a pair $\{i,j\}$, $\mu_{\{i,j\}}$ is a simple function of the "local" entries & the row and column sum

4. Aggregation procedure

ULB

$\kappa(A_S\,,\,P)\;\leq\;\max_i \mu_{G_i}$

- 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_i}
- Problem: repeated assessment of μ_G is costly
- For a pair $\{i,j\}$, $\mu_{\{i,j\}}$ is a simple function of the "local" entries & the row and column sum
- $\mu_G \;=\; \omega^{-1}\sup_{\mathbf{z}\notin\mathcal{N}(A_G^{(S)})} \dfrac{\mathbf{z}^T D_G(I-\mathbf{1}_G\left(\mathbf{1}_G^T D_G \mathbf{1}_G\right)^{-1}\mathbf{1}_G^T D_G)\,\mathbf{z}}{\mathbf{z}^T A_G^{(S)}\,\mathbf{z}}$

  It is always cheap to check that $\mu_G < \overline{\kappa}_{\text{TG}}$ holds:

  $$Z_G \;=\; \overline{\kappa}_{\text{TG}}\, A_G^{(S)} - \omega^{-1} D_G(I - \mathbf{1}_G\,(\mathbf{1}_G^T D_G \mathbf{1}_G)^{-1}\,\mathbf{1}_G^T D_G)$$

  is nonnegative definite if no negative pivot occurs while performing an LDL^T factorization

Pairwise aggregation

Input: threshold $\bar{\kappa}_{\text{TG}}$

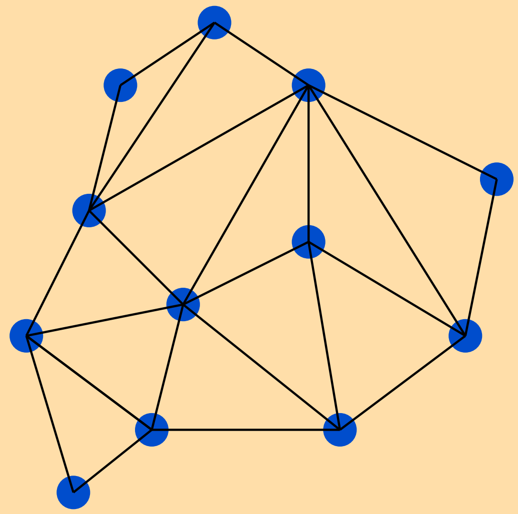
Output:

n_c and aggregates G_i , $i = 1 \ldots , n_c$

Initialization: $U = [1\,,\,n] \backslash G_0$, $n_c = 0$

Algorithm: While $U \neq \emptyset$ do

1. Select $i \in U$; $n_c = n_c + 1$
2. Select $j \in U$ such that $\mu_{\{i,j\}}$ is minimal
3. **If** $\mu_{\{i,j\}} < \bar{\kappa}_{\text{TG}}$ **then** $G_{n_c} = \{i, j\}$ **else** $G_{n_c} = \{i\}$
4. $U = U \backslash G_{n_c}$



4. Pairwise aggregation

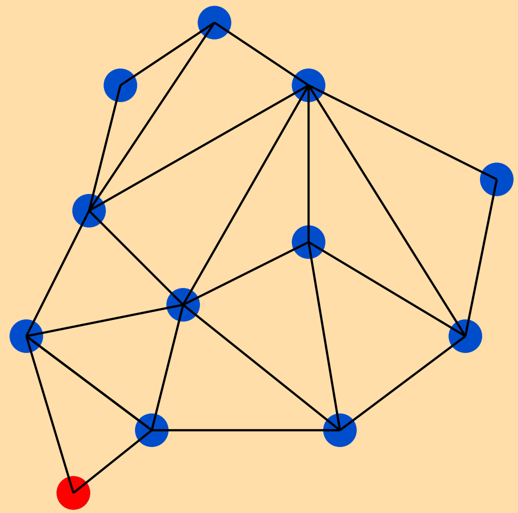
Input: threshold $\bar{\kappa}_{\text{TG}}$

Output:
n_c and aggregates G_i , $i = 1 \ldots, n_c$

Initialization: $U = [1\,,\, n]\backslash G_0$, $n_c = 0$

Algorithm: While $U \neq \emptyset$ do

1. Select $i \in U$; $n_c = n_c + 1$
2. Select $j \in U$ such that $\mu_{\{i,j\}}$ is minimal
3. **If** $\mu_{\{i,j\}} < \bar{\kappa}_{\text{TG}}$ **then** $G_{n_c} = \{i, j\}$ **else** $G_{n_c} = \{i\}$
4. $U = U \backslash G_{n_c}$



4. Pairwise aggregation

ULB

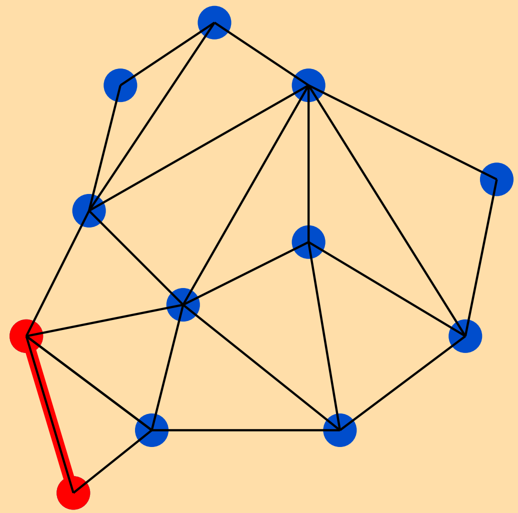
Input: threshold $\bar{\kappa}_{\text{TG}}$

Output:
n_c and aggregates G_i , $i = 1 \ldots , n_c$

Initialization: $U = [1\,,\,n] \backslash G_0$, $n_c = 0$

Algorithm: While $U \neq \emptyset$ do

1. Select $i \in U$; $n_c = n_c + 1$
2. Select $j \in U$ such that $\mu_{\{i,j\}}$ is minimal
3. **If** $\mu_{\{i,j\}} < \bar{\kappa}_{\text{TG}}$ **then** $G_{n_c} = \{i, j\}$ **else** $G_{n_c} = \{i\}$
4. $U = U \backslash G_{n_c}$



4. Pairwise aggregation

Input: threshold $\bar{\kappa}_{\text{TG}}$

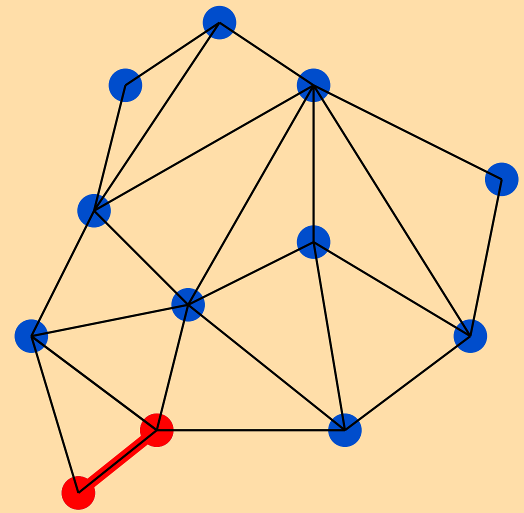
Output:

n_c and aggregates G_i , $i = 1\ldots, n_c$

Initialization: $U = [1\,,\,n]\backslash G_0$, $n_c = 0$

Algorithm: While $U \neq \emptyset$ do

1. Select $i \in U$; $n_c = n_c + 1$
2. Select $j \in U$ such that $\mu_{\{i,j\}}$ is minimal
3. **If** $\mu_{\{i,j\}} < \bar{\kappa}_{\text{TG}}$ **then** $G_{n_c} = \{i, j\}$
   else $G_{n_c} = \{i\}$
4. $U = U\backslash G_{n_c}$



4. Pairwise aggregation

Input: threshold $\bar{\kappa}_{\text{TG}}$

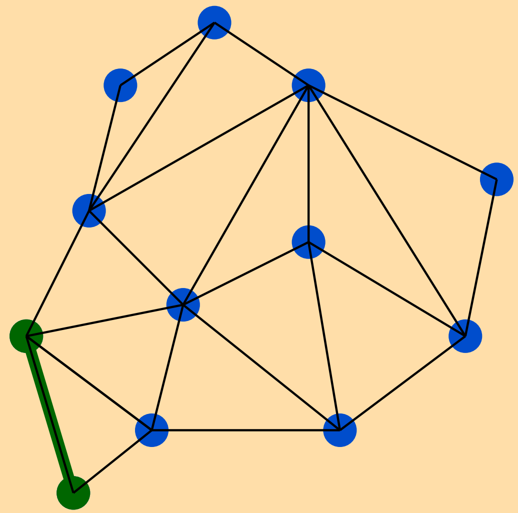
Output:

n_c and aggregates G_i , $i = 1 \ldots , n_c$

Initialization: $U = [1 , n] \backslash G_0$, $n_c = 0$

Algorithm: While $U \neq \emptyset$ do

1. Select $i \in U$; $n_c = n_c + 1$
2. Select $j \in U$ such that $\mu_{\{i,j\}}$ is minimal
3. **If** $\mu_{\{i,j\}} < \bar{\kappa}_{\text{TG}}$ **then** $G_{n_c} = \{i, j\}$ **else** $G_{n_c} = \{i\}$
4. $U = U \backslash G_{n_c}$



4. Pairwise aggregation

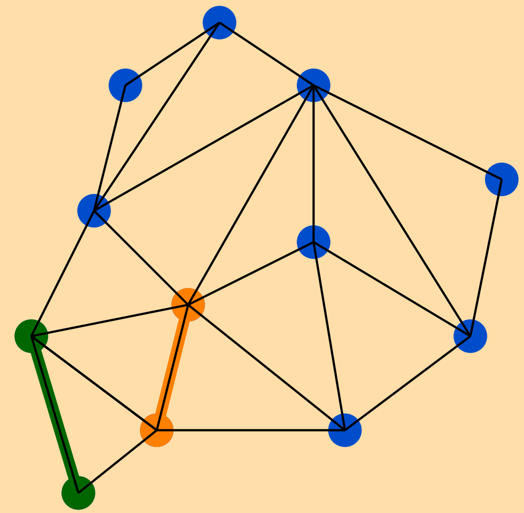
Input: threshold $\bar{\kappa}_{\text{TG}}$

Output:
n_c and aggregates G_i , $i = 1 \ldots, n_c$

Initialization: $U = [1\,,\,n] \backslash G_0$, $n_c = 0$

Algorithm: While $U \neq \emptyset$ do

1. Select $i \in U$; $n_c = n_c + 1$
2. Select $j \in U$ such that $\mu_{\{i,j\}}$ is minimal
3. **If** $\mu_{\{i,j\}} < \bar{\kappa}_{\text{TG}}$ **then** $G_{n_c} = \{i, j\}$
   else $G_{n_c} = \{i\}$
4. $U = U \backslash G_{n_c}$



4. Pairwise aggregation

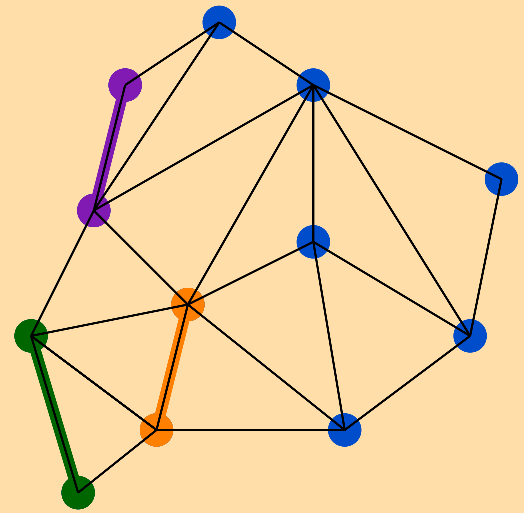
Input: threshold $\bar{\kappa}_{\mathsf{TG}}$

Output:
n_c and aggregates G_i , $i = 1 \ldots , n_c$

Initialization: $U = [1\,,\,n] \backslash G_0$, $n_c = 0$

Algorithm: While $U \neq \emptyset$ do

1. Select $i \in U$; $n_c = n_c + 1$
2. Select $j \in U$ such that $\mu_{\{i,j\}}$ is minimal
3. **If** $\mu_{\{i,j\}} < \bar{\kappa}_{\mathsf{TG}}$ **then** $G_{n_c} = \{i, j\}$
   else $G_{n_c} = \{i\}$
4. $U = U \backslash G_{n_c}$



4. Repeated Pairwise aggregation

ULB

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

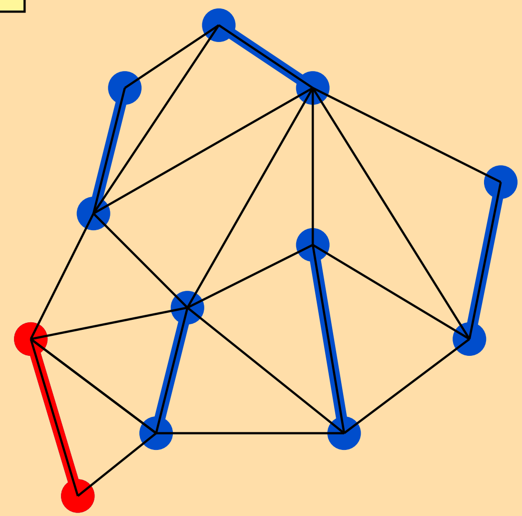
Apply pairwise aggregation to $A^{(s)}$

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

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

no: $s \leftarrow s + 1$

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



4. Repeated Pairwise aggregation

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

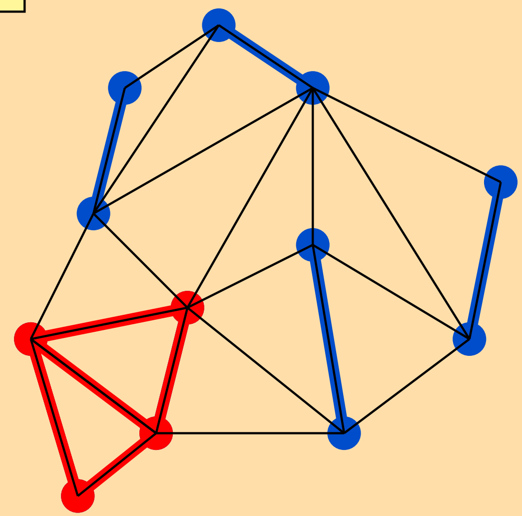
Apply pairwise aggregation to $A^{(s)}$

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

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

no: $s \leftarrow s + 1$

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



4. Repeated Pairwise aggregation

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

↓

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

↓

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

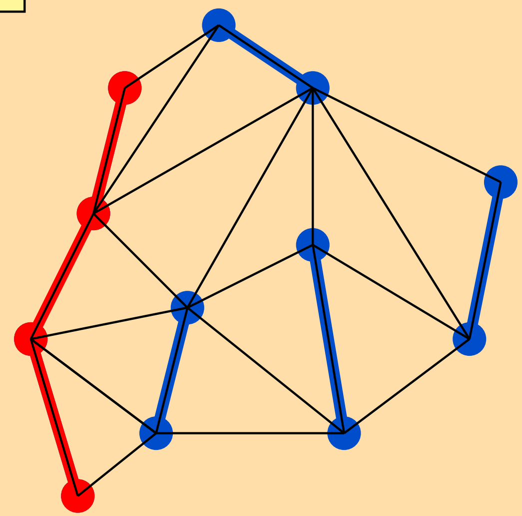
↓

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

no → $s \leftarrow s + 1$ (return to "Apply pairwise aggregation to $A^{(s)}$")

yes ↓

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



4. Repeated Pairwise aggregation

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

Check $\mu_G < \overline{\kappa}_{\text{TG}}$ in A

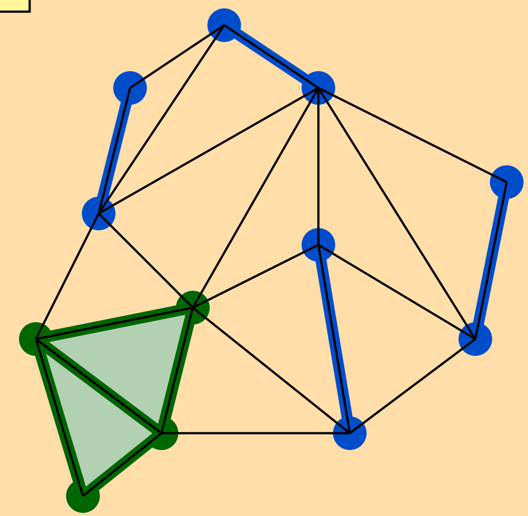
Apply pairwise aggregation to $A^{(s)}$

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

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

no: $s \leftarrow s + 1$

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



4. Repeated Pairwise aggregation

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

Check $\mu_G < \overline{\kappa}_{\text{TG}}$ in A

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

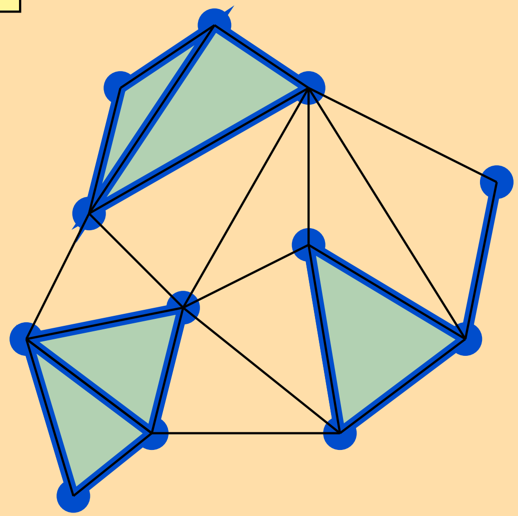
Form aggregated matrix $A^{(s+1)}$

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

no: $s \leftarrow s + 1$

yes

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

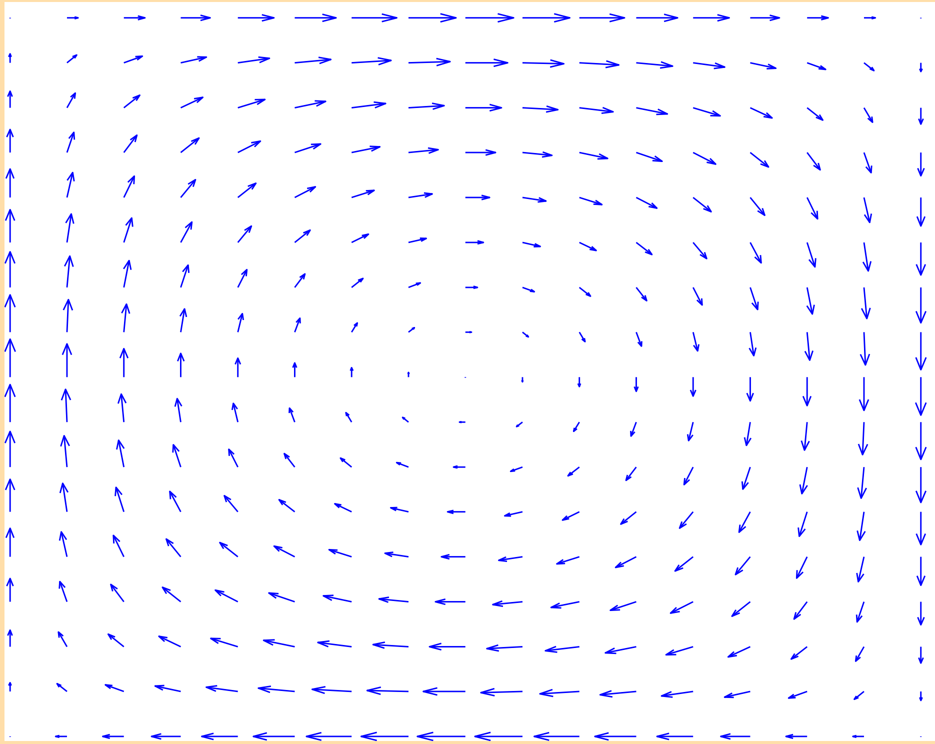


4. Aggregation procedure: Illustration

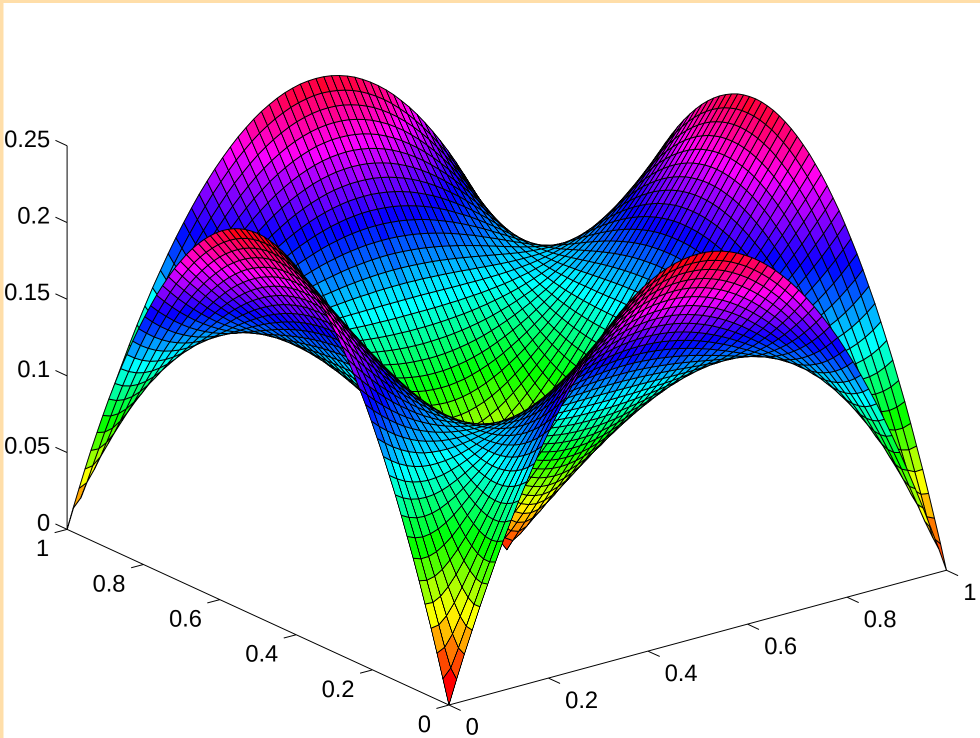
Upwind FD approximation of

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

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

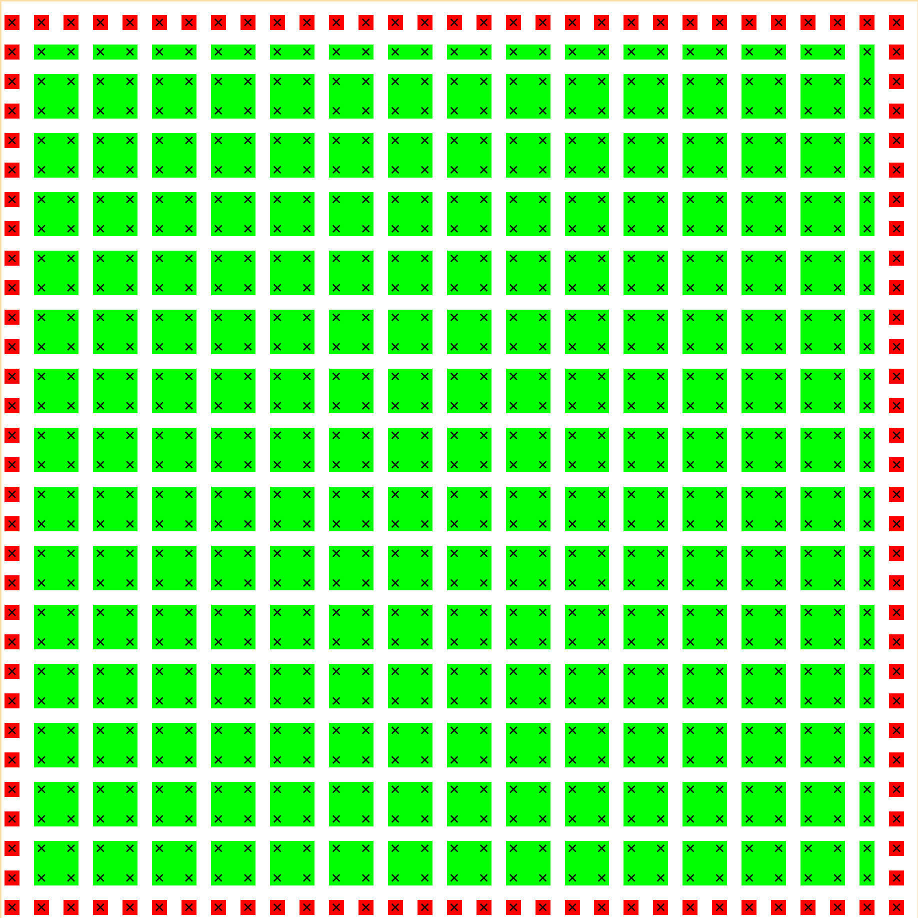


Magnitude

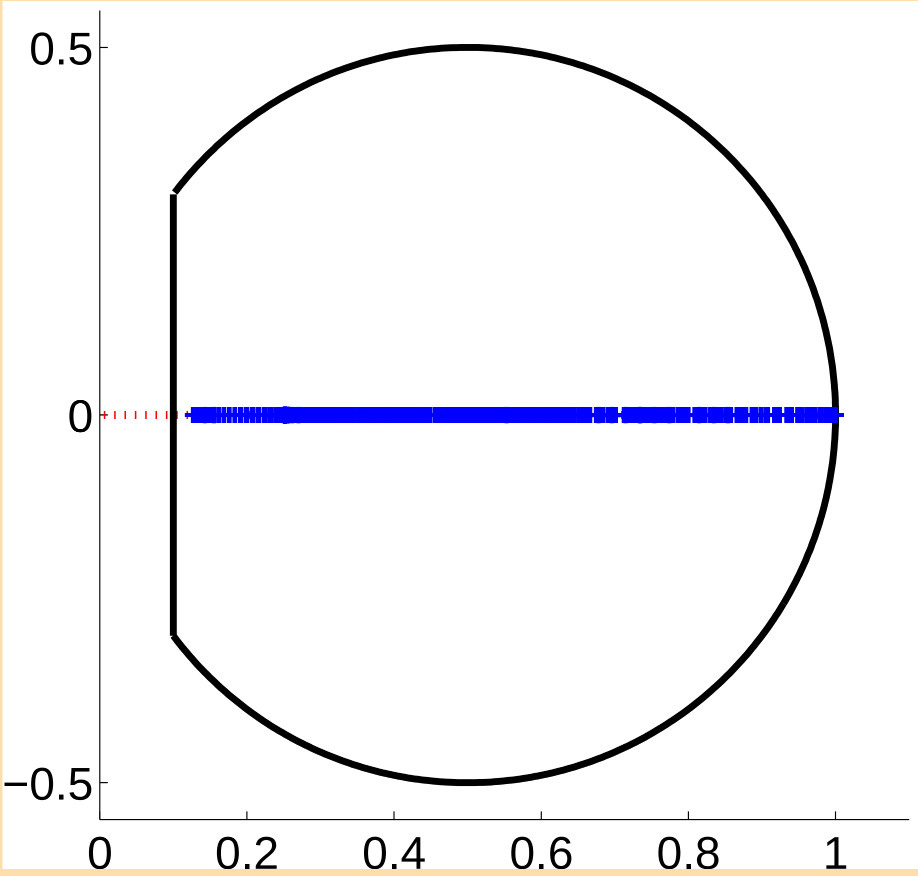
4. Aggregation procedure: Illustration

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

Aggregation



Spectrum



+ : $\sigma(I - T)$ — : theory

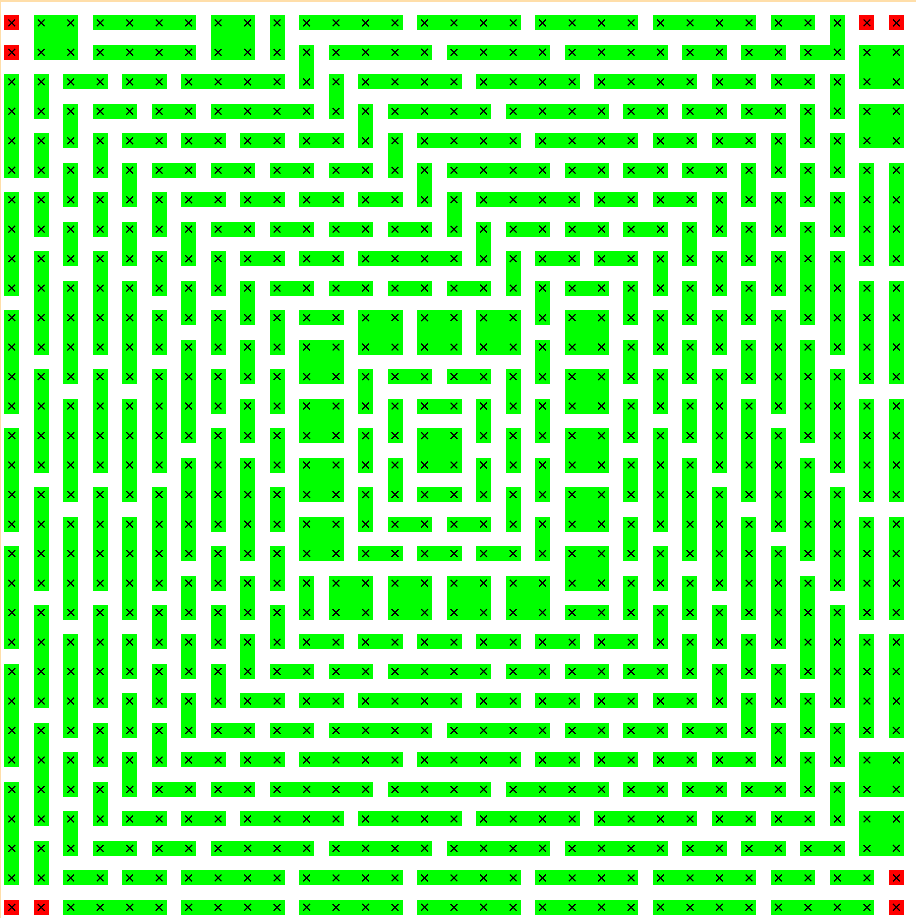
···· : $\sigma(\omega\, D^{-1} A)$ (convex hull)

4. Aggregation procedure: Illustration

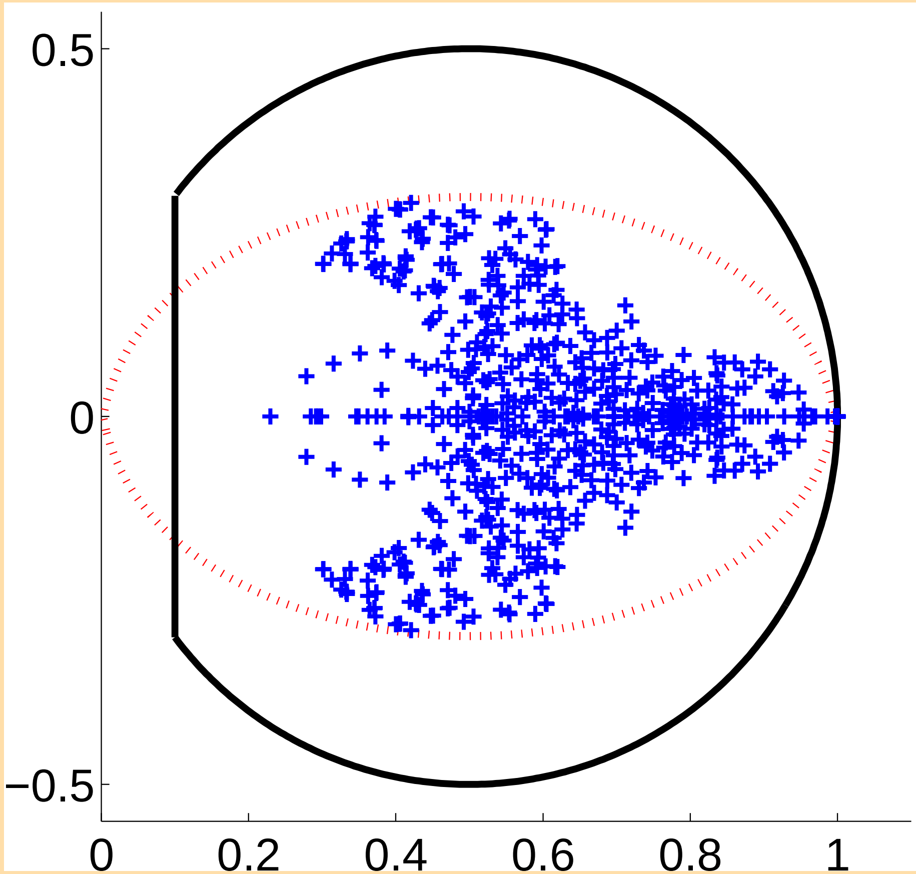
ULB

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

Aggregation



Spectrum



+ : $\sigma(I - T)$ — : theory

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

Outline

ULB

1. Introduction
2. AMG preconditioning and K-cycle
3. Two-grid analysis
4. Aggregation procedure
   Repeated pairwise aggregation
5. Multi-level analysis
6. Parallelization
7. Numerical results
8. Conclusions

5. Multi-level analysis

ULB

Requires to exchange the K-cycle (Krylov acceleration) for the AMLI-cycle (polynomial acceleration; i.e., frozen coefficients)

- less flexible: requires a known bound $\overline{\rho}$ on the two-grid convergence factor
- less efficient in practice
- avoid nonlinearities $\rightarrow$ convergence proof easier
- upper bound on the convergence rate independent of the number of levels can be guaranteed with the sole assumption that $\overline{\rho}$ is below a given threshold

5. Multi-level analysis

Requires to exchange the K-cycle (Krylov acceleration) for the AMLI-cycle (polynomial acceleration; i.e., frozen coefficients)

- less flexible: requires a known bound $\overline{\rho}$ on the two-grid convergence factor
- less efficient in practice
- avoid nonlinearities $\rightarrow$ convergence proof easier
- upper bound on the convergence rate independent of the number of levels can be guaranteed with the sole assumption that $\overline{\rho}$ is below a given threshold

Our aggregation procedure: allows to choose $\overline{\rho}$
(for symmetric M-matrices with nonnegative row sum)

5. Multi-level analysis:final result

- The method is purely algebraic and applies to any symmetric M-matrix with nonnegative row-sum

5. Multi-level analysis:final result

- The method is purely algebraic and applies to any symmetric M-matrix with nonnegative row-sum
- The condition number is bounded

5. Multi-level analysis:final result

- The method is purely algebraic and applies to any symmetric M-matrix with nonnegative row-sum
- The condition number is bounded
  - independently of mesh or problem size

5. Multi-level analysis:final result

- The method is purely algebraic and applies to any symmetric M-matrix with nonnegative row-sum
- The condition number is bounded
  - independently of mesh or problem size
  - independently of the number of levels

5. Multi-level analysis:final result

ULB

- The method is purely algebraic and applies to any symmetric M-matrix with nonnegative row-sum
- The condition number is bounded
  - independently of mesh or problem size
  - independently of the number of levels
  - independently of matrix coefficients (jumps, anisotropy, etc)

5. Multi-level analysis:final result

- The method is purely algebraic and applies to any symmetric M-matrix with nonnegative row-sum
- The condition number is bounded
  - independently of mesh or problem size
  - independently of the number of levels
  - independently of matrix coefficients (jumps, anisotropy, etc)
  - independently of the regularity of the grid

5. Multi-level analysis:final result

ULB

- The method is purely algebraic and applies to any symmetric M-matrix with nonnegative row-sum
- The condition number is bounded
  - independently of mesh or problem size
  - independently of the number of levels
  - independently of matrix coefficients (jumps, anisotropy, etc)
  - independently of the regularity of the grid
  - independently of the type of refinement (quasi uniformity, etc)

5. Multi-level analysis:final result

ULB

- The method is purely algebraic and applies to any symmetric M-matrix with nonnegative row-sum
- The condition number is bounded
  - independently of mesh or problem size
  - independently of the number of levels
  - independently of matrix coefficients (jumps, anisotropy, etc)
  - independently of the regularity of the grid
  - independently of the type of refinement (quasi uniformity, etc)
  - independently of any regularity assumption

5. Multi-level analysis:final result

Why ?

5. Multi-level analysis: final result

Why ?

. . . because the upper bound is 27.056

5. Multi-level analysis:final result

Why ?

...because the upper bound is 27.056

Optimality requires in addition bounded complexity:

- can be proved for model problems on regular grids;
- no proof in general, but, in practice, no more complexity issues than with other AMG schemes: coarsening parameters selected for this.

Outline

ULB

1. Introduction
2. AMG preconditioning and K-cycle
3. Two-grid analysis
4. Aggregation procedure
   Repeated pairwise aggregation
5. Multi-level analysis
6. Parallelization
7. Numerical results
8. Conclusions

6. Parallelization

- Partitioning of the unknowns

  $\rightarrow$ partitioning of matrix rows

6. Parallelization

- Partitioning of the unknowns
  $\rightarrow$ partitioning of matrix rows
- We apply exactly the same aggregation algorithm except that aggregates can only contain unknowns in a same partition.
  Hence, one needs only to know the local matrix rows (no communication except upon forming the next coarse grid matrix)

6. Parallelization

ULB

- Partitioning of the unknowns
  $\rightarrow$ partitioning of matrix rows
- We apply exactly the same aggregation algorithm except that aggregates can only contain unknowns in a same partition.
  Hence, one needs only to know the local matrix rows (no communication except upon forming the next coarse grid matrix)
- The prolongations & restrictions are then purely local

6. Parallelization

ULB

- Partitioning of the unknowns
  $\rightarrow$ partitioning of matrix rows
- We apply exactly the same aggregation algorithm except that aggregates can only contain unknowns in a same partition.
  Hence, one needs only to know the local matrix rows (no communication except upon forming the next coarse grid matrix)
- The prolongations & restrictions are then purely local
- Smoother: Gauss-Seidel, ignoring connections between different partitions $\rightarrow$ inherently parallel

6. Parallelization

- Partitioning of the unknowns
  $\rightarrow$ partitioning of matrix rows
- We apply exactly the same aggregation algorithm except that aggregates can only contain unknowns in a same partition.
  Hence, one needs only to know the local matrix rows (no communication except upon forming the next coarse grid matrix)
- The prolongations & restrictions are then purely local
- Smoother: Gauss-Seidel, ignoring connections between different partitions $\rightarrow$ inherently parallel
- During iterations: communications only for matvec and inner product computation

Outline

ULB

1. Introduction
2. AMG preconditioning and K-cycle
3. Two-grid analysis
4. Aggregation procedure
    Repeated pairwise aggregation
5. Multi-level analysis
6. Parallelization
7. Numerical results
8. Conclusions

7. Numerical results

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.

7. Numerical results

ULB

This talk

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

7. Numerical results

ULB

This talk

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

7. Numerical results

ULB

This talk

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

7. Numerical results

ULB

This talk

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

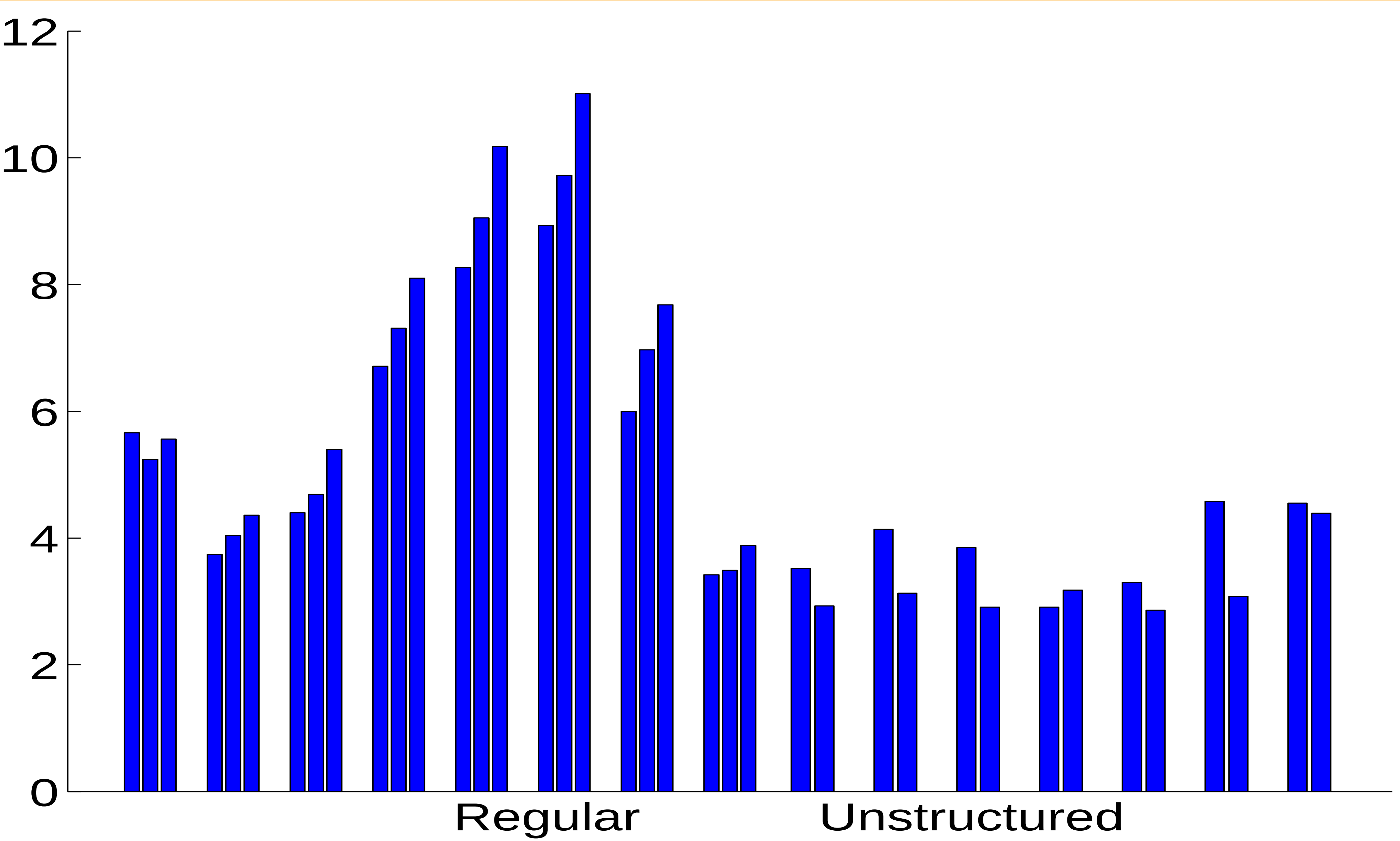
7. Numerical results

ULB

- Iterations stopped when $\frac{\|\mathbf{r}_k\|}{\|\mathbf{r}_0\|} < 10^{-6}$
- 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: $n = 0.15e6 \;\rightarrow\; n = 7.1e6$

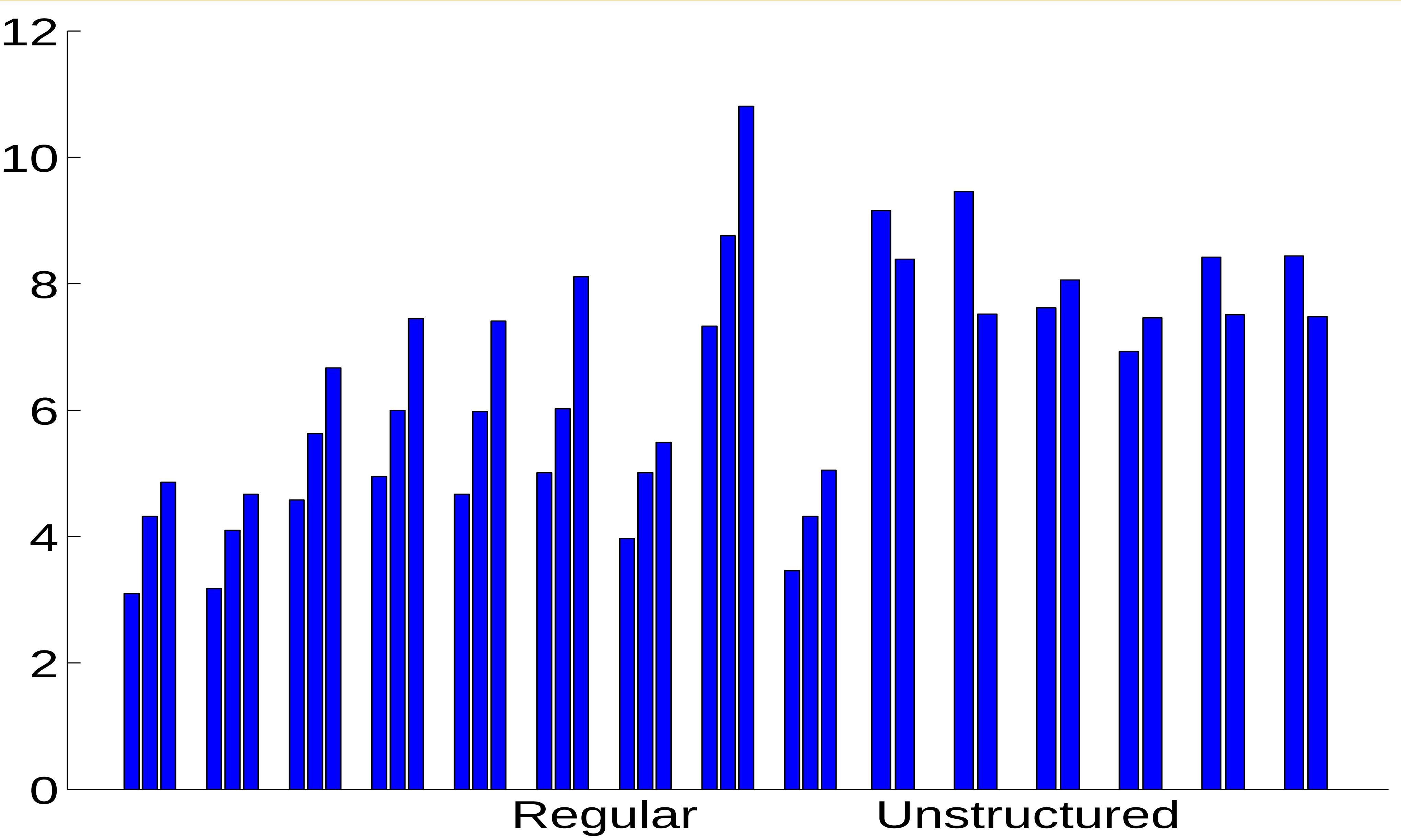
7. Numerical results

2D symmetric problems



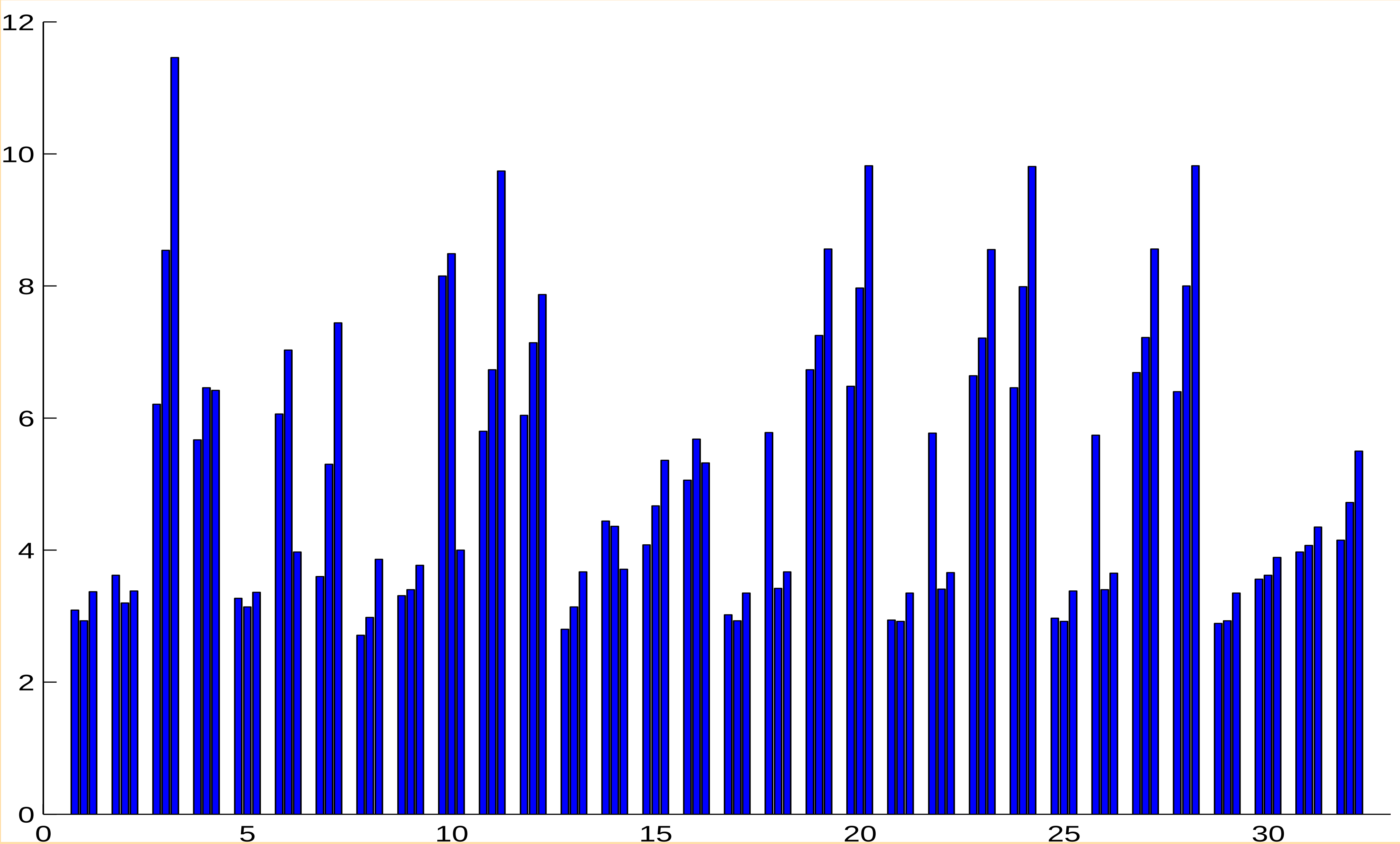
7. Numerical results

3D symmetric problems



7. Numerical results

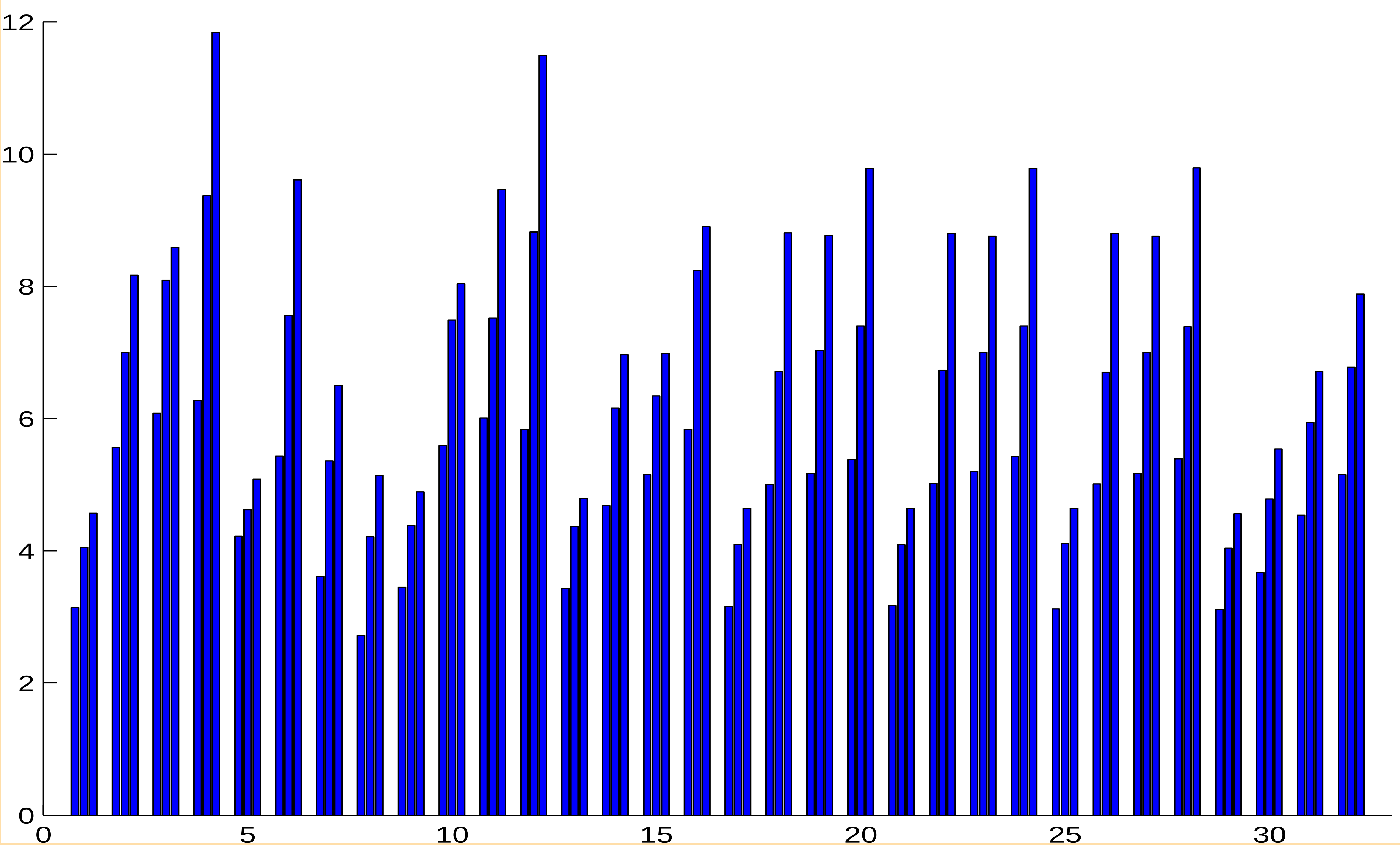
2D nonsymmetric problems



7. Numerical results

ULB

3D nonsymmetric problems



7. Numerical results

ULB

Comparison with other methods

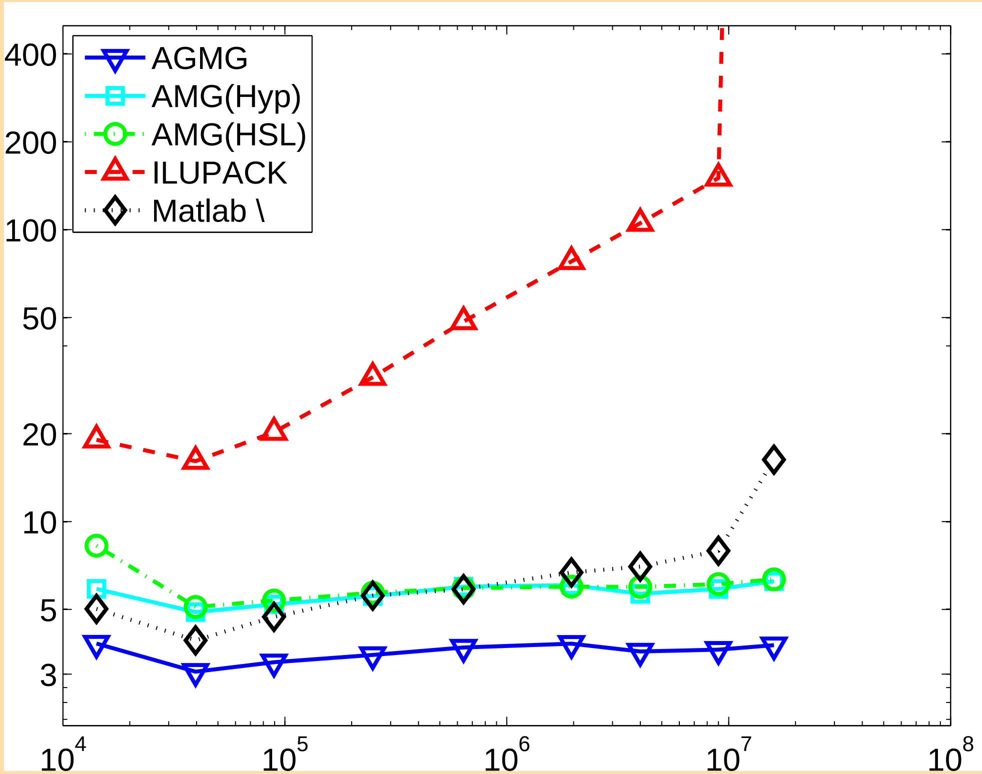
- AMG(Hyp): classical AMG method as implemented in the Hypre library (Boomer AMG)
- AMG(HSL): the classical AMG method as implemented in the HSL library
- ILUPACK: efficient threshold-based ILU preconditioner
- Matlab \: Matlab sparse direct solver (UMFPACK)

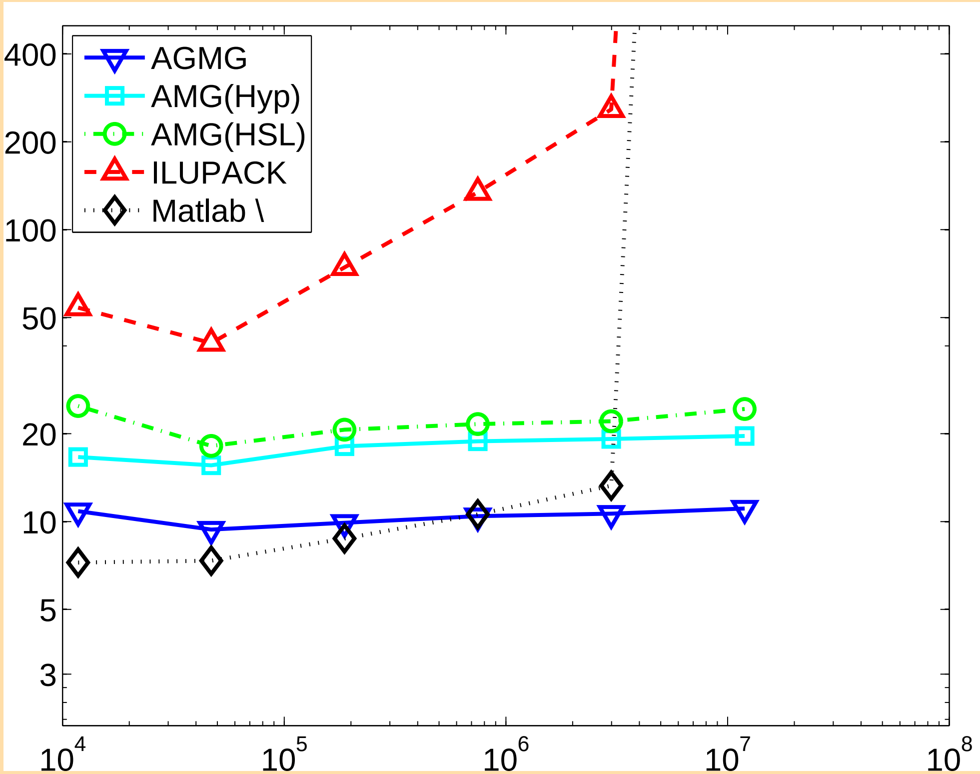
All methods but the last with Krylov subspace acceleration

7. Numerical results

ULB

Poisson 2D, FD	Laplace 2D, FE(p3)



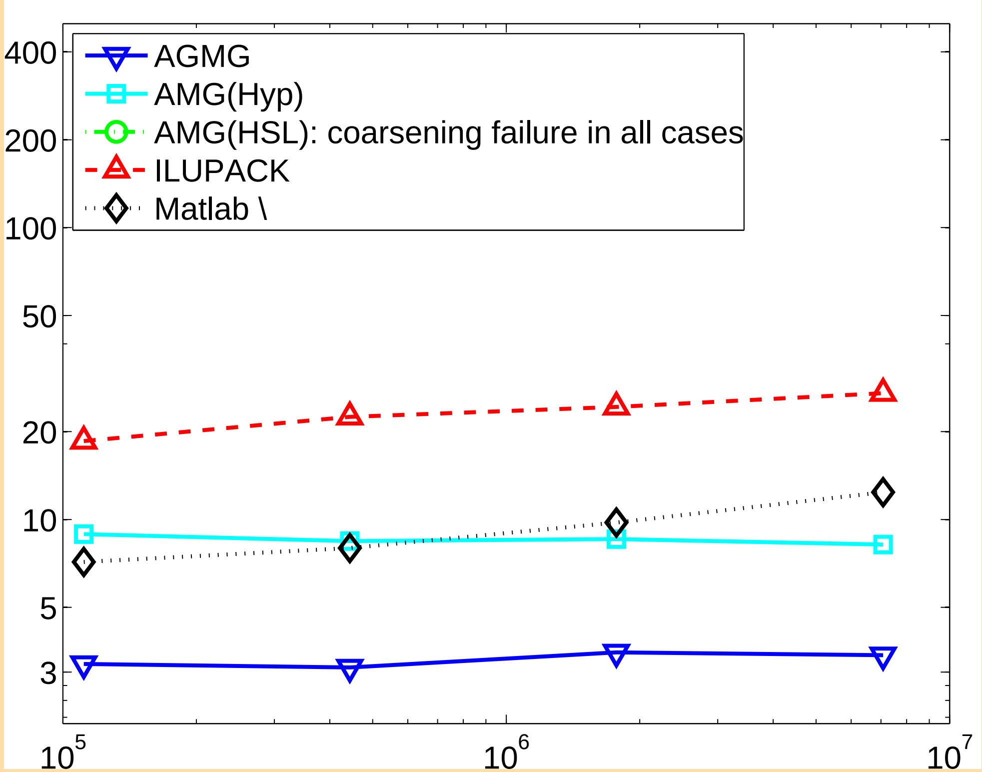


33% of nonzero offdiag > 0

7. Numerical results

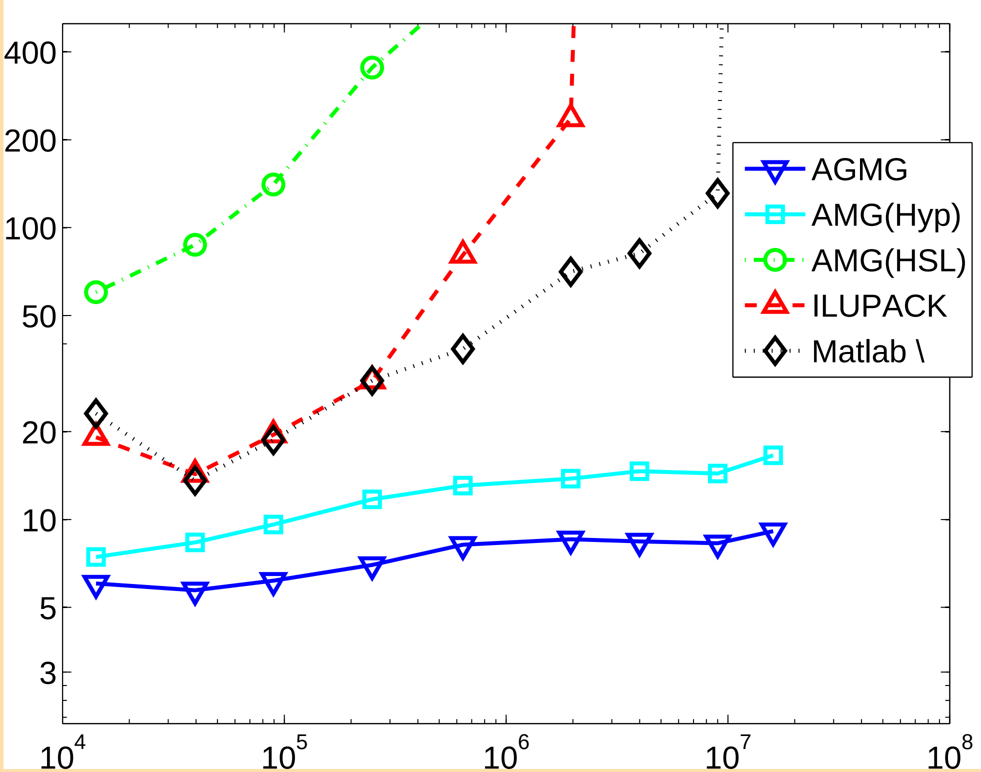
ULB

Poisson 2D, L-shaped, FE
Unstructured, Local refin.



Convection-Diffusion 2D, FD

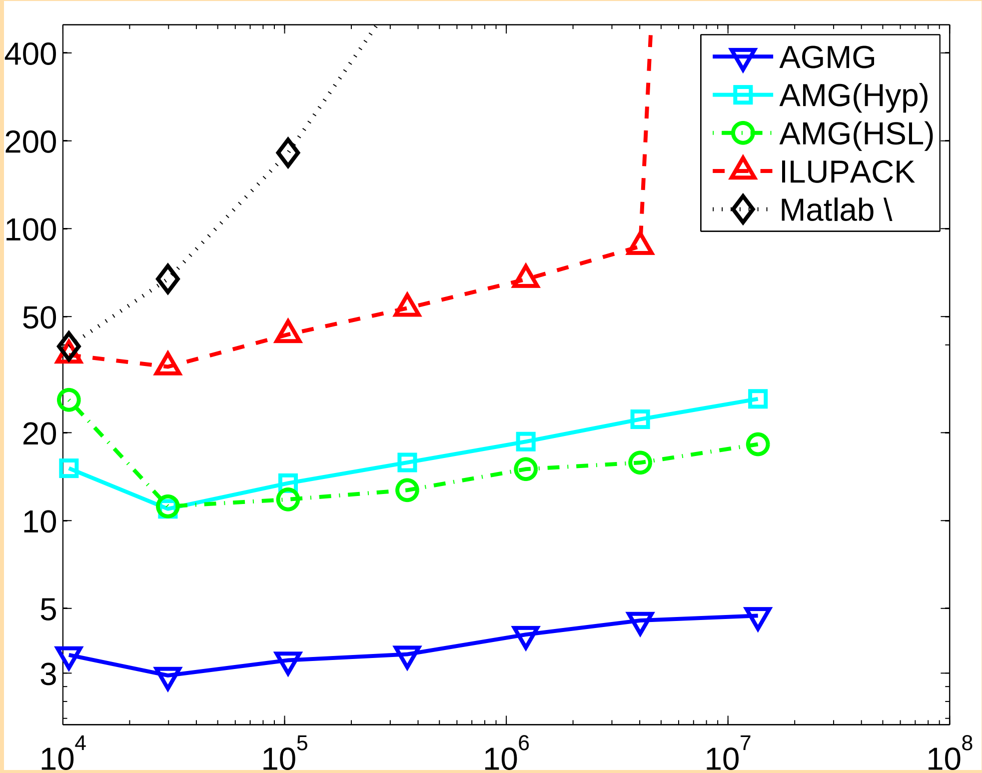
$\nu = 10^{-6}$



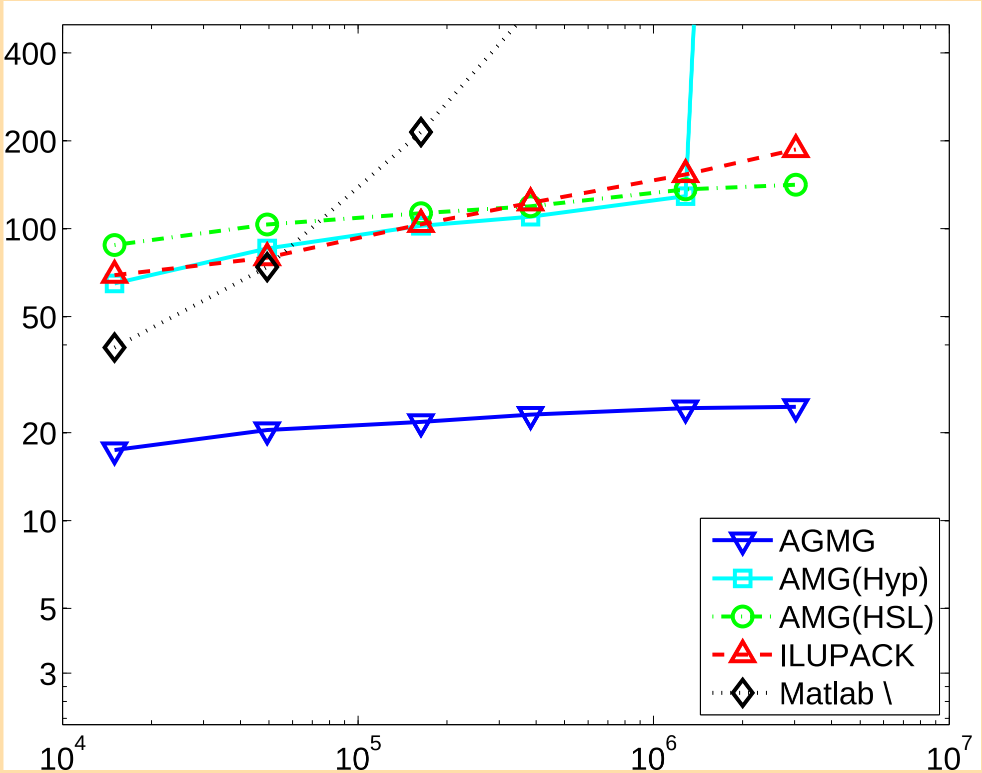
7. Numerical results

ULB

POISSON 3D, FD



LAPLACE 3D, FE(P3)

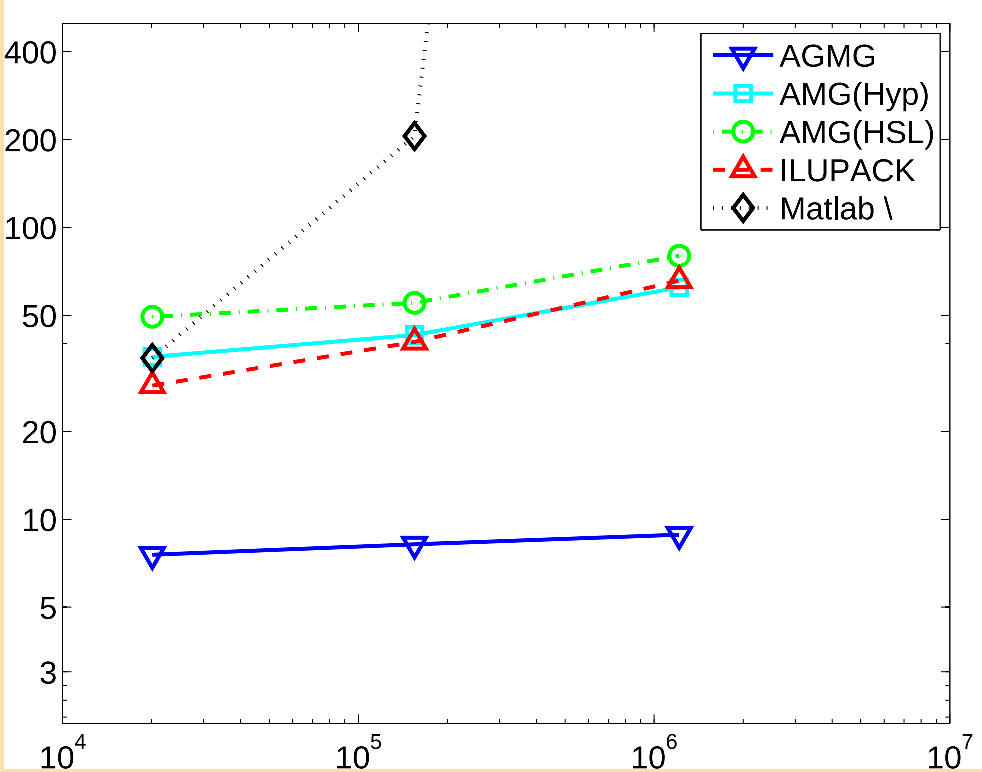


51% of nonzero offdiag > 0

7. Numerical results

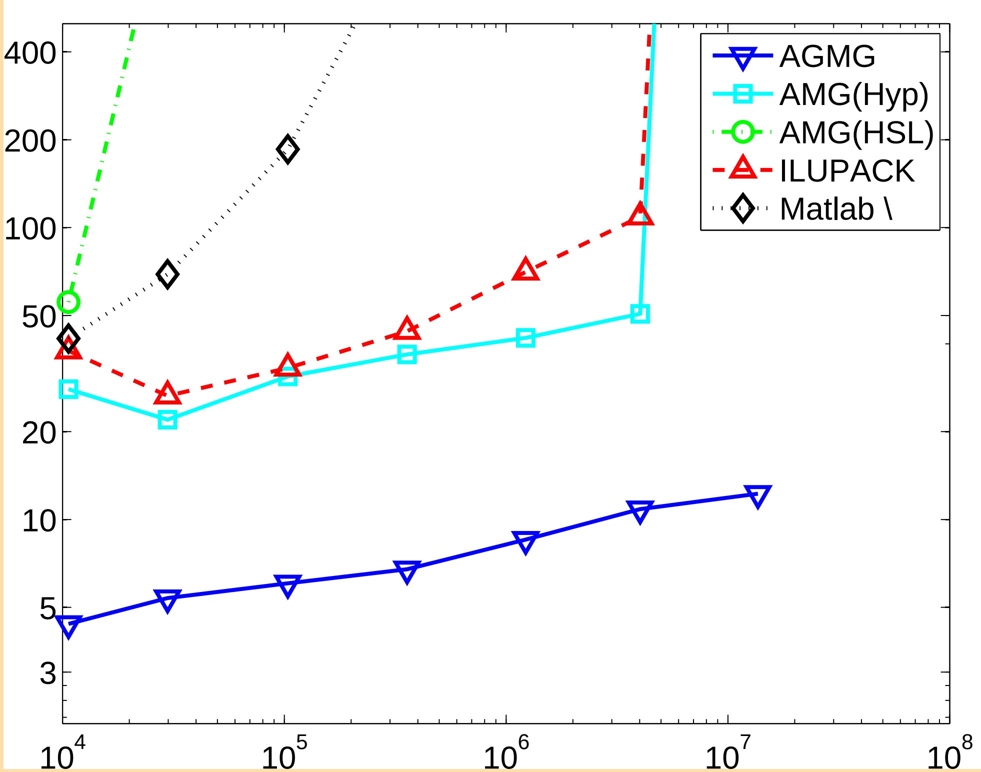
Poisson 3D, FE

Unstructured, Local refin.



Convection-Diffusion 3D, FD

$\nu = 10^{-6}$



7. Numerical results

Parallel run: with direct coarsest grid solver
Cray, 32 cores/node with 1GB/node
Poisson, 3D trilinear hexahedral FE

#Nodes	#Cores	$n/10^6$	#Iter.	Setup Time	Solve Time
1	32	31	17	5.9	40.4
2	64	63	17	6.2	40.8
4	128	125	17	6.9	41.5
8	256	251	17	9.5	41.8
16	512	501	17	14.8	42.5
32	1024	1003	17	27.3	44.0
64	2048	2007	17	69.0	48.2
128	4096	4014	17	383.0	59.4

(By courtesy of Mark Walkley, Univ. of Leeds)

7. Numerical results

ULB

Parallel run: with (new) iterative coarsest grid solver
Intel(R) Xeon(R) CPU E5649 @ 2.53GHz
3D problem with jumps, FD

#Nodes	#Cores	$n/10^6$	#Iter.	Setup Time	Solve Time
1	8	64	12	14.9	89.
16	128	1026	16	17.4	191.
48	384	3065	14	18.0	165.
96	768	6155	13	17.4	170.

8. Conclusions

ULB

- Robust method for scalar elliptic PDEs

8. Conclusions

- Robust method for scalar elliptic PDEs
- Purely algebraic convergence theory:
  do not depend on FE spaces, regularity assumption;
  applies also to the nonsymmetric case.

8. Conclusions

- Robust method for scalar elliptic PDEs
- Purely algebraic convergence theory:
  do not depend on FE spaces, regularity assumption;
  applies also to the nonsymmetric case.
- Can be used (and is used!) black box
  (does not require tuning or adaptation)

8. Conclusions

- Robust method for scalar elliptic PDEs
- Purely algebraic convergence theory:
  do not depend on FE spaces, regularity assumption;
  applies also to the nonsymmetric case.
- Can be used (and is used!) black box
  (does not require tuning or adaptation)
- Faster than some solvers based on classical AMG

8. Conclusions

ULB

- Robust method for scalar elliptic PDEs
- Purely algebraic convergence theory:
  do not depend on FE spaces, regularity assumption;
  applies also to the nonsymmetric case.
- Can be used (and is used!) black box
  (does not require tuning or adaptation)
- Faster than some solvers based on classical AMG
- Fairly small setup time: especially well suited when
  only a modest accuracy is needed
  (e.g., linear solve within Newton steps)

8. Conclusions

- Robust method for scalar elliptic PDEs
- Purely algebraic convergence theory:
  do not depend on FE spaces, regularity assumption;
  applies also to the nonsymmetric case.
- Can be used (and is used!) black box
  (does not require tuning or adaptation)
- Faster than some solvers based on classical AMG
- Fairly small setup time: especially well suited when only a modest accuracy is needed
  (e.g., linear solve within Newton steps)
- Efficient parallelization

8. Conclusions

ULB

- Robust method for scalar elliptic PDEs
- Purely algebraic convergence theory:
  do not depend on FE spaces, regularity assumption;
  applies also to the nonsymmetric case.
- Can be used (and is used!) black box
  (does not require tuning or adaptation)
- Faster than some solvers based on classical AMG
- Fairly small setup time: especially well suited when
  only a modest accuracy is needed
  (e.g., linear solve within Newton steps)
- Efficient parallelization
- Professional code available, free academic license

References

- Analysis of aggregation–based multigrid (with A. C. Muresan), SISC (2008).
- Recursive Krylov-based multigrid cycles (with P. S. Vassilevski), NLAA (2008).
- An aggregation-based algebraic multigrid method, ETNA (2010).
- Algebraic analysis of two-grid methods: the nonsymmetric case, NLAA (2010).
- Algebraic analysis of aggregation-based multigrid, (with A. Napov) NLAA (2011).
- An algebraic multigrid method with guaranteed convergence rate (with A. Napov), SISC (2012).
- Aggregation-based algebraic multigrid for convection-diffusion equations, SISC (2012, to appear).

AGMG software: Google AGMG
(`http://homepages.ulb.ac.be/~ynotay/AGMG`)

Thank you for your attention !