Robust solution of Poisson-like problems with aggregation-based AMG

Yvan Notay*

Université Libre de Bruxelles
Service de Métrologie Nucléaire

Paris, January 26, 2015

* Supported by the Belgian FNRS

http://homepages.ulb.ac.be/∼ynotay

Outline

ULB

1. Introduction
2. Why multigrid
3. AMG & Aggregation
4. AGMG
   - Robustness study
   - Comparative study
5. Parallelization of AGMG
6. Conclusions

1. Introduction (1)

ULB

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

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

(or closely related).

Thus we need a good solver for

discrete Poisson-like problems

1. Introduction (1)

ULB

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

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

(or closely related).

Thus we need a good solver for

discrete Poisson-like problems

Efficiently:

robustly (stable performances)

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

1. Introduction (2)

ULB

Discrete Poisson-like problems

- For not too large 2D problems, direct methods OK (solve the system in linear time)
  $\rightarrow$ de facto standard for a long time

1. Introduction (2)

ULB

Discrete Poisson-like problems

- For not too large 2D problems, direct methods OK (solve the system in linear time)
  $\rightarrow$ de facto standard for a long time
- Large 3D problems: untractable for direct methods (In addition, scale poorly in parallel)
  $\rightarrow$ Iterative solvers needed

1. Introduction (2)

ULB

Discrete Poisson-like problems

- For not too large 2D problems, direct methods OK (solve the system in linear time)
  $\rightarrow$ de facto standard for a long time
- Large 3D problems: untractable for direct methods (In addition, scale poorly in parallel)
  $\rightarrow$ Iterative solvers needed
- Multigrid method are good candidates (see below)
  But to substitute a direct solver we need
  - a black box solver
    (You provide the matrix & rhs, I return the solution)
  - that is robust
    (convergence not affected by changes in the BC, PDE coeff., geometry & discretization grid)

2. Why multigrid (1)

ULB

Standard iterative methods typically very slow

Consider the error from different scales:

small scale $\rightarrow$ strong local variations

large scale $\rightarrow$ smooth variations

Iterative methods mainly act locally, i.e. damp error modes seen from small scale, but not much smooth modes

More steps needed to propagate information about smooth modes as the grid is refined

(linear time out of reach)

2. Why multigrid (1)

ULB

Standard iterative methods typically very slow

Consider the error from different scales:

small scale $\rightarrow$ strong local variations

large scale $\rightarrow$ smooth variations

Iterative methods mainly act locally, i.e. damp error modes seen from small scale, but not much smooth modes

More steps needed to propagate information about smooth modes as the grid is refined

(linear time out of reach)

Multigrid: solving the problem on a coarser grid (less unknowns $\rightarrow$ easier) yields an approximate solution which is essentially correct from the large scale viewpoint

2. Why multigrid (2)

ULB

A multigrid method alternates:

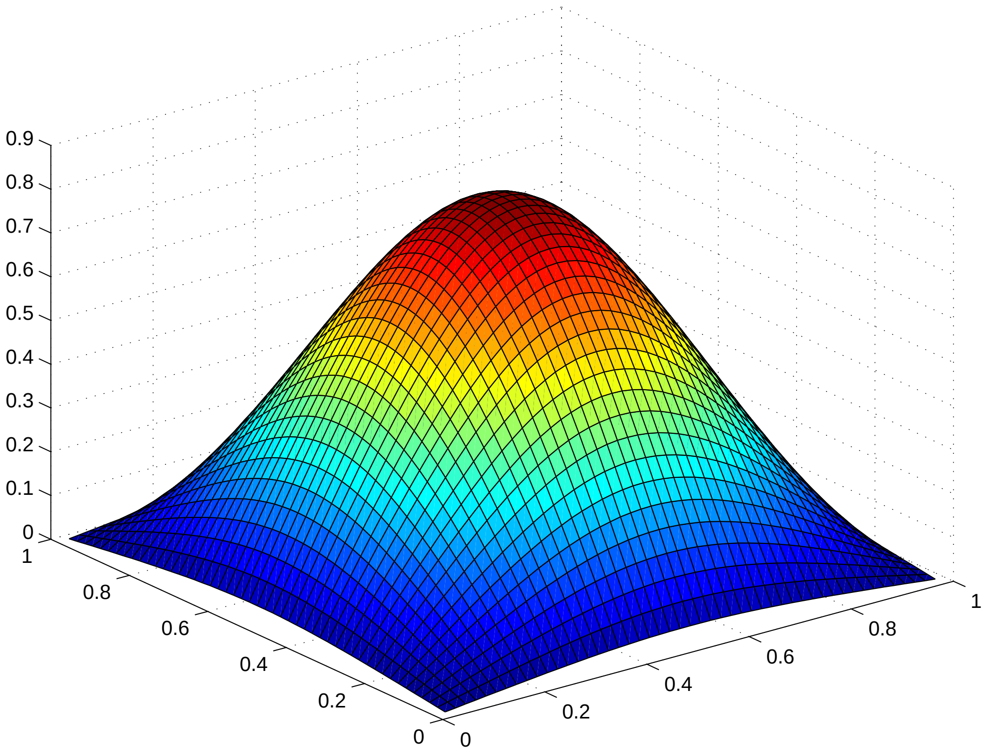
- smoothing iterations: improve current solution $\mathbf{u}_k$ using a basic iterative method $\rightarrow \widetilde{\mathbf{u}}_k$
- coarse grid correction:
  - project the residual equation
    $A(\mathbf{u}-\widetilde{\mathbf{u}}_k)=\mathbf{b}-A\,\widetilde{\mathbf{u}}_k\equiv\mathbf{r}$
    on the coarse grid: $\mathbf{r}_c = R\,\mathbf{r}$ (R : $n_c \times n$)
  - solve the coarse problem: $\mathbf{v}_c = A_c^{-1}\mathbf{r}_c$
  - prolongate (interpolate) coarse correction on the fine grid: $\mathbf{u}_{k+1}=\widetilde{\mathbf{u}}_k + P\,\mathbf{v}_c$ (P : $n \times n_c$)

2. Why multigrid (3)

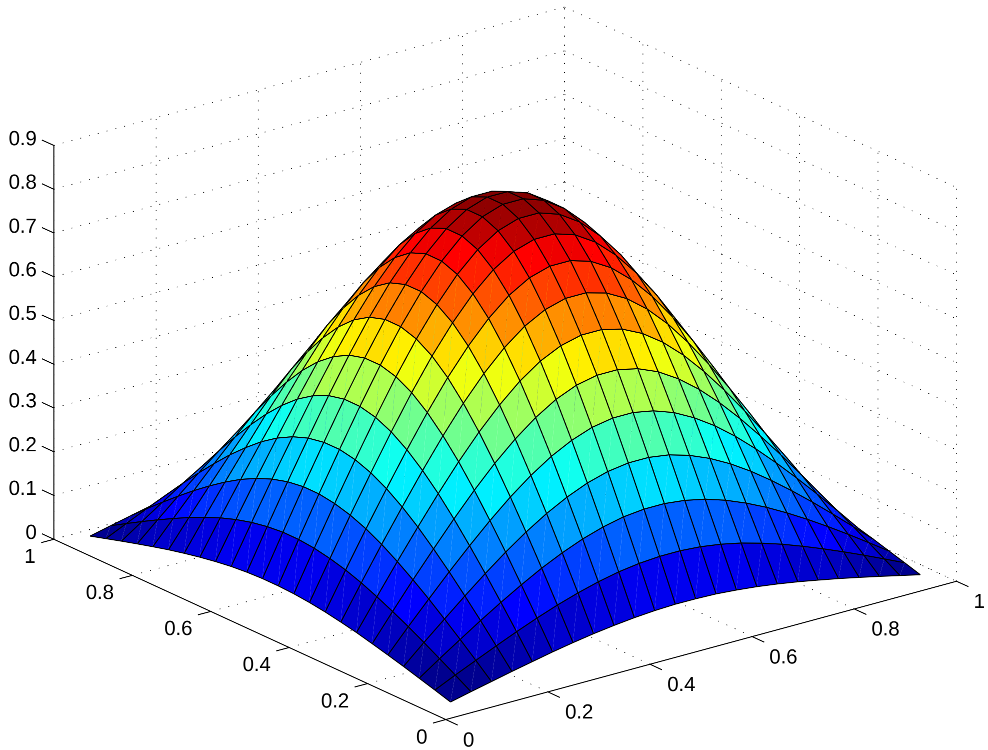
Example

$$-\Delta u = 20\, e^{-10\,((x-0.5)^2+(y-0.5)^2)} \quad \text{in } \Omega = (0,1)\times(0,1)$$
$$u = 0 \quad \text{on } \partial\Omega$$

Uniform grid with mesh size h, five-point finite difference.



Solution with $h^{-1} = 50$

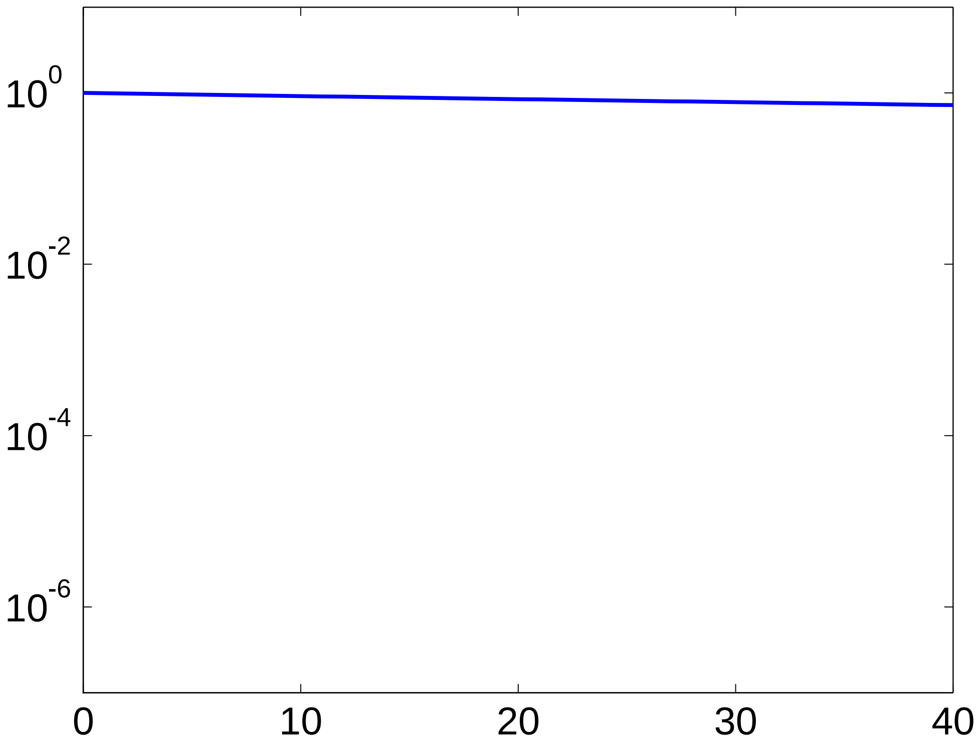


Solution with $h^{-1} = 25$

Simple iterative methods are not efficient

Example: symmetric Gauss–Seidel iterations (1 forward Gauss–Seidel sweep + 1 backward Gauss–Seidel sweep)

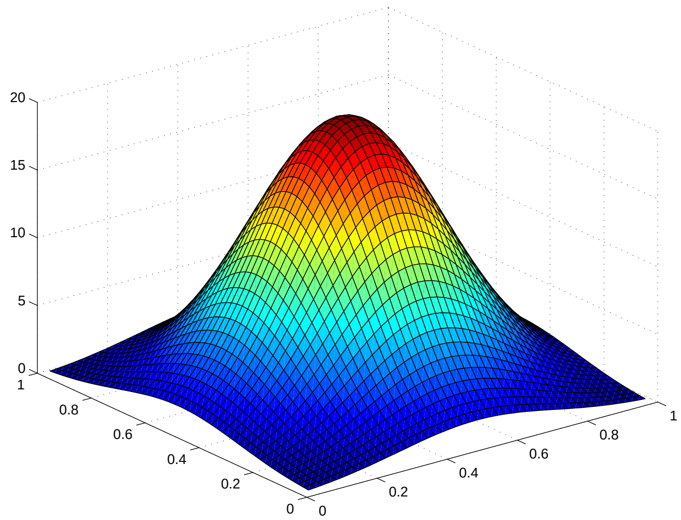
$\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}$ – vs – number of iter.



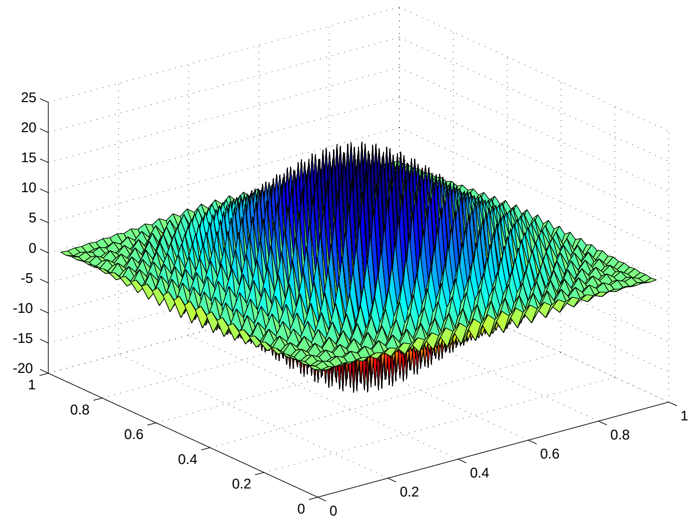
$\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} = 0.72$ after 40 steps

2. Why multigrid (5)

Initial residual (r.h.s.)



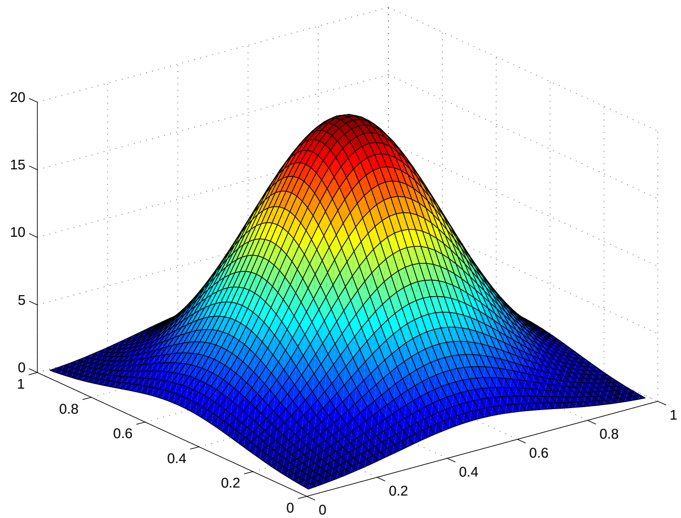
Residual after solve on the coarse grid



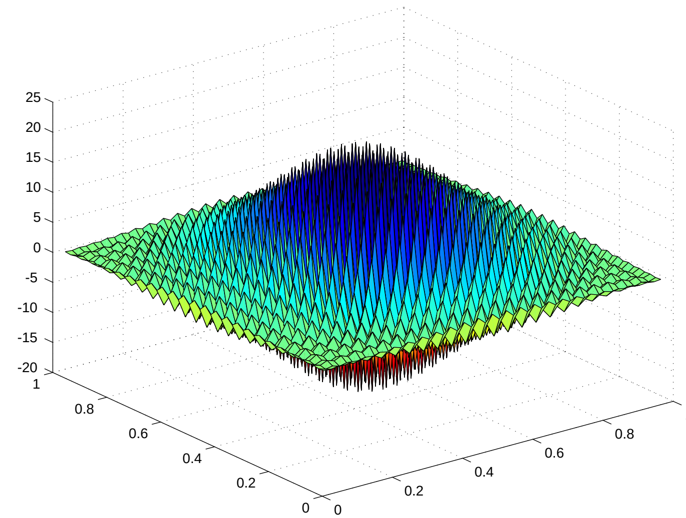
$$\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} = 0.71$$

2. Why multigrid (5)

Initial residual (r.h.s.)

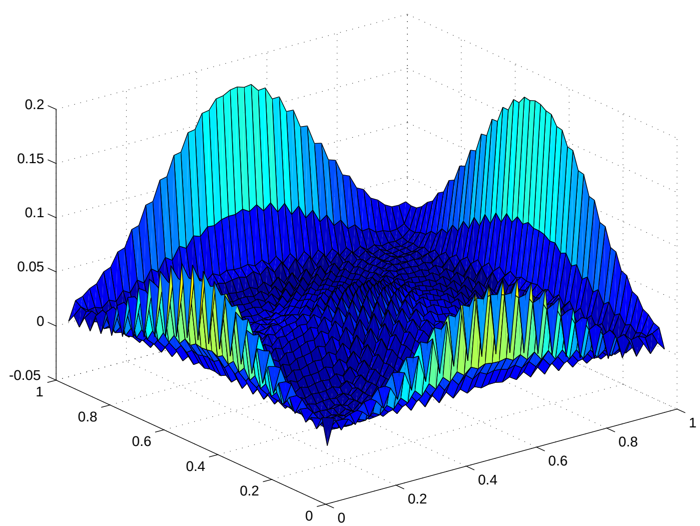


Residual after solve on the coarse grid



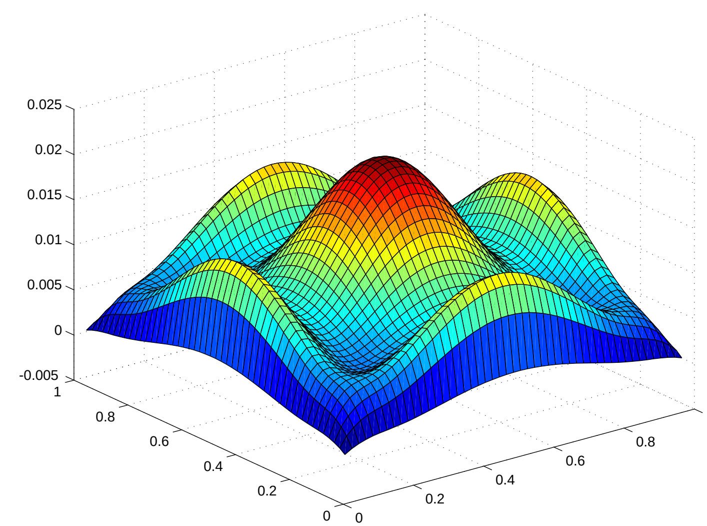
$$\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} = 0.71$$

+ 1 sym. Gauss-Seidel step



$$\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} = 0.0039$$

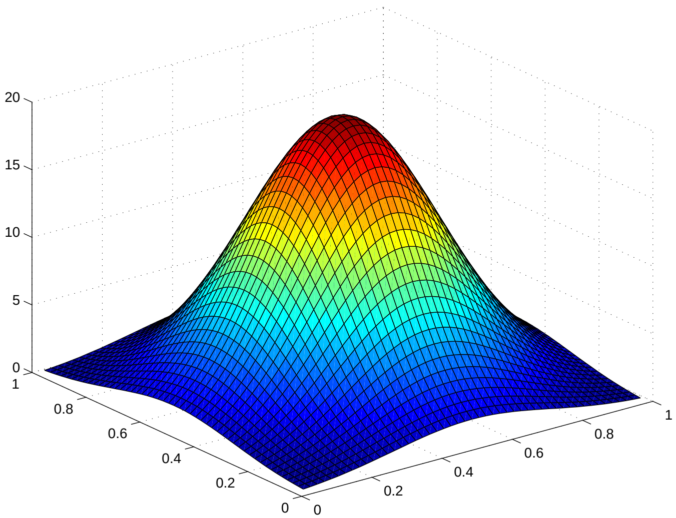
+ 8 sym. Gauss-Seidel steps



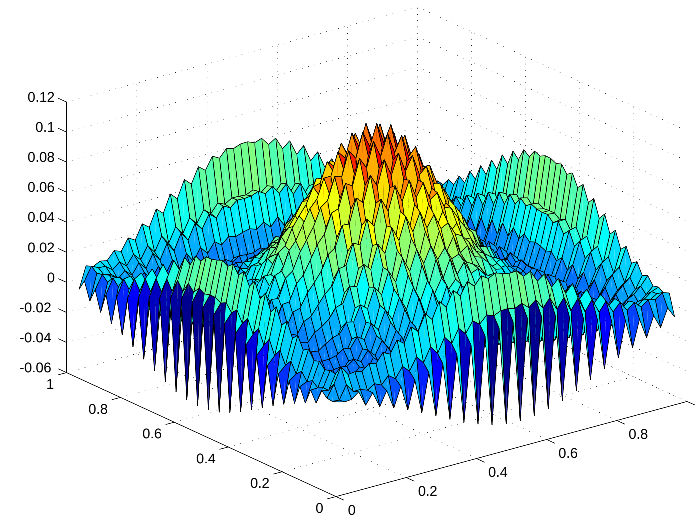
$$\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} = 0.0012$$

2. Why multigrid (6)

Initial residual

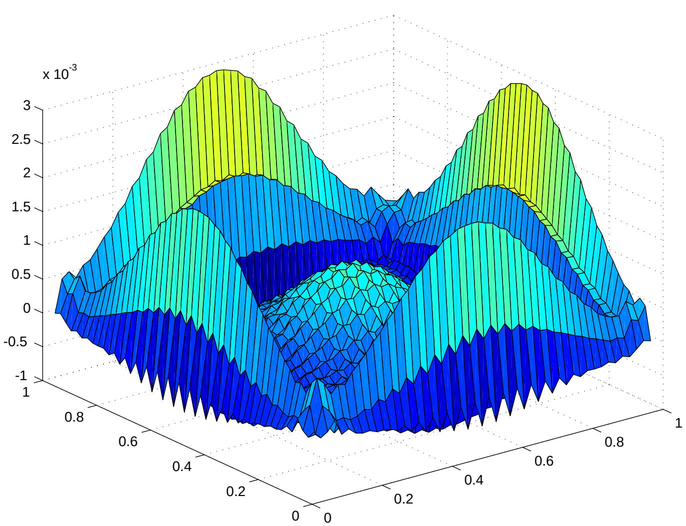


1 $\times$ (SGS – coarse solve – SGS)



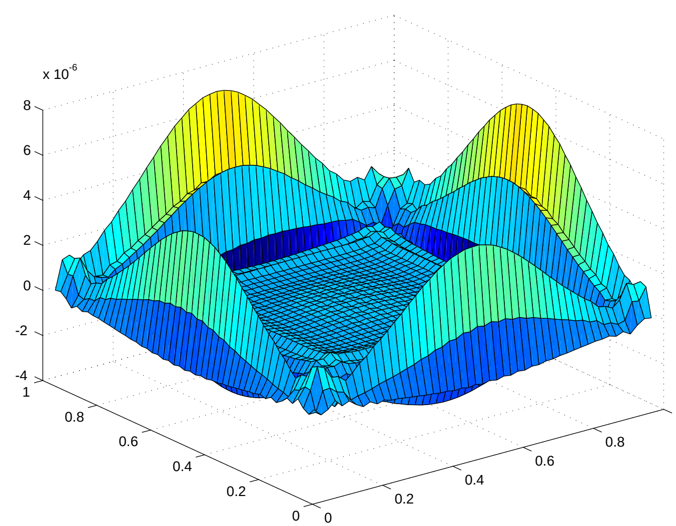
$$\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} = 3.9\,10^{-3}$$

2 $\times$ (SGS – coarse solve – SGS)



$$\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} = 7.7\,10^{-5}$$

4 $\times$ (SGS – coarse solve – SGS)



$$\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} = 1.7\,10^{-7}$$

3. AMG & Aggregation (1)

- Geometric Multigrid is not black box
  Often also not robust (e.g., influenced by BC)

3. AMG & Aggregation (1)

(p. 11)

ULB

- Geometric Multigrid is not black box
  Often also not robust (e.g., influenced by BC)
- Classical Algebraic Multigrid (AMG)
  - Attempt to imitate geometric MG in black box mode
  - With robustness enhancements
  - Issues still open after 30 years of research
  - New issues came with massive parallelism

3. AMG & Aggregation (1)

(p. 11)

ULB

- Geometric Multigrid is not black box
  Often also not robust (e.g., influenced by BC)

- Classical Algebraic Multigrid (AMG)
  - Attempt to imitate geometric MG in black box mode
  - With robustness enhancements
  - Issues still open after 30 years of research
  - New issues came with massive parallelism

- Aggregation-based AMG
  - Overlooked for a long time, revival since 2007
  - Solves issues of classical AMG in a natural way
  - Faster and more robust (controversial)

3. AMG & Aggregation (2)

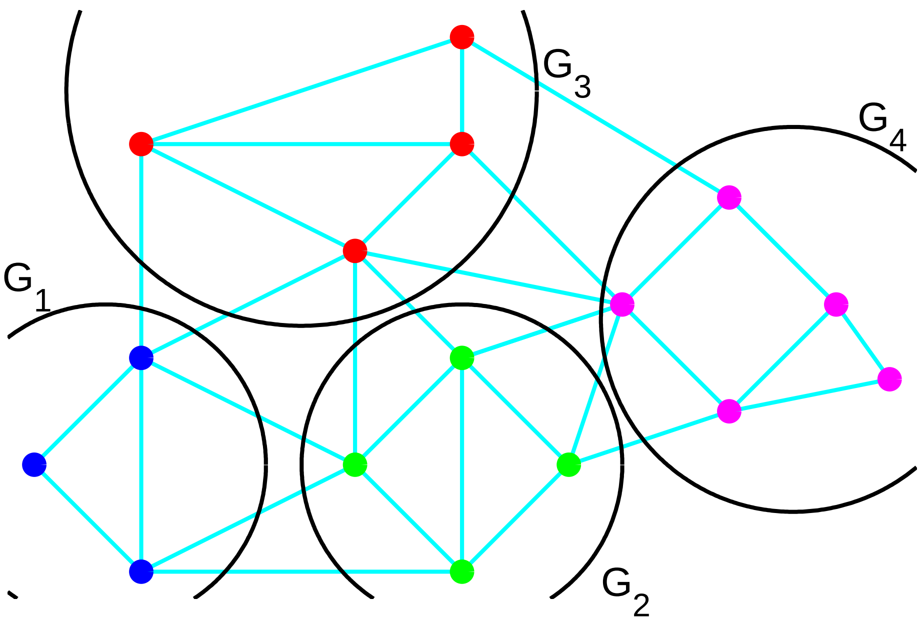
ULB

Aggregation-based AMG

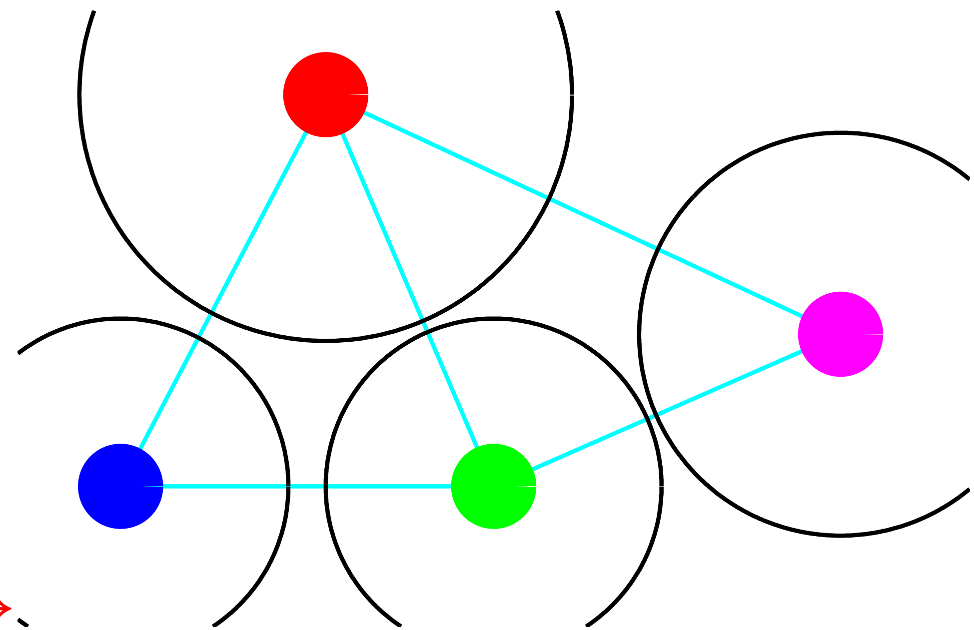
Coarse unknowns: obtained by mere aggregation

Coarse grid matrix: obtained by a simple summation

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



$\rightarrow$



3. AMG & Aggregation (2)

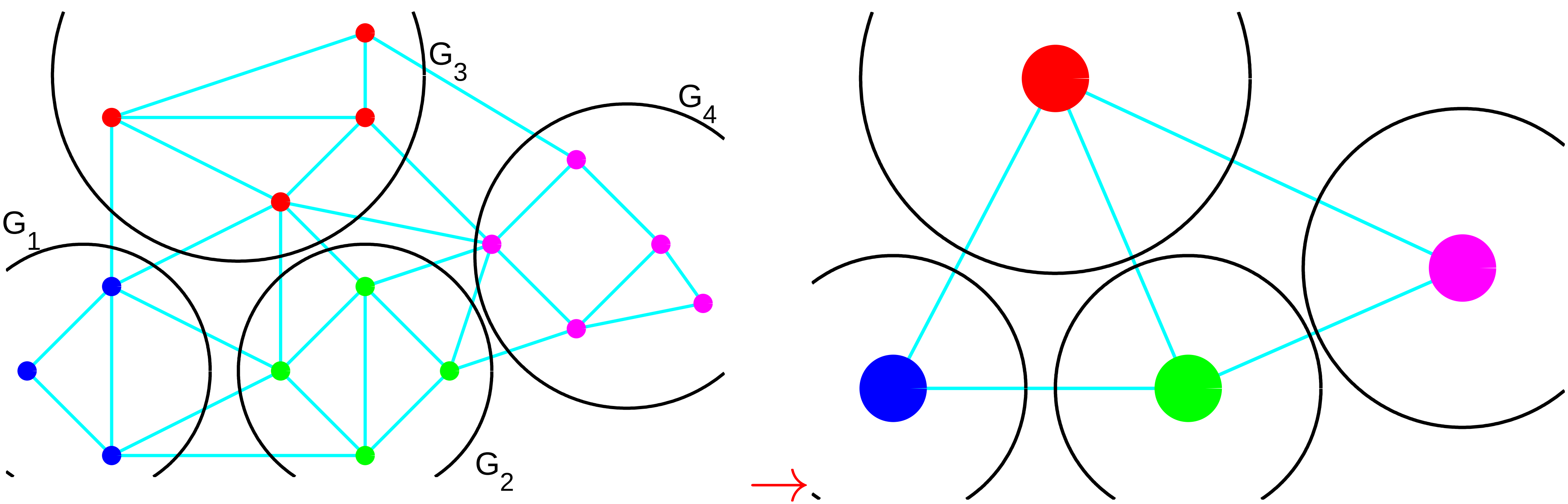
ULB

Aggregation-based AMG

Coarse unknowns: obtained by mere aggregation

Coarse grid matrix: obtained by a simple summation

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



In parallel: Aggregates formed with unknowns assigned to a same process $\rightarrow$ natural parallelization

3. AMG & Aggregation (3)

ULB

How to solve the coarse problem?

By recursivity:

- apply the same two-grid scheme at the coarse level
- do only a few iteration (cost)
- $\rightarrow$ go to a further coarse level: recursivity again
- $\rightarrow$ and so on, until problem size is small enough

How to solve the coarse problem?

By recursivity:

- apply the same two-grid scheme at the coarse level
- do only a few iteration (cost)
- $\rightarrow$ go to a further coarse level: recursivity again
- $\rightarrow$ and so on, until problem size is small enough

Geometric MG & Classical AMG:

use the V-cycle – 1 iteration at each level

ULB

How to solve the coarse problem?

By recursivity:

- apply the same two-grid scheme at the coarse level
- do only a few iteration (cost)
- $\rightarrow$ go to a further coarse level: recursivity again
- $\rightarrow$ and so on, until problem size is small enough

Geometric MG & Classical AMG:

use the V-cycle – 1 iteration at each level

Aggregation-based AMG:

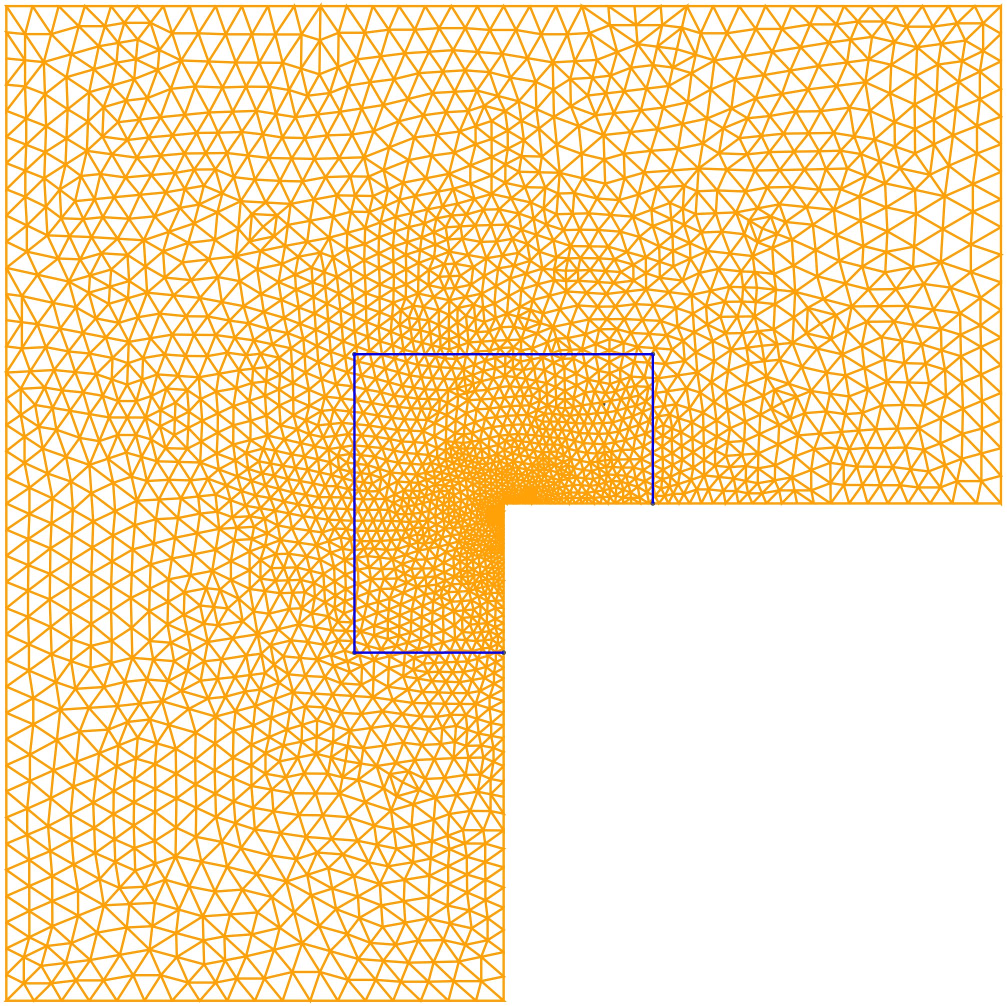
use the K-cycle – 2 iterations at each level
with Krylov (CG) acceleration

Downside of the simplicity, but not an issue

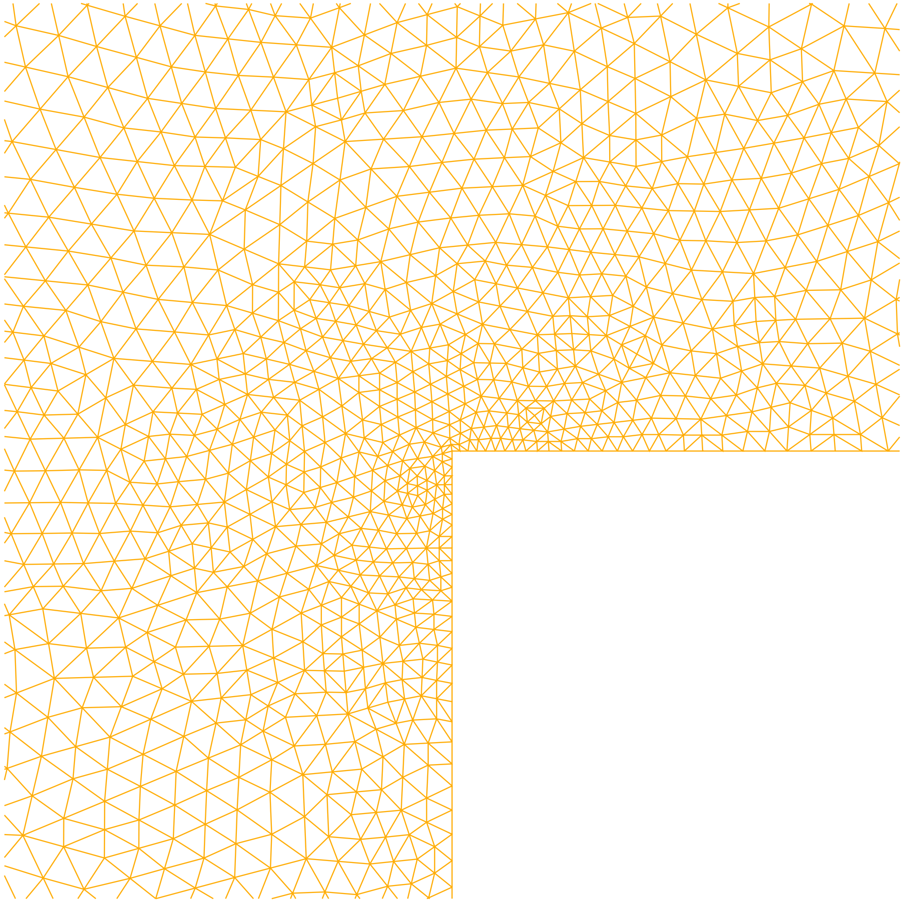
3. AMG & Aggregation (4)

ULB

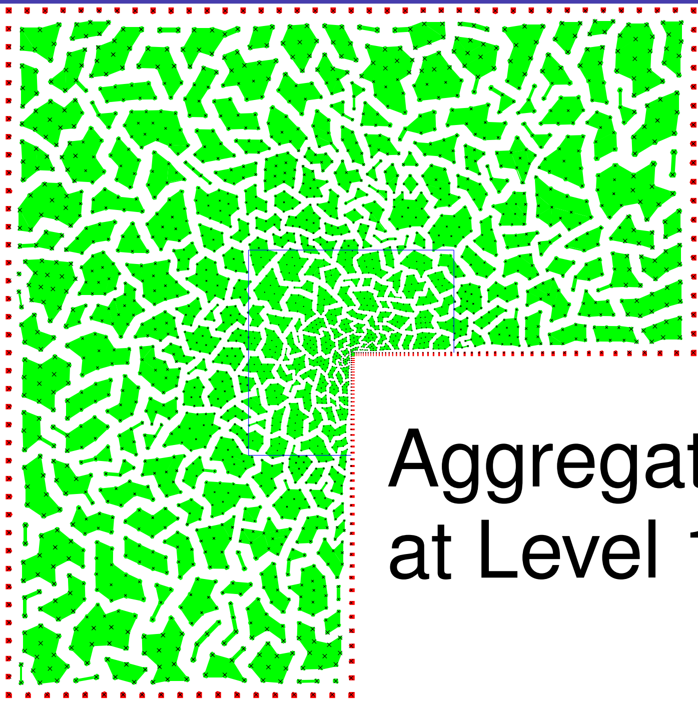
Example: recursive Quality Aware Aggregation
for the discrete Poisson linear finite element
matrix associated with the mesh:



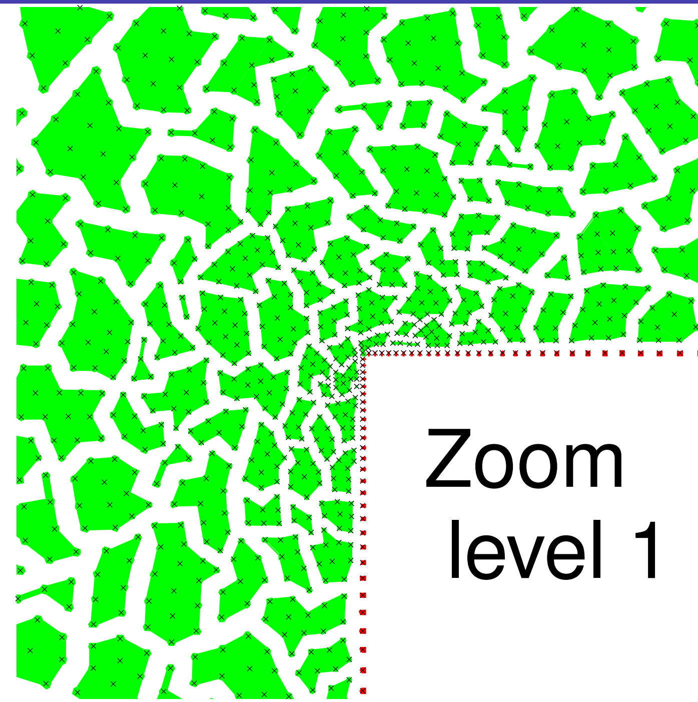
Zoom:



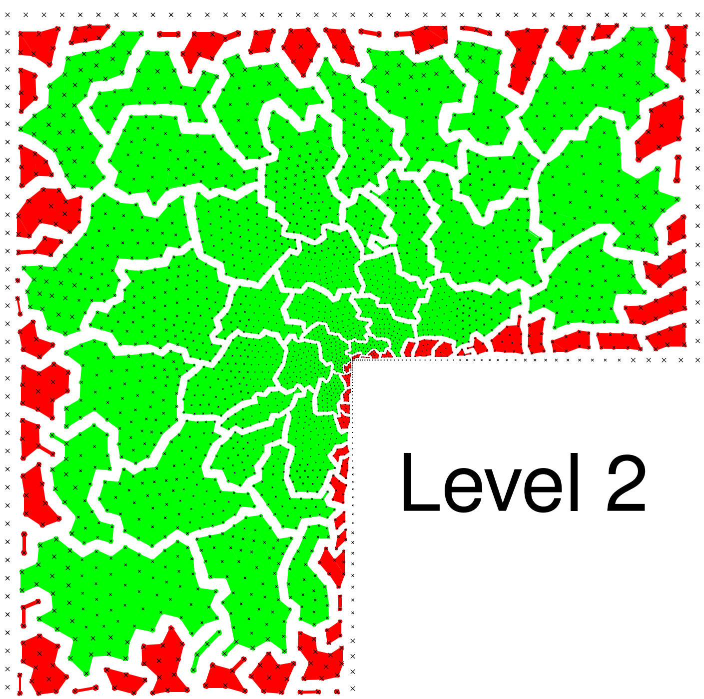
3. AMG & Aggregation (5)



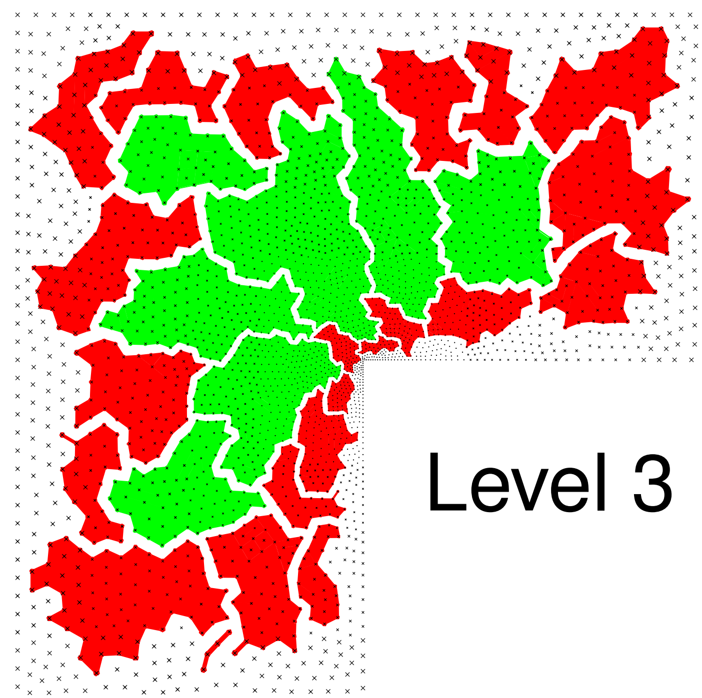
Aggregation at Level 1



Zoom level 1



Level 2



Level 3

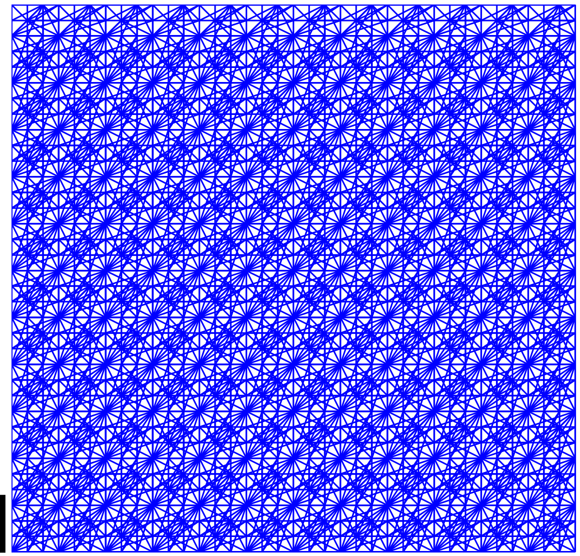
3. AMG & Aggregation (6)

Aggregation works also for higher order FE matrices

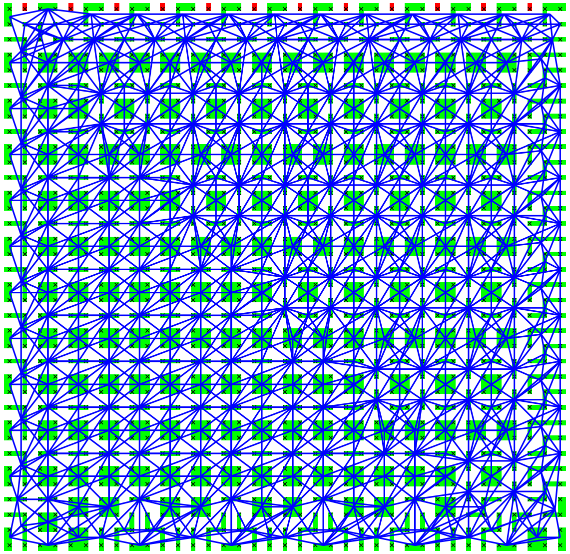
Example:

3rd order (P3)

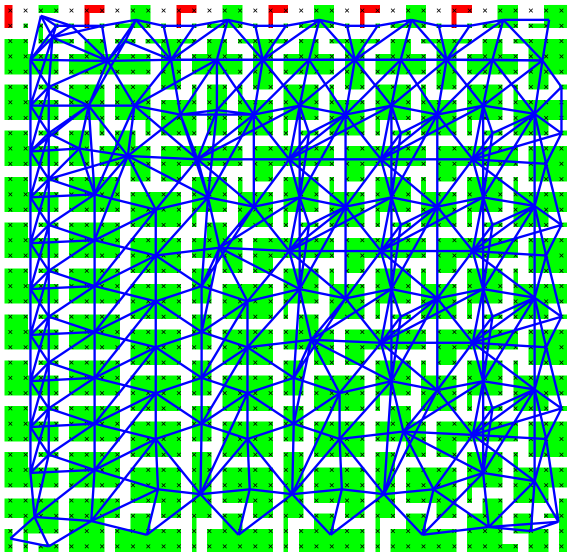
$nnz(A) \approx 16\,n$



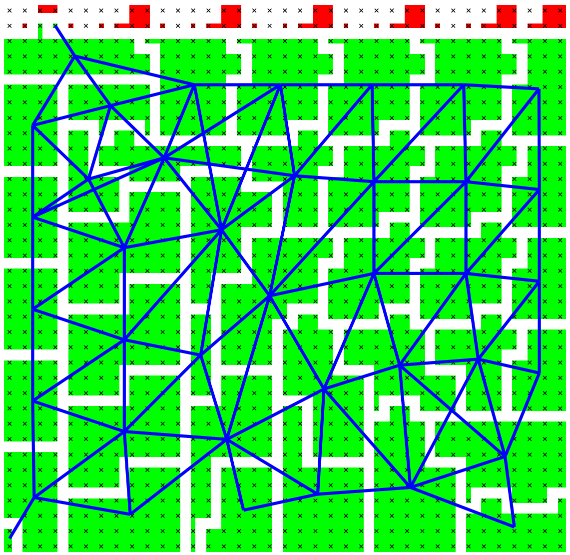
Fine grid



Level 1



Level 2

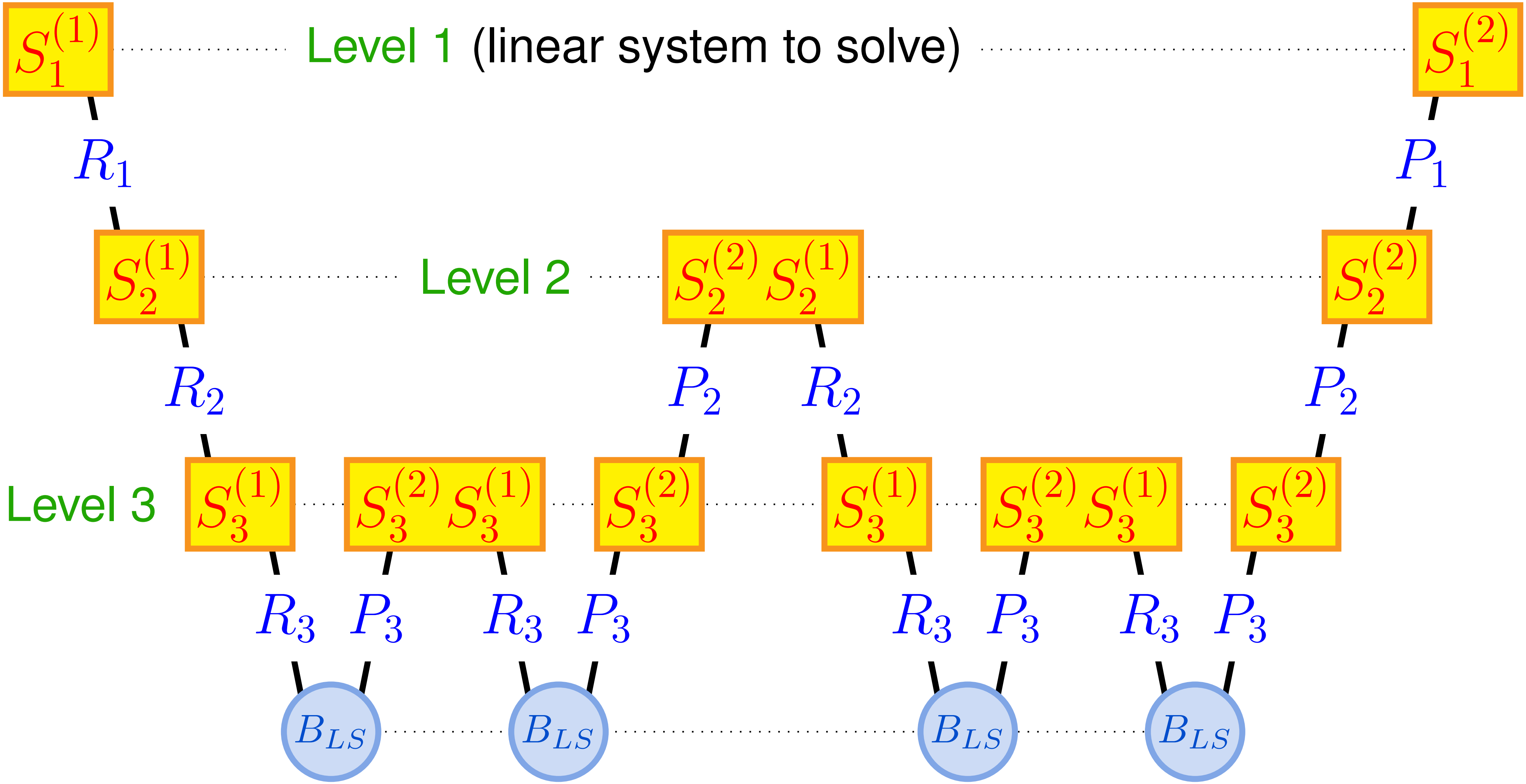


Level 3

3. AMG & Aggregation (7)

ULB

Solve phase: Workflow for 1 iteration using the K-cycle (4 levels)



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

Iterative solution with AGgregation-based algebraic MultiGrid

Linear system solver software package

- Black box
- FORTRAN 90 (easy interface with C & C++)
- Matlab interface

```
>> x=agmg(A,y);
>> x=agmg(A,y,1);   % SPD case
```

- Free academic license

4. AGMG: Test suite (1)

(p. 19)

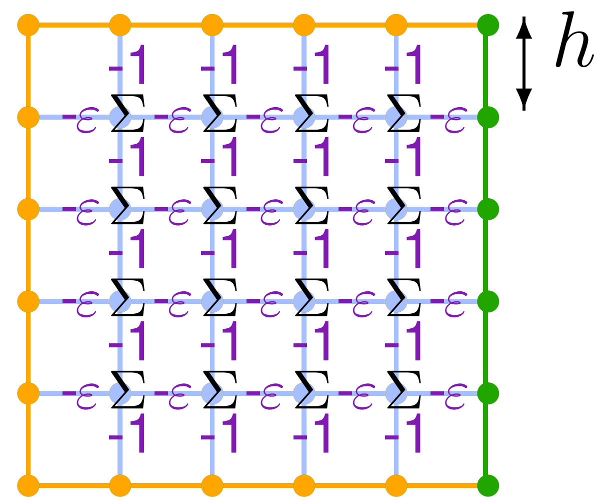
MODEL2D : 5-point & 9 point discretizations of

$$\begin{cases} -\Delta u = 1 & \text{on } \Omega = [0,1]\times[0,1] \\ u = 0 & \text{on } \partial\Omega \end{cases}$$

with $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

ANI2D : 5-point discretization of

$$\begin{cases} -\nabla\cdot D\nabla u = 1 & \text{on } \Omega = [0,1]\times[0,1] \\ u = 0 & \text{for } x = 1\,,\ 0 \le y \le 1 \\ \frac{\partial u}{\partial n} = 0 & \text{elsewhere on } \partial\Omega \end{cases}$$



$\Sigma = 2(1+\varepsilon)$

with $\nabla = \mathbf{1}_x\frac{\partial}{\partial x} + \mathbf{1}_y\frac{\partial}{\partial y}$, $D = \mathrm{diag}(\,\varepsilon\,,\ 1\,)$

Tested: $\varepsilon = 10^{-2}\,,\ 10^{-4}$

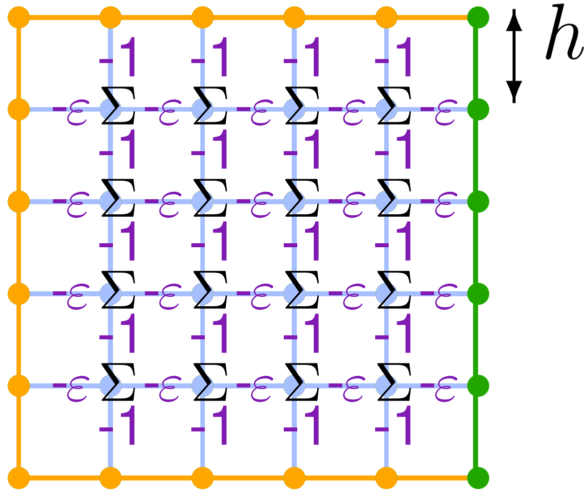
ULB

Non M-matrices

ANI2DBIFE :

bilinear FE element discretization of

$$\begin{cases} -\nabla \cdot D \nabla u = 1 & \text{on } \Omega = [0,1] \times [0,1] \\ u = 0 & \text{for } x = 1\,, \; 0 \le y \le 1 \\ \frac{\partial u}{\partial n} = 0 & \text{elsewhere on } \partial\Omega \end{cases}$$



$\Sigma = 2(1+\varepsilon)$

with $\nabla = \mathbf{1}_x \frac{\partial}{\partial x} + \mathbf{1}_y \frac{\partial}{\partial y}\,, \; D = \mathrm{diag}(\,\varepsilon\,,\,1\,)$

Tested: $\varepsilon = 10^{-2}\,,\, 10^{-2}\,,\, 10^{-3}$

(i.e. some strong positive offdiagonal elements)

4. AGMG: Test suite (3)

ULB

MODEL3D : 7-point discretization of

$$\begin{cases} -\Delta u = 1 & \text{on } \Omega = [0,1]\times[0,1]\times[0,1] \\ u = 0 & \text{on } \partial\Omega \end{cases}$$

with $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$

ANI3D : 7-point discretization of

$$\begin{cases} -\nabla\cdot D\nabla u = 1 & \text{on } \Omega = [0,1]\times[0,1]\times[0,1] \\ u = 0 & \text{for } x = 1\,,\ 0 \le y,\ z \le 1 \\ \frac{\partial u}{\partial n} = 0 & \text{elsewhere on } \partial\Omega \end{cases}$$

with $\nabla = \mathbf{1}_x\frac{\partial}{\partial x} + \mathbf{1}_y\frac{\partial}{\partial y} + \mathbf{1}_z\frac{\partial}{\partial z}$, $D = \mathrm{diag}(\,\varepsilon_x\,,\,\varepsilon_y\,,1\,)$

Tested: $(0.07, 1, 1)$,$(0.07, 0.25, 1)$, $(0.07, 0.07, 1)$, $(0.005, 1, 1)$, $(0.005, 0.07, 1)$, $(0.005, 0.005, 1)$

4. AGMG: Test suite (4)

(p. 22)

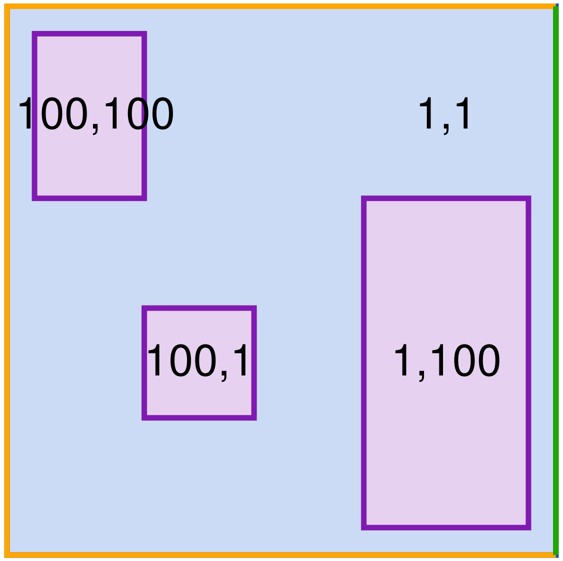
ULB

Problems with Jumps (FD)

JUMP2D : 5-point discretization of

$$\begin{cases} -\nabla \cdot D\nabla u = f & \text{on } \Omega = [0,1] \times [0,1] \\ u = 0 & \text{for } x = 1\,,\ 0 \le y \le 1 \\ \frac{\partial u}{\partial n} = 0 & \text{elsewhere on } \partial\Omega \end{cases}$$

with $\nabla = \mathbf{1}_x \frac{\partial}{\partial x} + \mathbf{1}_y \frac{\partial}{\partial y}$, $D = \mathrm{diag}(\, D_x\,,\ D_y\,)$

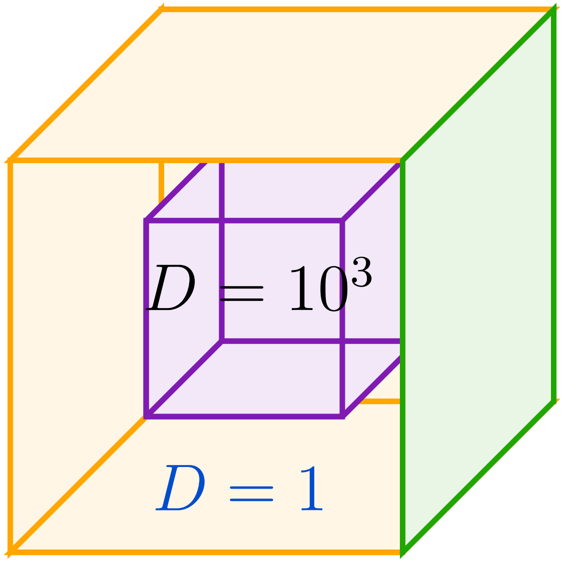


$D_x,\ D_y$

JUMP3D : 7-point discretization of

$$\begin{cases} -\nabla \cdot D\nabla u = f & \text{on } \Omega = [0,1] \times [0,1] \times [0,1] \\ u = 0 & \text{for } x = 1\,,\ 0 \le y,\ z\ \le 1 \\ \frac{\partial u}{\partial n} = 0 & \text{elsewhere on } \partial\Omega \end{cases}$$

with $\nabla = \mathbf{1}_x \frac{\partial}{\partial x} + \mathbf{1}_y \frac{\partial}{\partial y} + \mathbf{1}_z \frac{\partial}{\partial z}$



4. AGMG: Test suite (5)

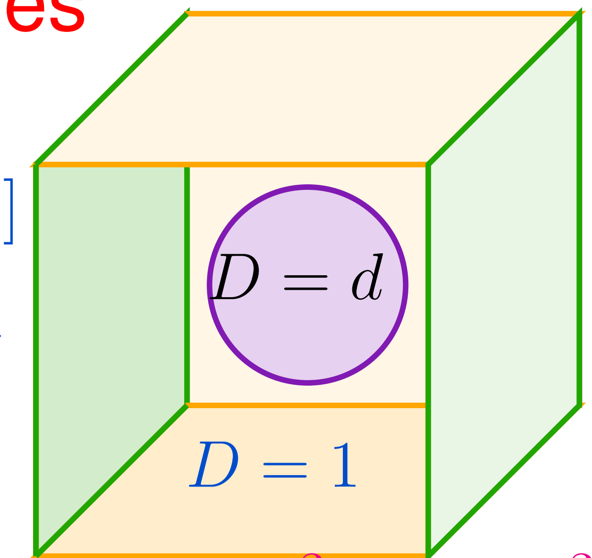
ULB

Sphere in a cube, Unstructured 3D meshes

Finite element discretization of

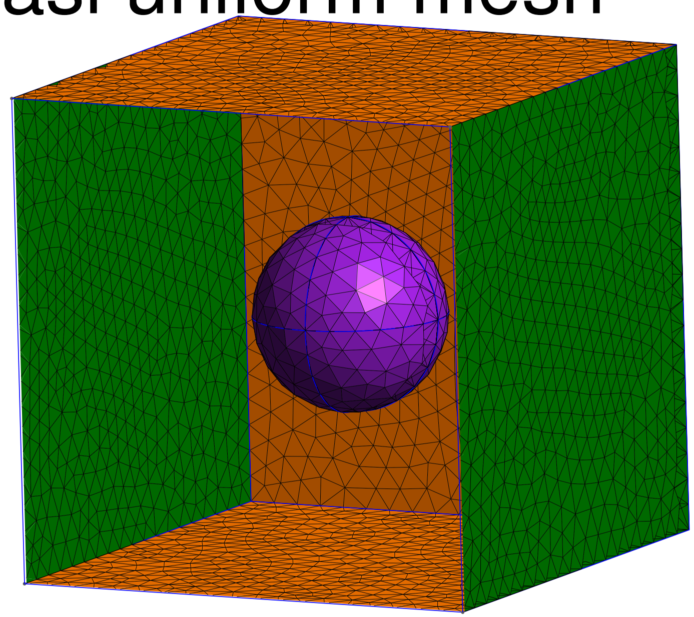
$$\begin{cases} -\nabla \cdot D \nabla u = 0 & \text{on } \Omega = [0,1] \times [0,1] \times [0,1] \\ u = 0 & \text{for } x = 0, 1\,,\ 0 \le y,\ z\ \le 1 \\ u = 1 & \text{elsewhere on } \partial\Omega \end{cases}$$

with $\nabla = \mathbf{1}_x \frac{\partial}{\partial x} + \mathbf{1}_y \frac{\partial}{\partial y} + \mathbf{1}_z \frac{\partial}{\partial z}$

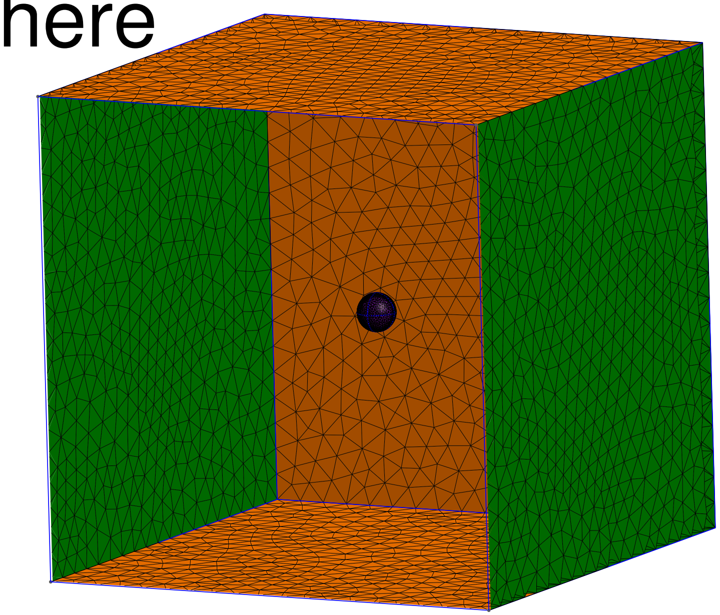


$d = 10^{-3}\,,\ 1\,,\ 10^3$

SPHUNF : quasi uniform mesh



SPHRF : mesh $10\,\times$finer on the sphere



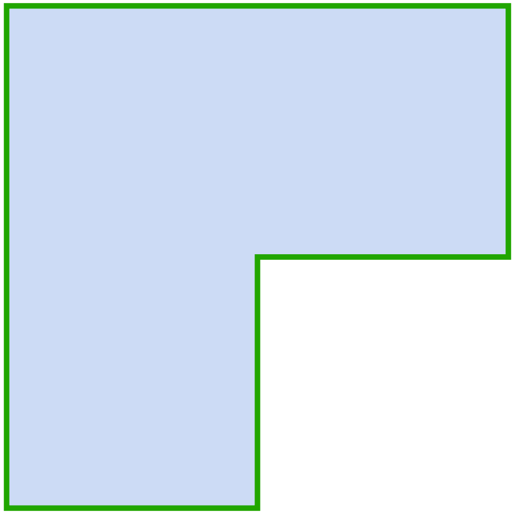
4. AGMG: Test suite (6)

(p. 24)

Reentering corner, local refinement

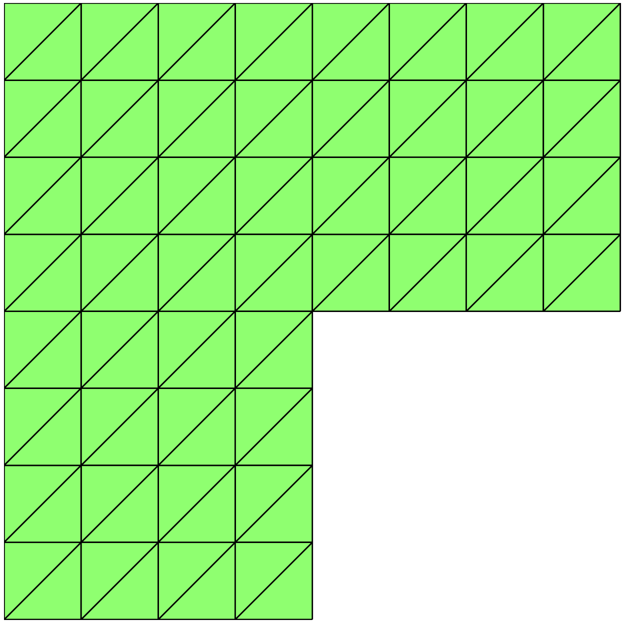
Finite element discretization of

$$\begin{cases} -\Delta u = 0 & \text{on } \Omega = [0,1]\times[0,1] \\ u = r^{\frac{2}{3}} \sin(\frac{2\theta}{3}) & \text{on } \partial\Omega \end{cases}$$



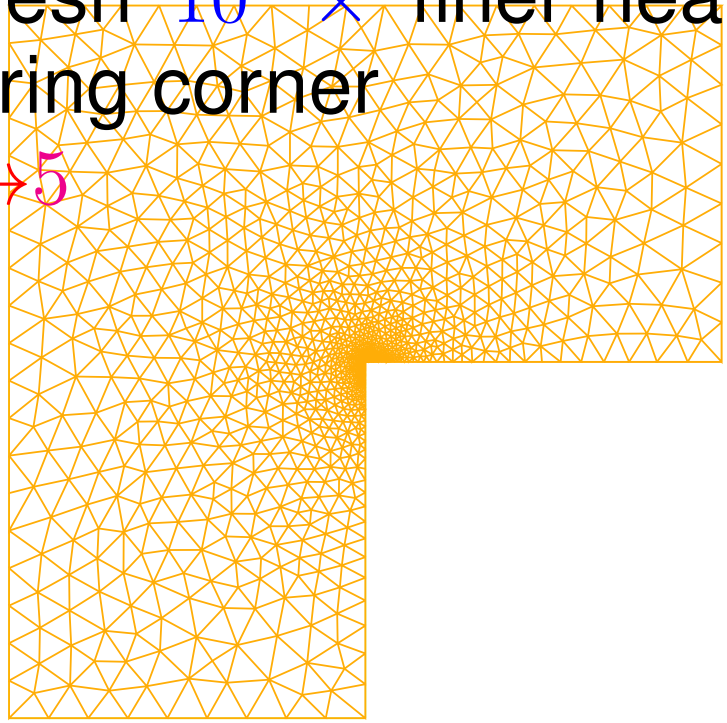
with $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

LUnfSt : uniform structured grid



LRfUSt : unstructured grid with mesh $10^{\,r}\times$ finer near reentering corner

$r = 0 \rightarrow 5$



ULB

Challenging convection-diffusion problems

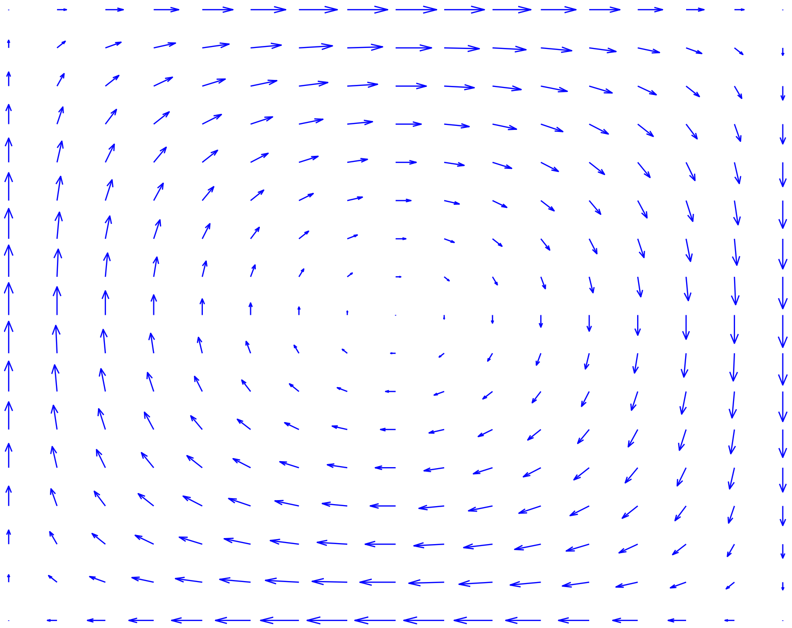
Upwind FD approximation of

$$\begin{cases} -\nu\,\Delta\,u \;+\; \overline{v}\cdot\textbf{grad}(u) \;=\; f & \text{in } \Omega \\ u \;=\; g & \text{on } \partial\Omega \end{cases}$$

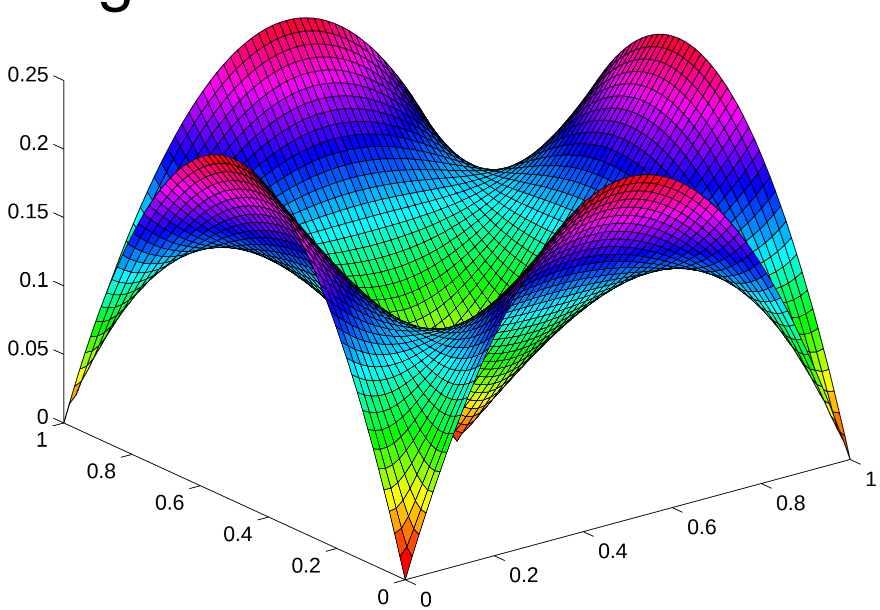
In all cases: tests for $\nu = 1\,,\ 10^{-2}\,,\ 10^{-4}\,,\ 10^{-6}$

$\nu \ll 1 \quad \rightarrow$ highly nonsymmetric matrices

Example of flow



Magnitude:



4. AGMG: Robustness study (1)

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
- ♦ FD on regular grids; 3 sizes:

  2D: $h^{-1} = 600\,,\ 1600\,,\ 5000$

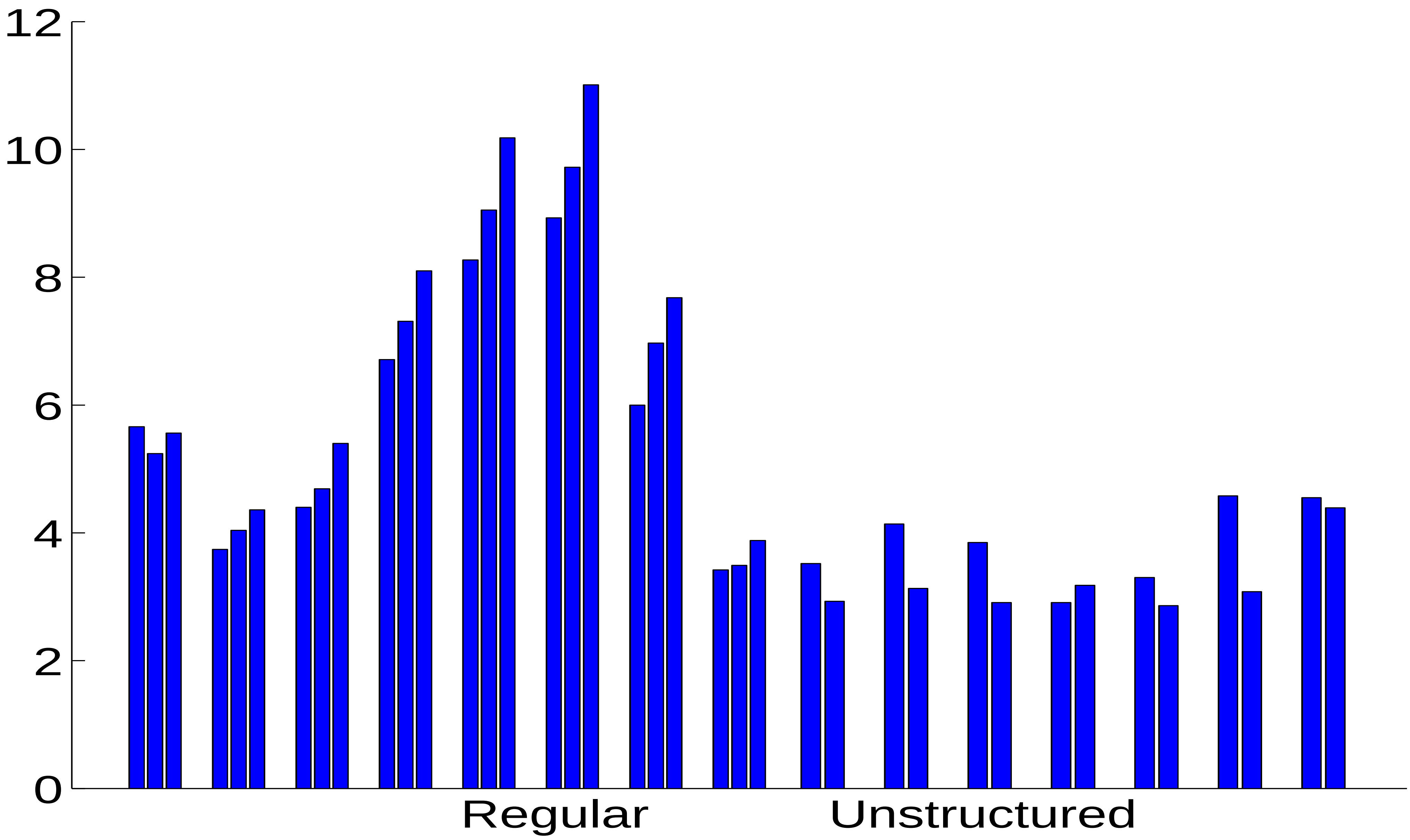
  3D: $h^{-1} = \ 80\,,\ \ 160\,,\ \ 320$

  ♦ FE on (un)structured meshes
  (with different levels of local refinement);
  2 sizes per problem: $n = 0.15e6 \rightarrow n = 7.1e6$

4. AGMG: Robustness study (2)

ULB

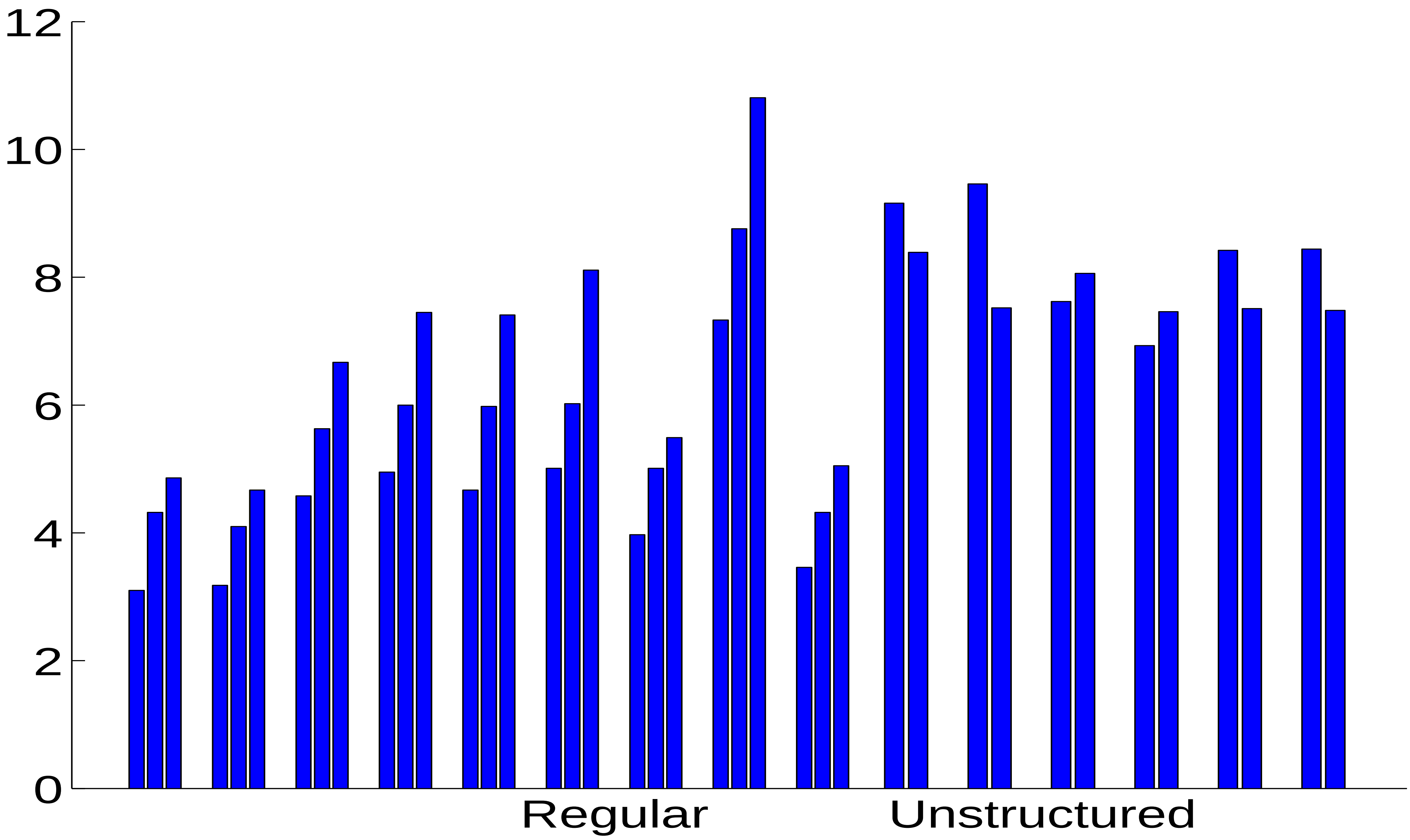
2D symmetric problems



4. AGMG: Robustness study (3)

ULB

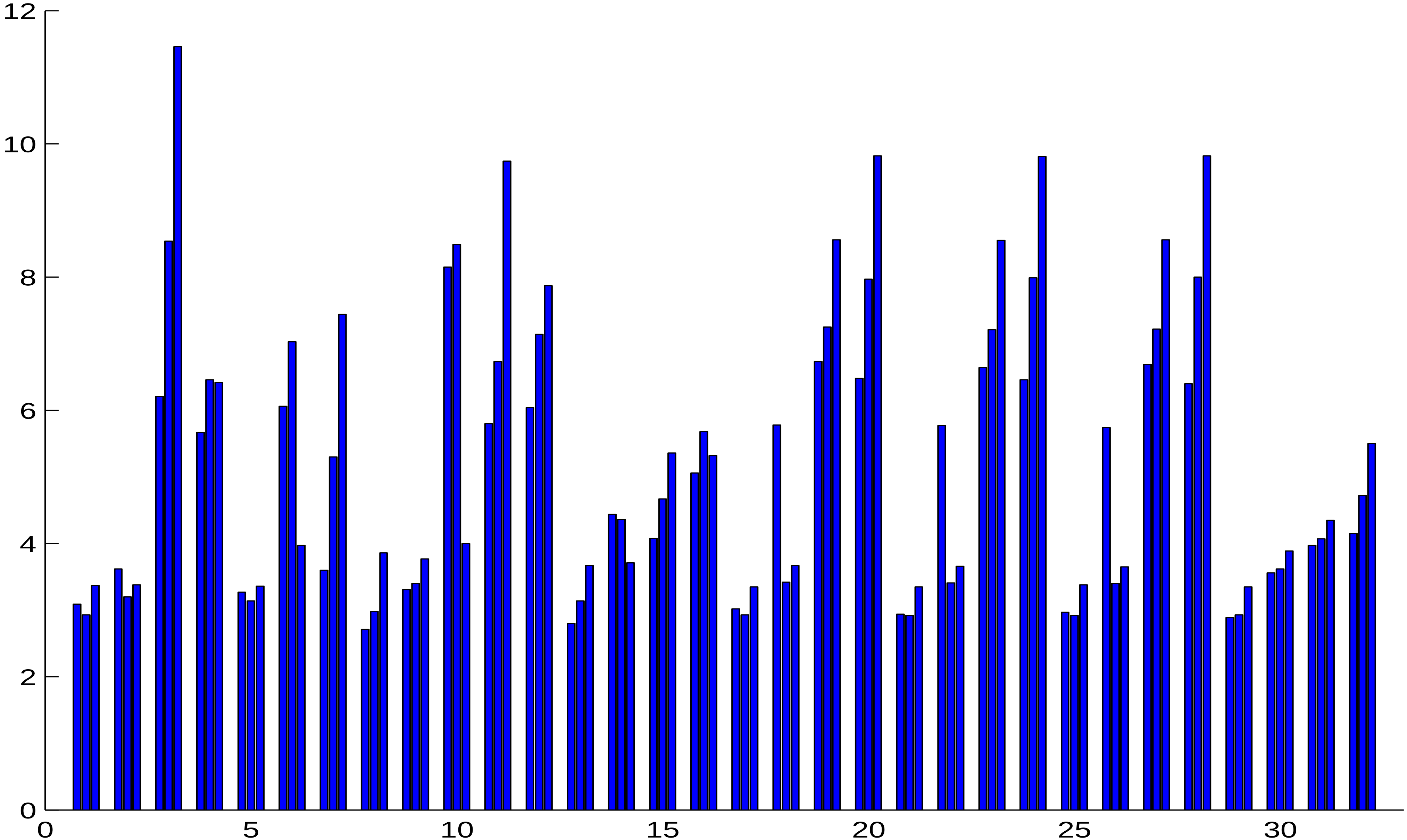
3D symmetric problems



4. AGMG: Robustness study (4)

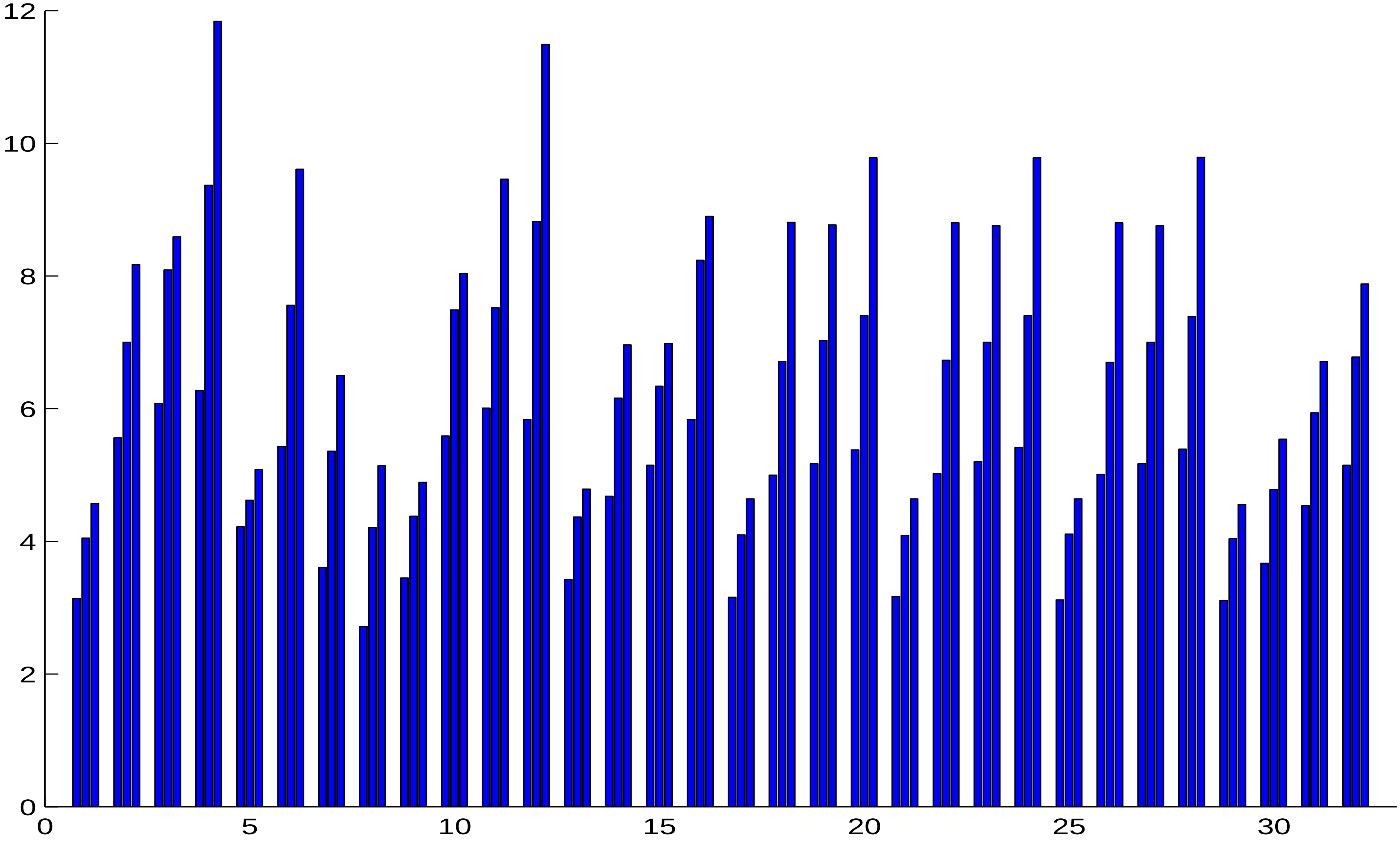
ULB

2D nonsymmetric problems



4. AGMG: Robustness study (5)

3D nonsymmetric problems



4. AGMG: comparative study (1)

ULB

Comparison with some other software

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

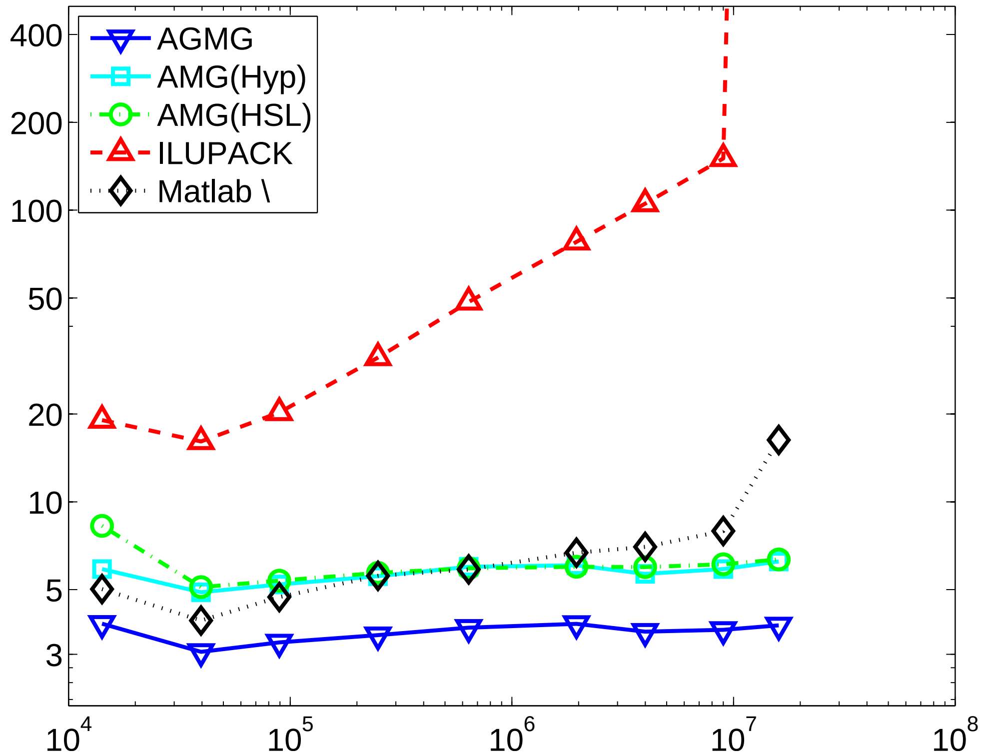
Quantity reported:

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)

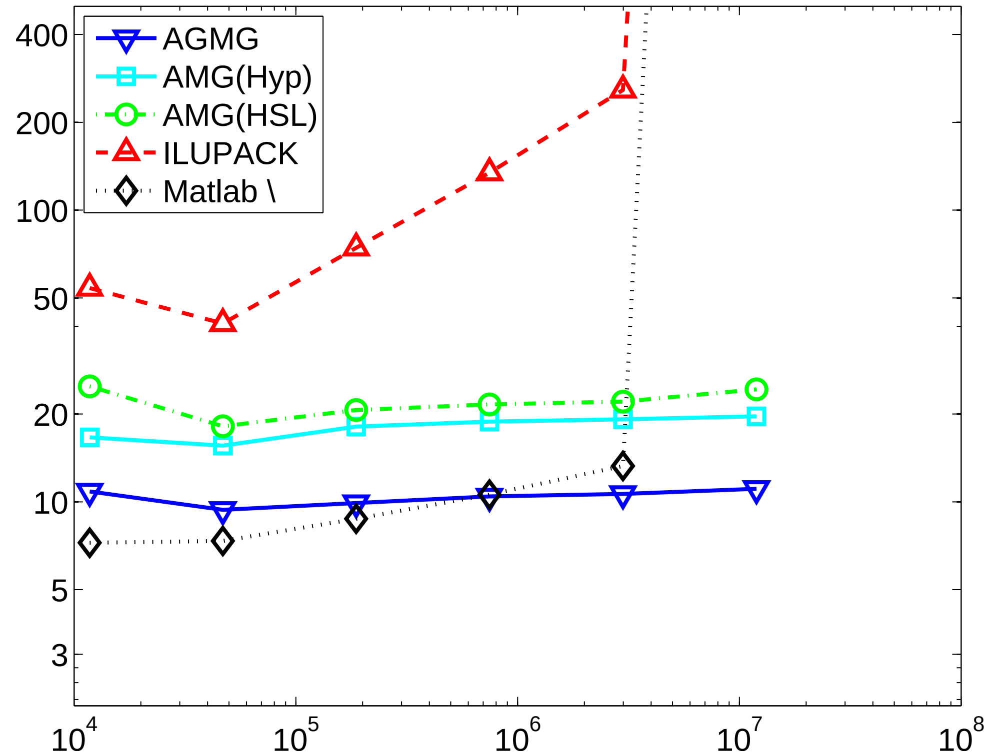
4. AGMG: comparative study (2)

(p. 32)

Poisson 2D, FD



Laplace 2D, FE(P3)

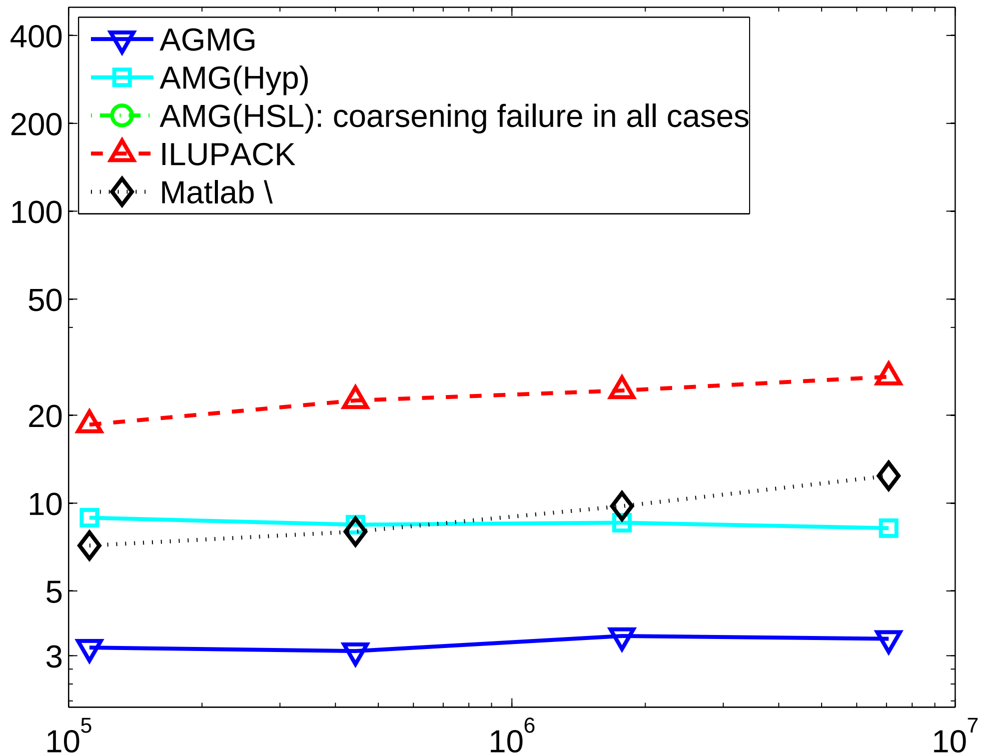


33% of nonzero offdiag > 0

4. AGMG: comparative study (3)

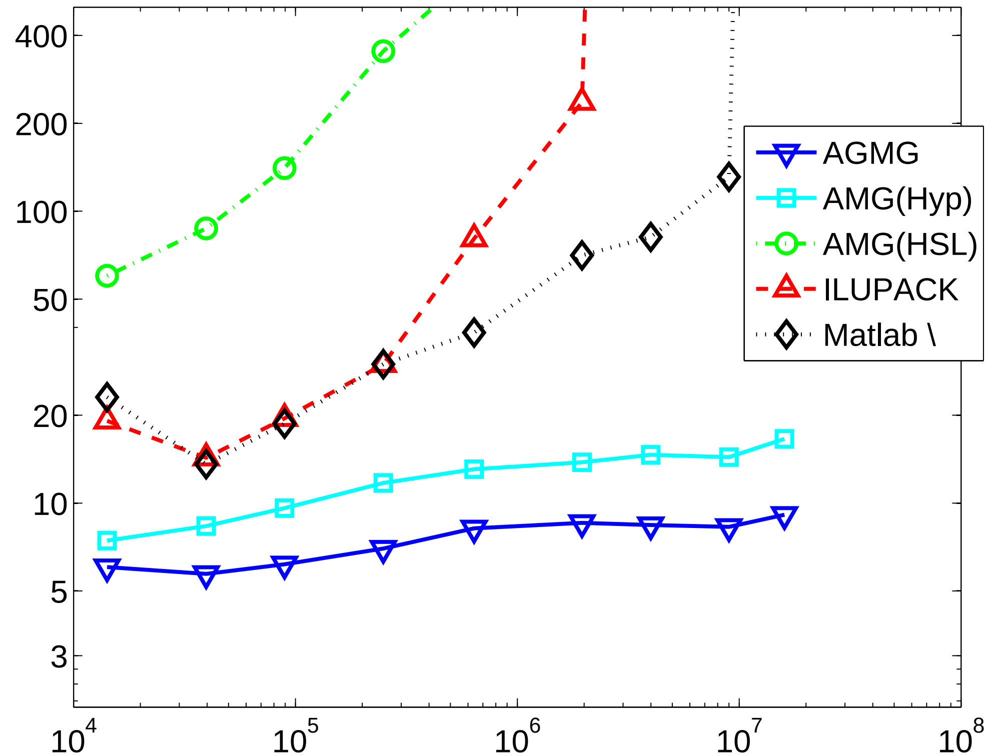
ULB

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



Convection-Diffusion 2D, FD

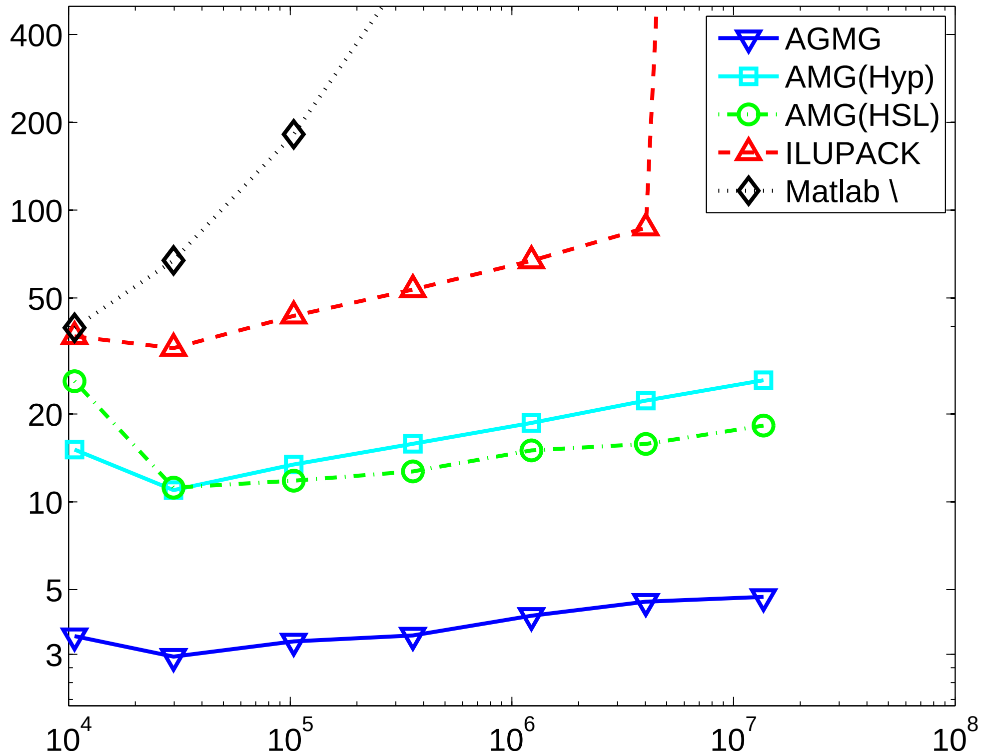
$\nu = 10^{-6}$



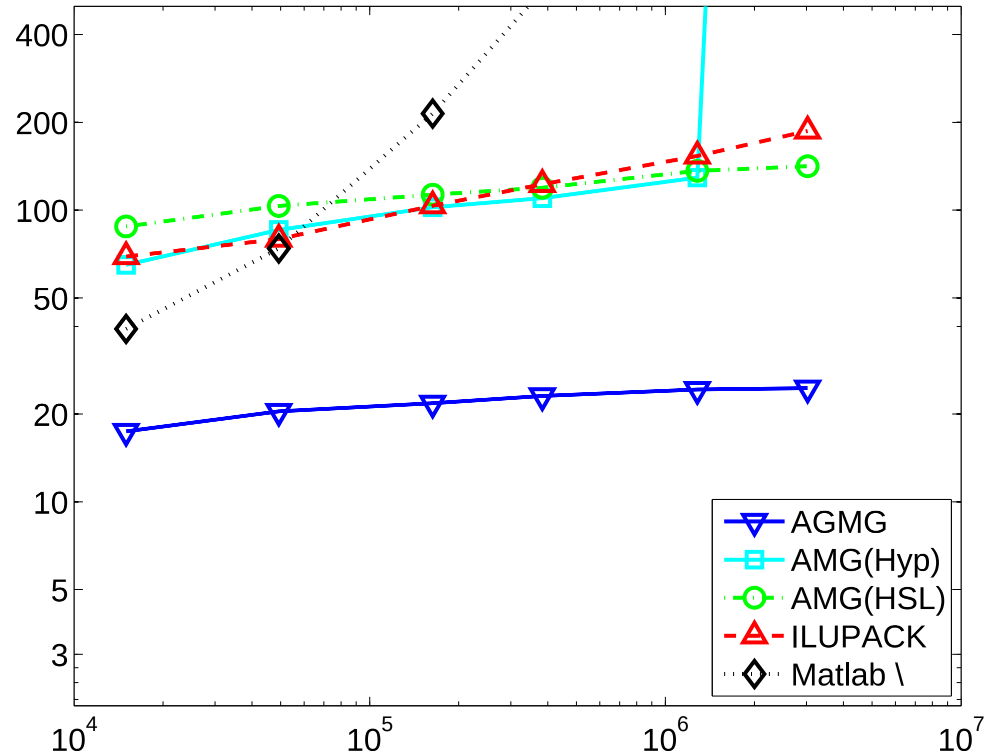
4. AGMG: comparative study (4)

ULB

Poisson 3D, FD



Laplace 3D, FE(P3)



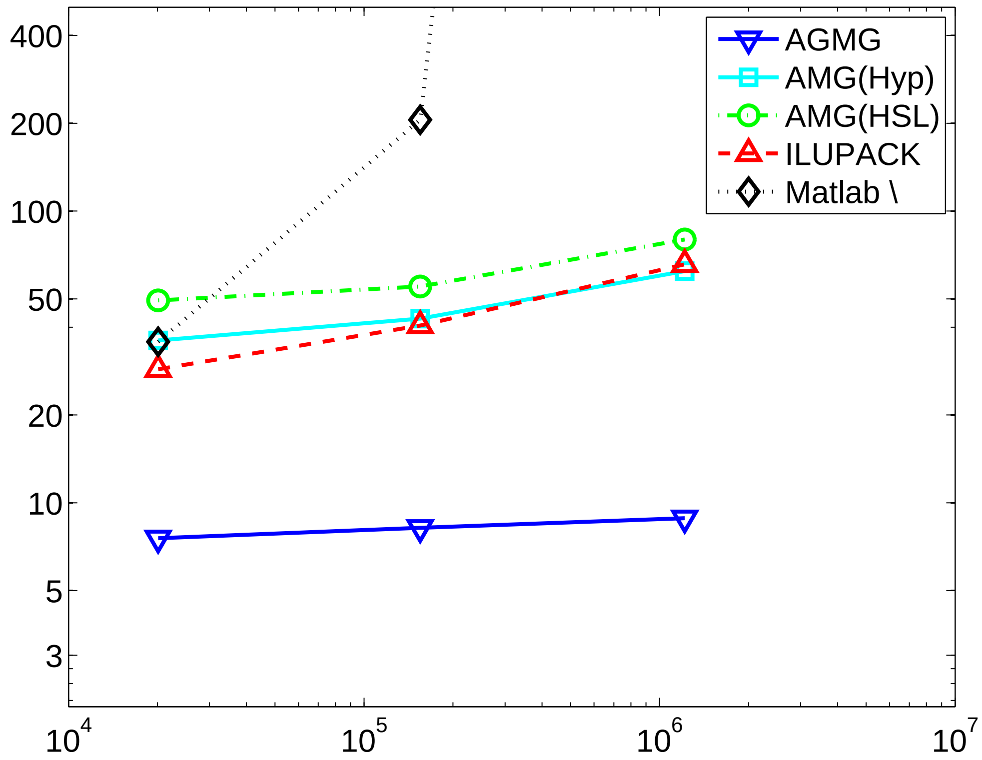
51% of nonzero offdiag > 0

4. AGMG: comparative study (5)

(p. 35)

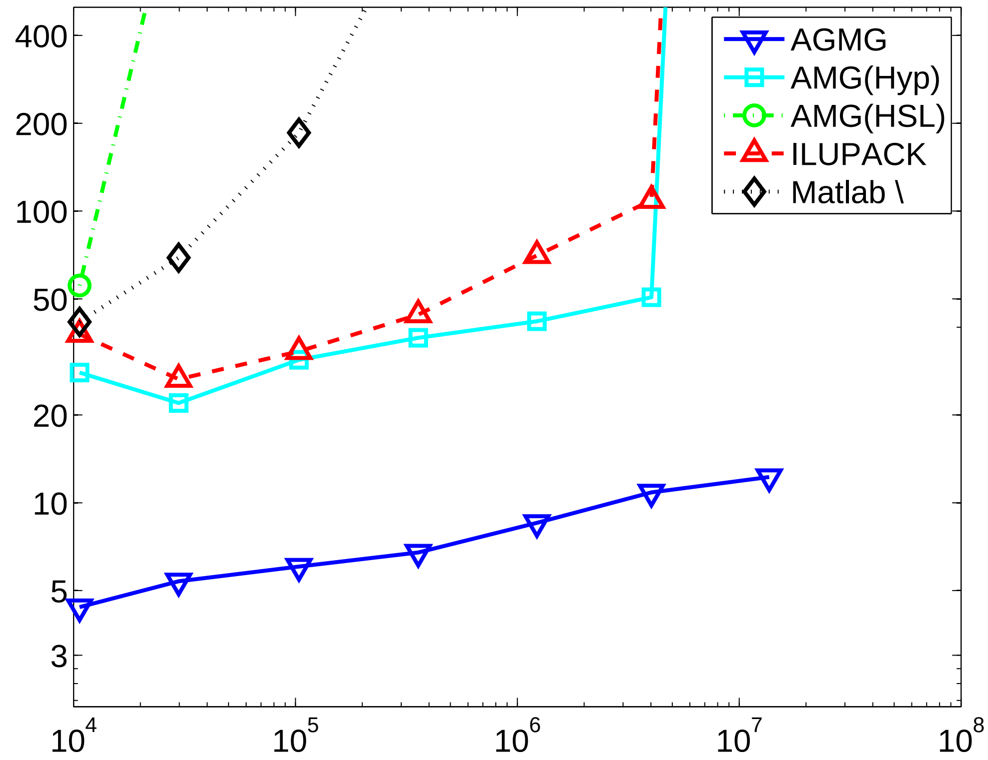
ULB

Poisson 3D, FE
Unstructured, Local refin.



Convection-Diffusion 3D, FD

$\nu = 10^{-6}$



5. Parallelization of AGMG (1)

(p. 36)

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?

5. 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

5. Parallelization of AGMG (2)

ULB

General parallelization strategy

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

5. Parallelization of AGMG (2)

ULB

General parallelization strategy

- Partitioning of the unknowns
  $\longrightarrow$ partitioning of matrix rows
- Aggregation algorithm
  Unchanged, except that aggregates are only formed with unknowns in a same partition.
  $\longrightarrow$ inherently parallel

5. 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
- Solve phase
  The parallelization raises no particular difficulties, except regarding the bottom level solver

  In sequential: sparse direct solver

  Parallel direct solver $\rightarrow$ bottleneck for many proc.

5. Parallelization of AGMG (2)

ULB

Algorithm redesign

- Only four levels, whatever problem size

5. Parallelization of AGMG (2)

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

5. Parallelization of AGMG (2)

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

5. Parallelization of AGMG (2)

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

5. Parallelization of AGMG (2)

(p. 38)

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)

5. Parallelization of AGMG (3)

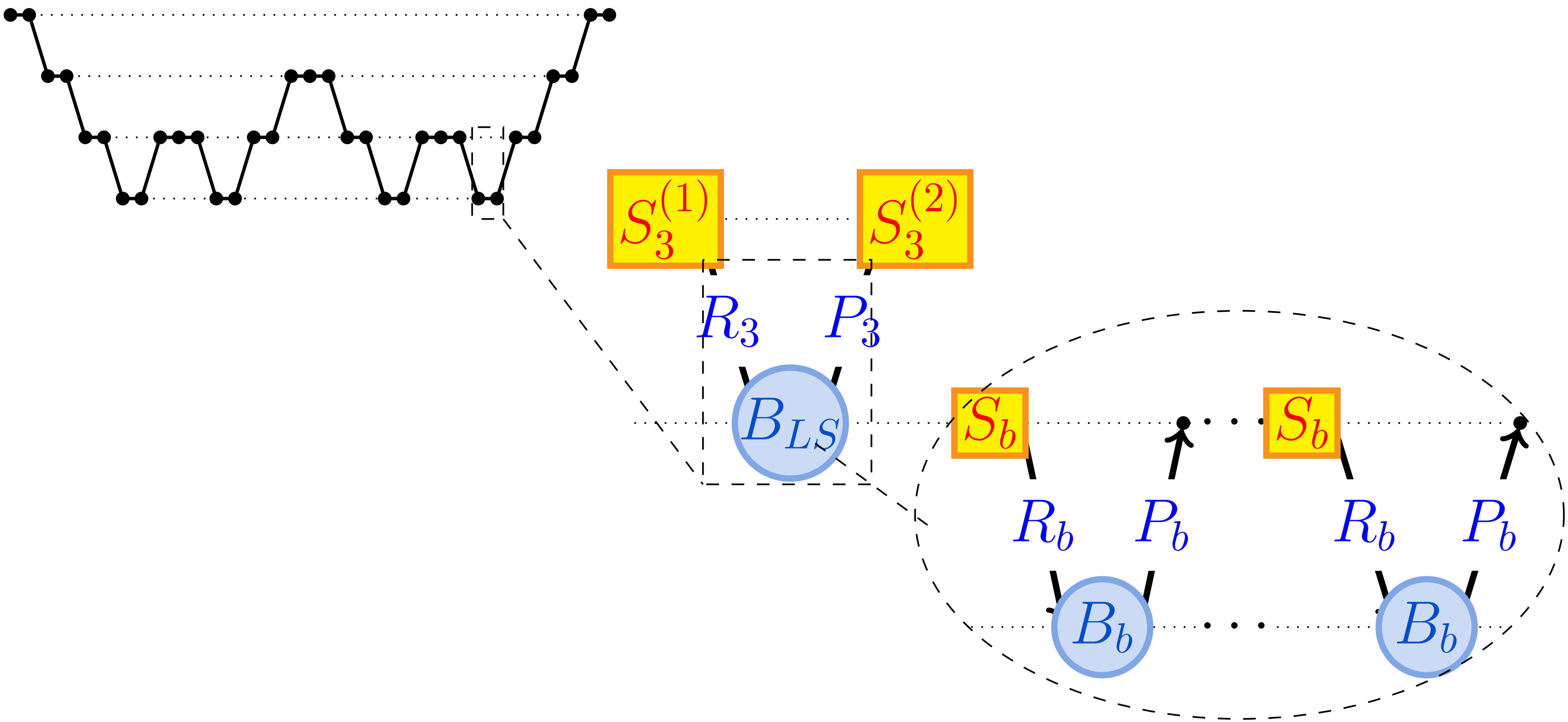
ULB

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)

5. Parallelization of AGMG (4)

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)

5. Parallelization of AGMG (5)

(p. 41)

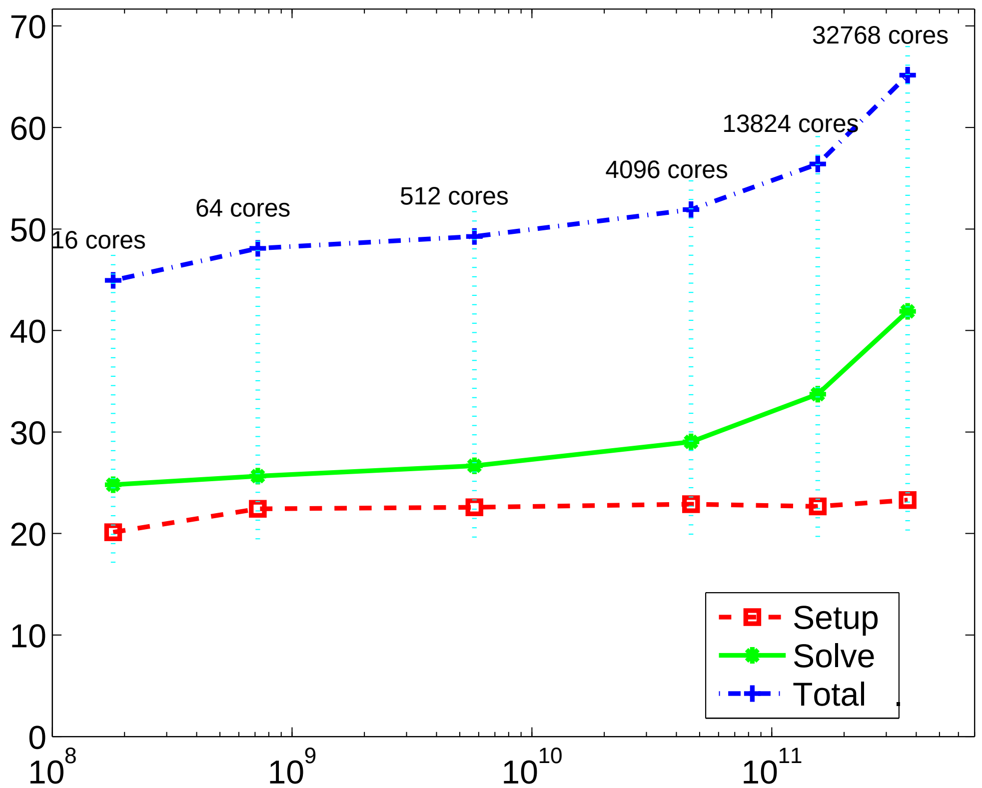
ULB

Results: the magic works

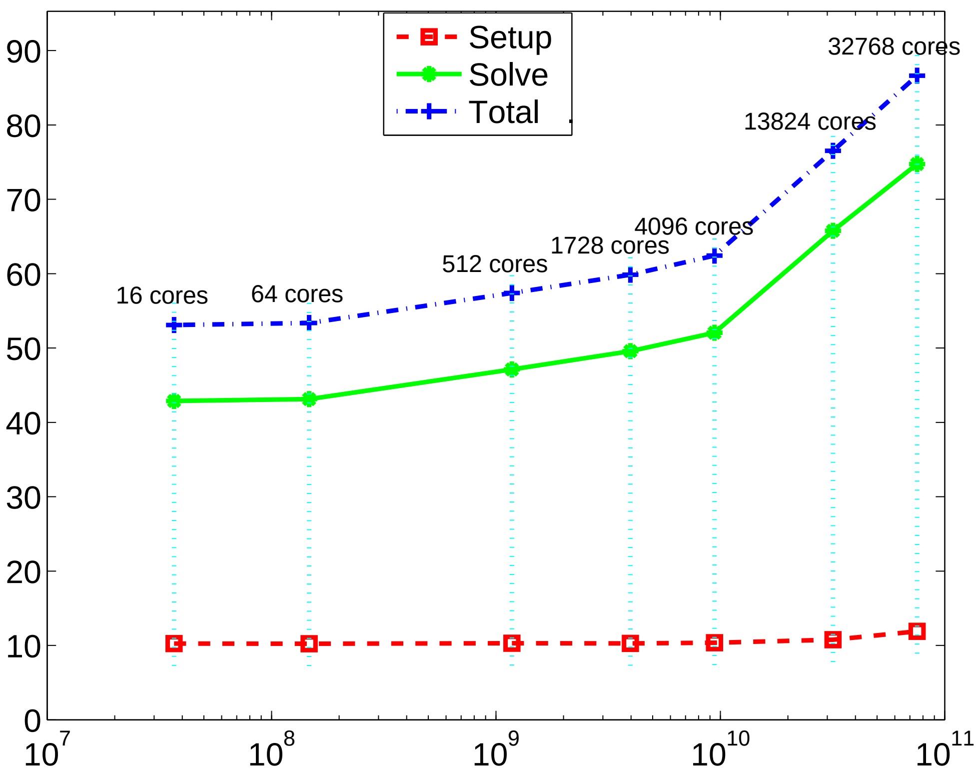
Weak scalability on CURIE (Intel Farm) for 3D Poisson

Elapsed time (seconds) – vs – number of unknowns

Finite Difference



P3 Finite Elements



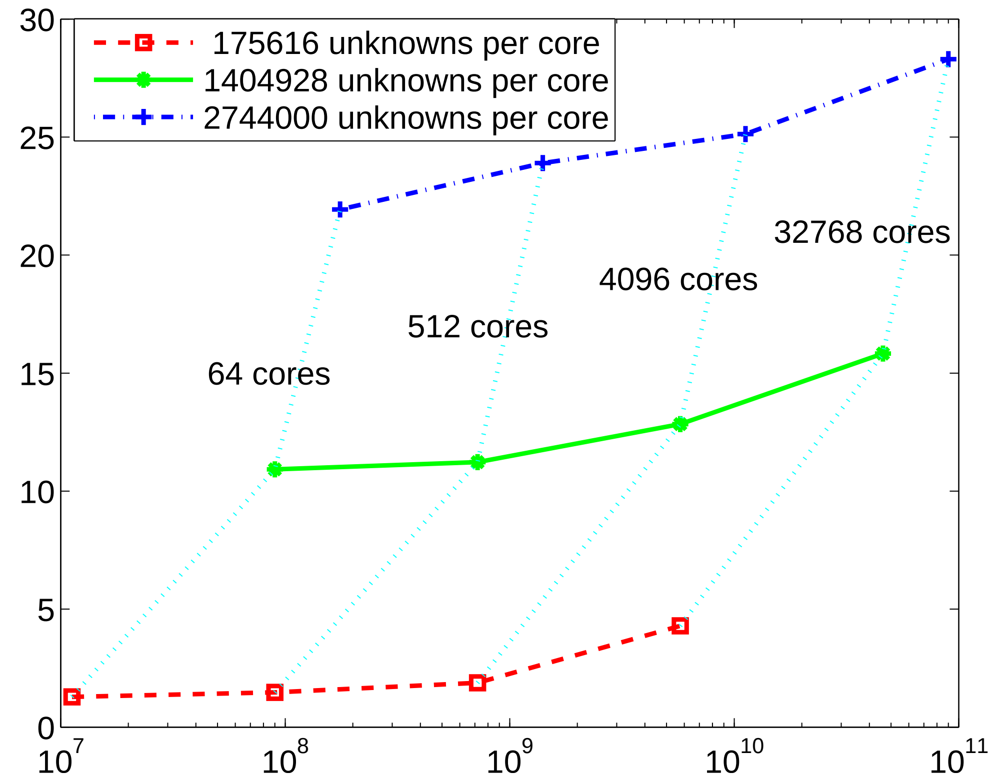
5. Parallelization of AGMG (6)

ULB

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

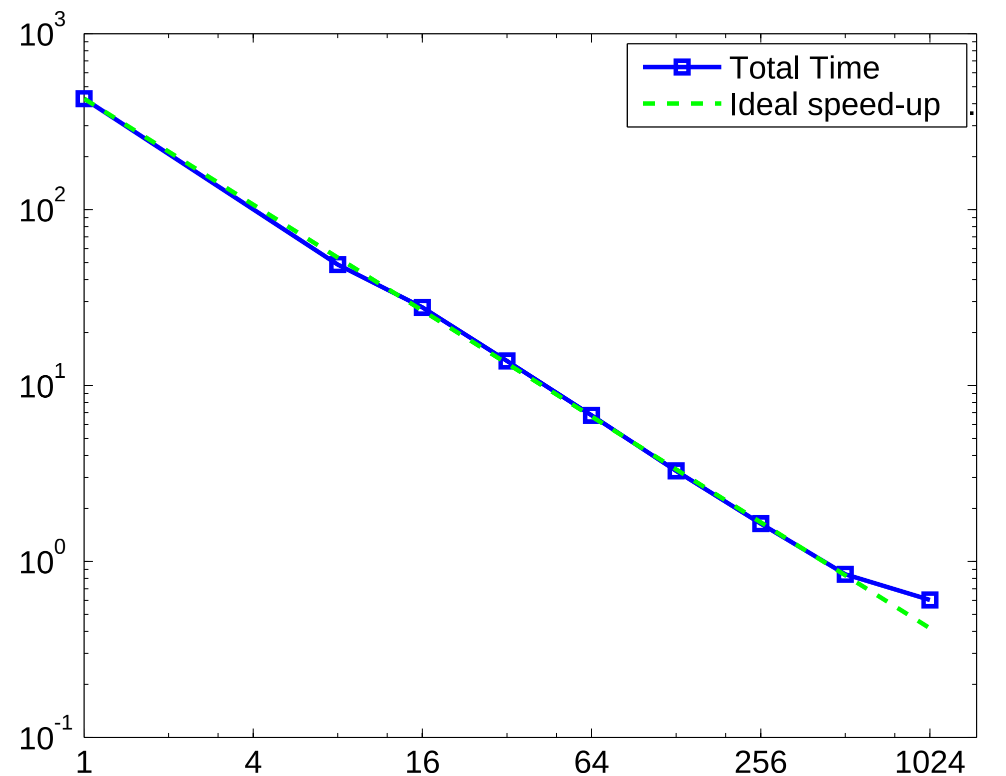
Weak scalability

Time – vs – # unknowns



Strong scalability

Time – vs – number of cores

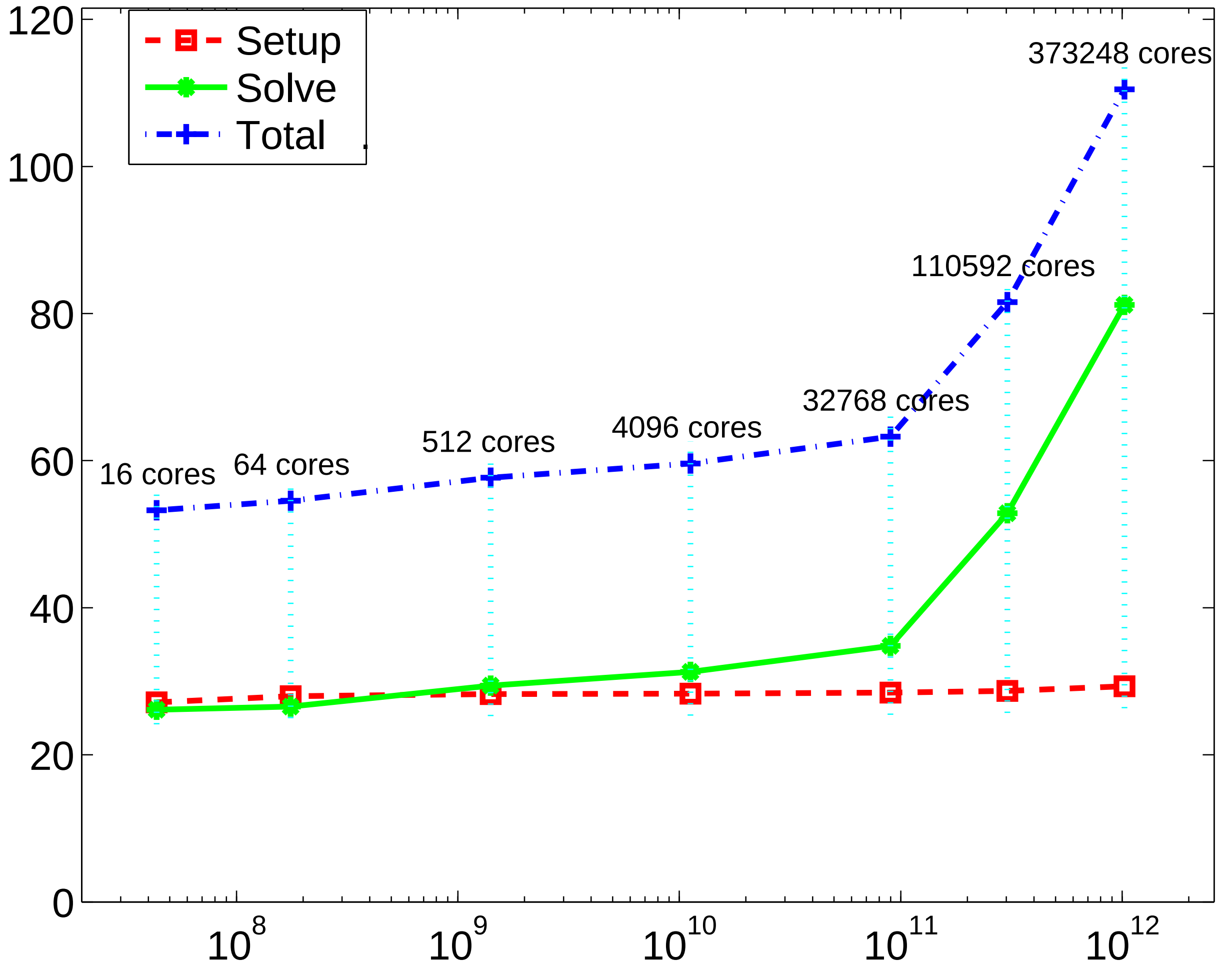


5. Parallelization of AGMG (7)

(p. 43)

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

Elapsed time – vs – number of unknowns



6. Conclusions

ULB

- Robust method for discrete Poisson-like problems

6. Conclusions

- Robust method for discrete Poisson-like problems
- Can be used (and is used!) black box
  (does not require tuning or adaptation)

6. Conclusions

- Robust method for discrete Poisson-like problems
- Can be used (and is used!) black box
  (does not require tuning or adaptation)
- Faster than other state-of-the-art solvers

6. Conclusions

ULB

- Robust method for discrete Poisson-like problems
- Can be used (and is used!) black box
  (does not require tuning or adaptation)
- Faster than other state-of-the-art solvers
- Fairly small setup time: especially well suited when only a modest accuracy is needed

6. Conclusions

ULB

- Robust method for discrete Poisson-like problems
- Can be used (and is used!) black box (does not require tuning or adaptation)
- Faster than other state-of-the-art solvers
- Fairly small setup time: especially well suited when only a modest accuracy is needed
- Efficient parallelization:
  Linear system with $> 10^{12}$ unknowns
  solved in less than 2 minutes
  using 373,248 cores on IBM BG/Q;
  that is: in about 0.1 nanoseconds per unknown

6. Conclusions

ULB

- Robust method for discrete Poisson-like problems
- Can be used (and is used!) black box
  (does not require tuning or adaptation)
- Faster than other state-of-the-art solvers
- Fairly small setup time: especially well suited when only a modest accuracy is needed
- Efficient parallelization:
  Linear system with $> 10^{12}$ unknowns
  solved in less than 2 minutes
  using 373,248 cores on IBM BG/Q;
  that is: in about 0.1 nanoseconds per unknown
- Professional code available, free academic license

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)

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)