Domain Decomposition Strategies for Adaptively Refined Meshes

Pavel Kůs, Jakub Šístek

Institute of Mathematics, Czech Academy of Sciences, Prague

Praha, February 11, 2020

Improvements of the finite element calculations

Many ways how to improve performance of finite element codes, two very different strategies are:

- parallel, domain decomposition, the use of supercomputers, very large linear systems
- adaptivity, higher order, the goal is to have smaller linear systems solvable on PC, still with good accuracy

It would be nice to combine both strategies

- Parallel calculations
  - take advantage from similarity of structure over the domain
  - most of the research done in linear solver part
  - Exascale brings many challenges. It could be good alternative to get more from petascale instead.
- Adaptivity
  - different treatment of different areas, based on solution behavior
  - a lot of effort in assembly part, trying to minimize number of DOFs
  - There are certain limits for single PC, no matter how smart the algorithm is.

Improvements of the finite element calculations

Many ways how to improve performance of finite element codes, two very different strategies are:

- parallel, domain decomposition, the use of supercomputers, very large linear systems
- adaptivity, higher order, the goal is to have smaller linear systems solvable on PC, still with good accuracy

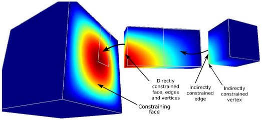
It would be nice to combine both strategies

- Parallel calculations
  - take advantage from similarity of structure over the domain
  - most of the research done in linear solver part
  - Exascale brings many challenges. It could be good alternative to get more from petascale instead.
- Adaptivity
  - different treatment of different areas, based on solution behavior
  - a lot of effort in assembly part, trying to minimize number of DOFs
  - There are certain limits for single PC, no matter how smart the algorithm is.

Ingredients

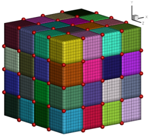
1. Experience with adaptivity and higher order finite elements

[P. K., P. Šolín, D. Andrš, *Arbitrary-level hanging nodes for adaptive hp-FEM approximations in 3D*, JCAM, 270, pp. 121–133, 2014]

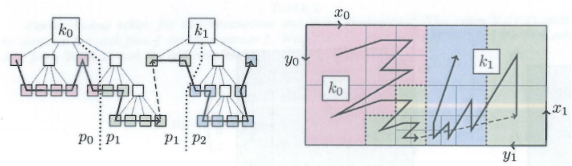


2. Experience with domain decomposition, BDDCML library

[B. Sousedík, J. Šístek, and J. Mandel, *Adaptive-Multilevel BDDC and its parallel implementation*, Computing, 95 (12), pp. 1087–1119, 2013.]

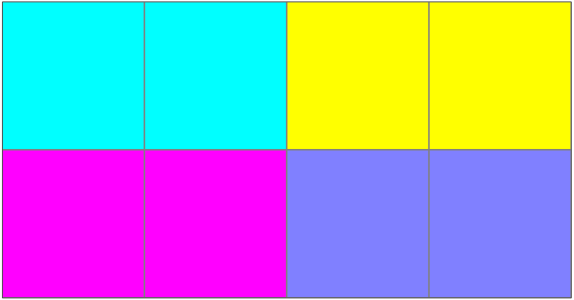


3. Parallel mesh handler p4est

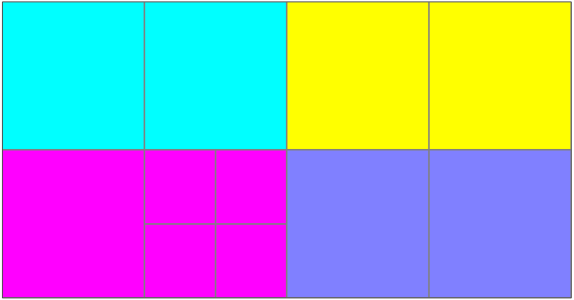


[C. Burstedde, L. Wilcox, and O. Ghattas, *p4est: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees*, SIAM J. Sci. Comput., 3 (33), pp. 1103–1133, 2011.]

Refinements without balancing

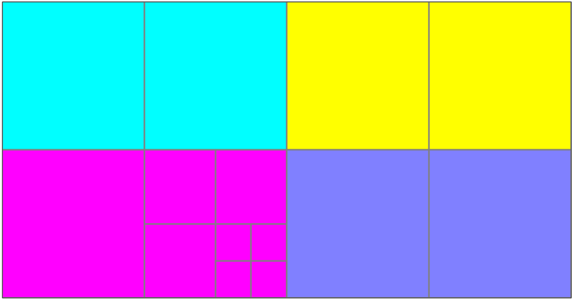


- Without balancing has no sense in parallel
- Most of the refinements would concentrate in those domains, where singularities, boundary or internal layers are present
- It might be just few subdomains



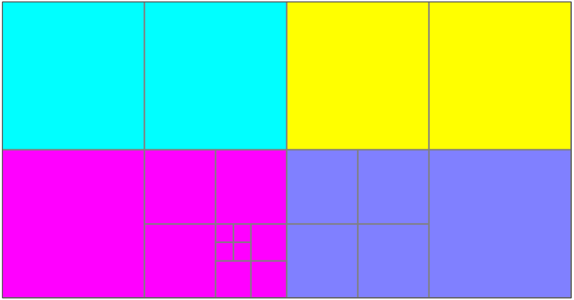
- Without balancing has no sense in parallel
- Most of the refinements would concentrate in those domains, where singularities, boundary or internal layers are present
- It might be just few subdomains

Refinements without balancing



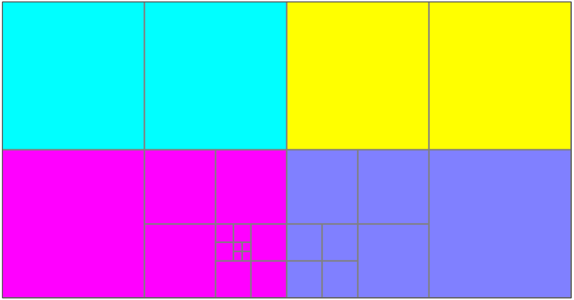
- Without balancing has no sense in parallel
- Most of the refinements would concentrate in those domains, where singularities, boundary or internal layers are present
- It might be just few subdomains

Refinements without balancing



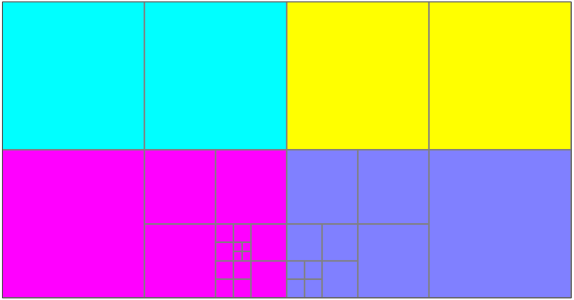
- Without balancing has no sense in parallel
- Most of the refinements would concentrate in those domains, where singularities, boundary or internal layers are present
- It might be just few subdomains

Refinements without balancing



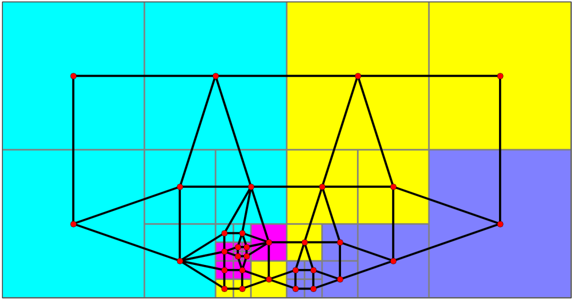
- Without balancing has no sense in parallel
- Most of the refinements would concentrate in those domains, where singularities, boundary or internal layers are present
- It might be just few subdomains

Refinements without balancing



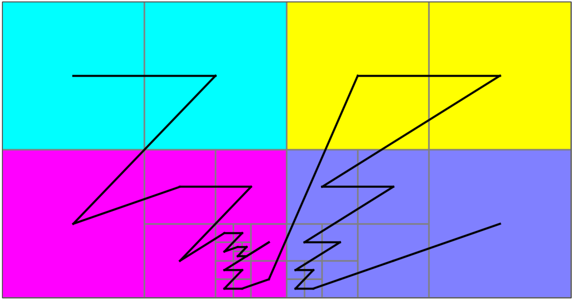
- Without balancing has no sense in parallel
- Most of the refinements would concentrate in those domains, where singularities, boundary or internal layers are present
- It might be just few subdomains.

Partitioning of graph of mesh



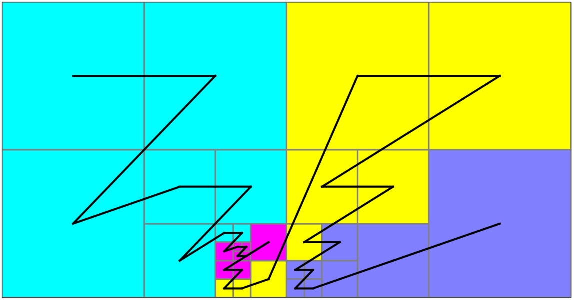
- Element incidence graph can be created and partitioned
- Advantage: usually nice shape of subdomains
- Disadvantage: it is not scalable to large number of processors

Partitioning of space-filling curve



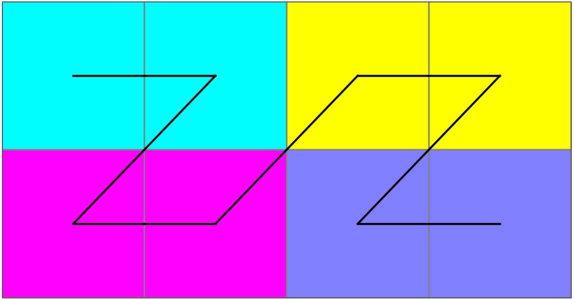
- Z-order (space filling) curve can be used
- Each element in the refinement hierarchy might be identified by number, coding bitwise its position and level of refinement
- Can be used both in 2D (quadrilaterals) or 3D (hexahedra)

Partitioning of space-filling curve

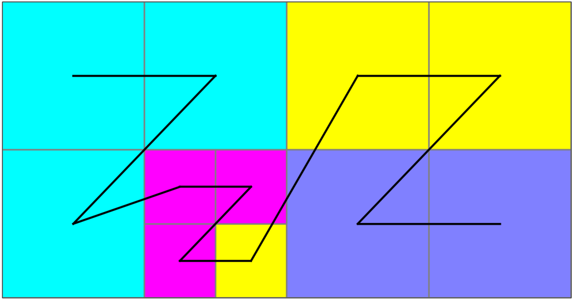


- Curve will be split equally among individual processors
- Each element in the refinement hierarchy might be identified by number, coding bitwise its position and level of refinement
- Can be used both in 2D (quadrilaterals) or 3D (hexahedra)

Partitioning of space-filling curve

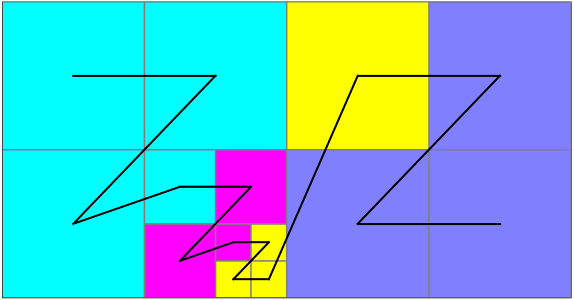


- Can be used for partitioning
- Initial mesh can be small, just to express the geometry. It has to be made of quadrilaterals or hexahedra, which might be limiting.
- Disadvantage: shape of the subdomains far from optimal



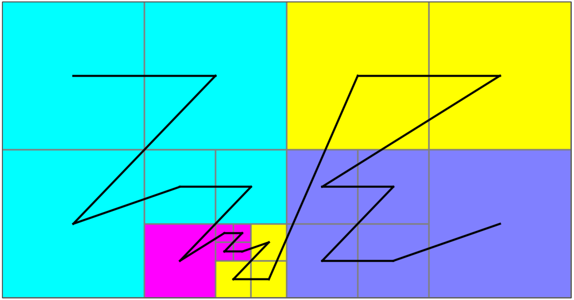
- Can be used for partitioning
- Initial mesh can be small, just to express the geometry. It has to be made of quadrilaterals or hexahedra, which might be limiting.
- Disadvantage: shape of the subdomains far from optimal

Partitioning of space-filling curve



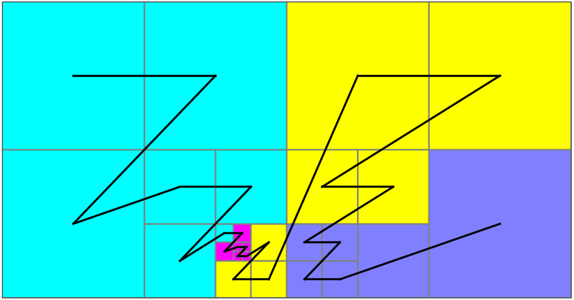
- Can be used for partitioning
- Initial mesh can be small, just to express the geometry. It has to be made of quadrilaterals or hexahedra, which might be limiting.
- Disadvantage: shape of the subdomains far from optimal

Partitioning of space-filling curve



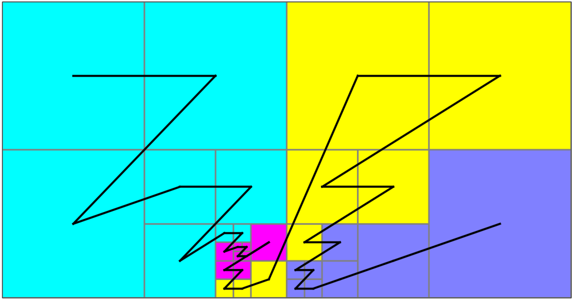
- Can be used for partitioning
- Initial mesh can be small, just to express the geometry. It has to be made of quadrilaterals or hexahedra, which might be limiting.
- Disadvantage: shape of the subdomains far from optimal

Partitioning of space-filling curve

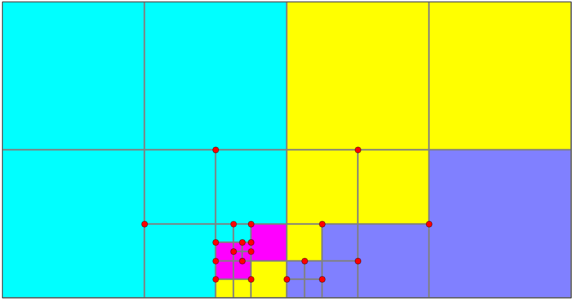


- Can be used for partitioning
- Initial mesh can be small, just to express the geometry. It has to be made of quadrilaterals or hexahedra, which might be limiting.
- Disadvantage: shape of the subdomains far from optimal

Partitioning of space-filling curve

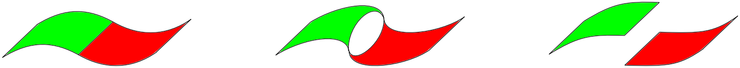


- Can be used for partitioning
- Initial mesh can be small, just to express the geometry. It has to be made of quadrilaterals or hexahedra, which might be limiting.
- Disadvantage: shape of the subdomains far from optimal.



1. **Hanging nodes**
   - Hanging nodes have to be eliminated
   - They can also appear at the subdomain interface
2. **Shape of the subdomains**
   - The shape is far from perfect
   - Subdomains might be disconnected or only loosely coupled (e.g. by one node in elasticity)

The BDDC preconditioner

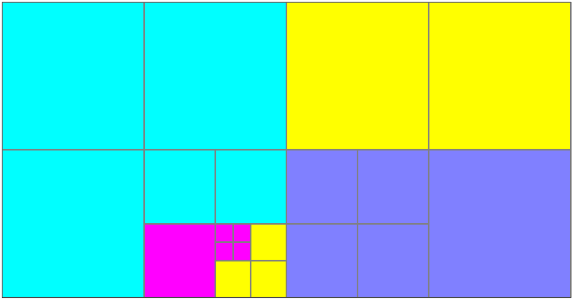


U	$\subset$	$\widetilde{W}$	$\subset$	W
continuous at all nodes at interface		continuous at selected coarse dofs		no continuity at interface

- Balancing Domain Decomposition based on Constraints [Dohrmann (2003)], [Cros (2003)], [Fragakis, Papadrakakis (2003)]
- continuity at *corners*, and of averages (arithmetic or weighted) over *edges* or *faces* considered
- enough constraints to **fix floating subdomains** — $a(\cdot,\cdot)$ symmetric positive definite on $\widetilde{W}$
- corresponding matrix $\widetilde{A}$ symmetric positive definite, almost block diagonal structure, larger dimension than A
- used to construct (an action of) preconditioner M^{-1} to solve

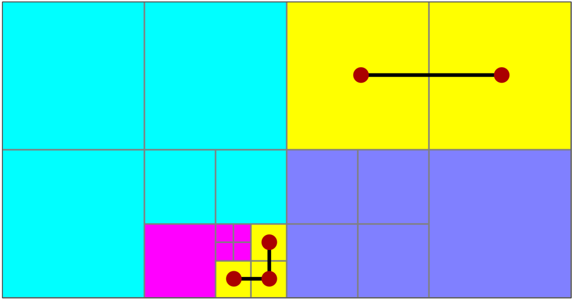
$$M^{-1} S u_{\Gamma} = M^{-1} g$$

Disconnected and loosely coupled subdomains



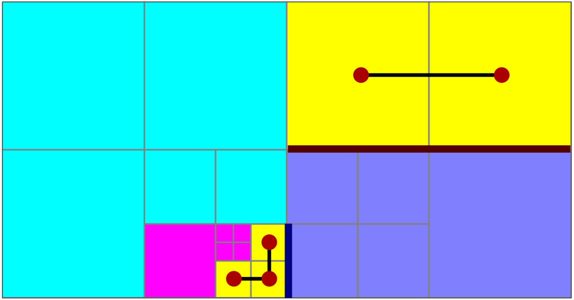
- nullspaces of subdomain matrices unknown a priori
- detect **graph components** of subdomain mesh
- **components independent** during classification of interface into faces, edges and vertices
- corners selected by the face-based algorithm [Šístek et al. (2011)]
- size of local problems unchanged, but larger nullspaces lead locally to more constraints — still potential load imbalance

Disconnected and loosely coupled subdomains



- nullspaces of subdomain matrices unknown a priori
- detect **graph components** of subdomain mesh
- **components independent** during classification of interface into faces, edges and vertices
- corners selected by the face-based algorithm [Šístek et al. (2011)]
- size of local problems unchanged, but larger nullspaces lead locally to more constraints — still potential load imbalance

Disconnected and loosely coupled subdomains



- nullspaces of subdomain matrices unknown a priori
- detect **graph components** of subdomain mesh
- **components independent** during classification of interface into faces, edges and vertices
- corners selected by the face-based algorithm [Šístek et al. (2011)]
- size of local problems unchanged, but larger nullspaces lead locally to more constraints — still potential load imbalance

Parallel implementation

Parallel FEM solver with AMR

- experimental in-house code
- high order finite elements
- Poisson equation and linear elasticity
- C++ (object oriented) + MPI

p4est mesh manager for AMR

- rebalancing based on Z-curves
- ANSI C + MPI
- open-source (GPL)
- scalability reported for 1e5–1e6 cores

http://www.p4est.org

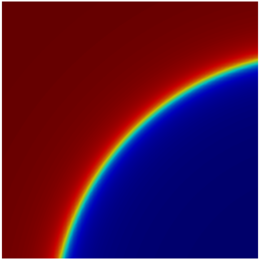
BDDCML equation solver

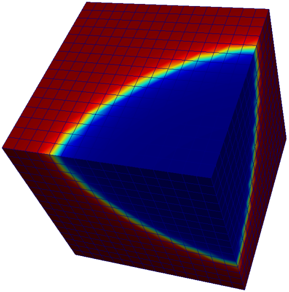
- Adaptive-Multilevel BDDC
- Fortran 95 + MPI
- open-source (LGPL)
- current version 2.5 (8/6/'15)
- tested on up to 65e3 cores and 2e9 unknowns

http://www.math.cas.cz/~sistek/software/bddcml.html

Internal layer benchmark

$$-\triangle u = f \quad \text{on} \quad (0,1)^d$$
$$u = \arctan\Big(s \cdot (r - \frac{\pi}{3})\Big)$$

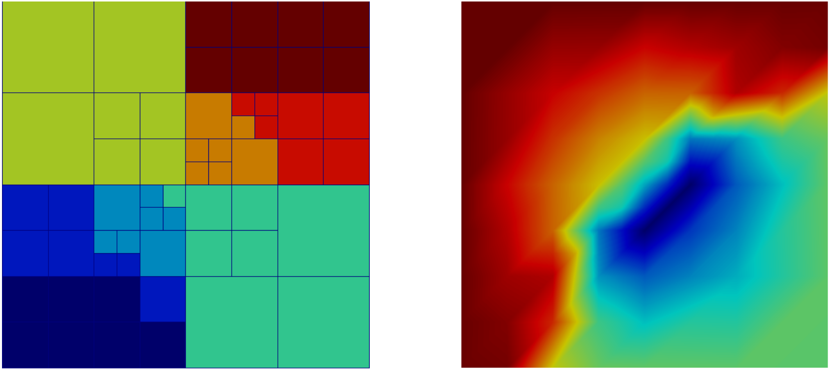




- solution exhibits sharp internal layer
- r is a distance from a given point
- s controls “steepness” of the layer

Adaptivity in 2D on 8 subdomains

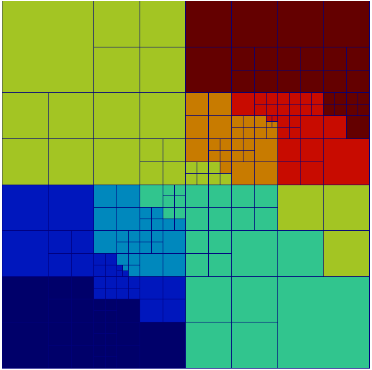
- Adaptivity tested for element orders 1-4 (showed order 1)
- Guided by exact solution, using H^1 semi-norm for error calculation

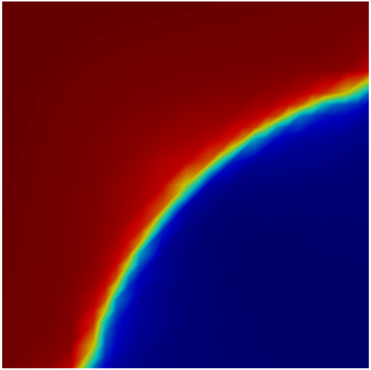


Iteration 3, mesh and solution

Adaptivity in 2D on 8 subdomains

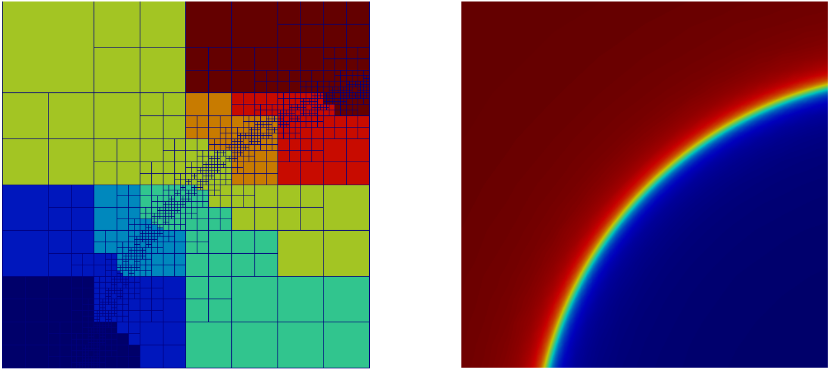
- Adaptivity tested for element orders 1-4 (showed order 1)
- Guided by exact solution, using H^1 semi-norm for error calculation





Iteration 5, mesh and solution

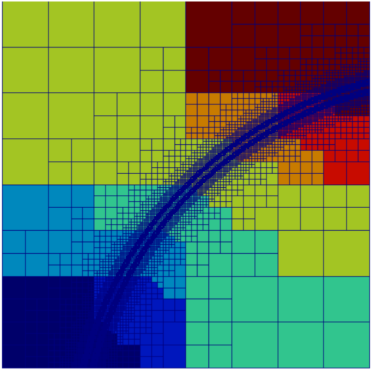
- Adaptivity tested for element orders 1-4 (showed order 1)
- Guided by exact solution, using H^1 semi-norm for error calculation

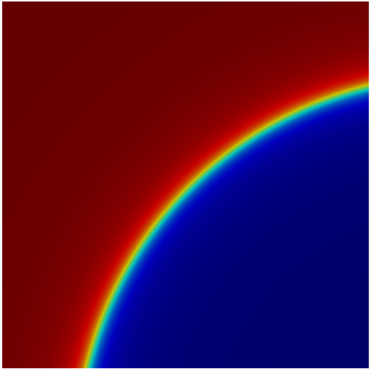


Iteration 8, mesh and solution

Adaptivity in 2D on 8 subdomains

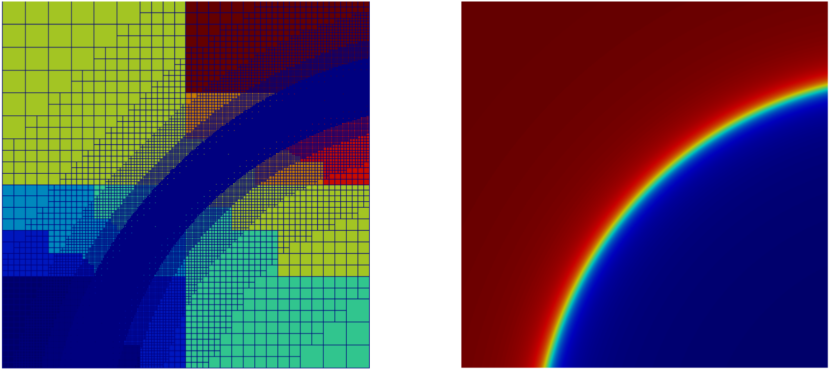
- Adaptivity tested for element orders 1-4 (showed order 1)
- Guided by exact solution, using H^1 semi-norm for error calculation





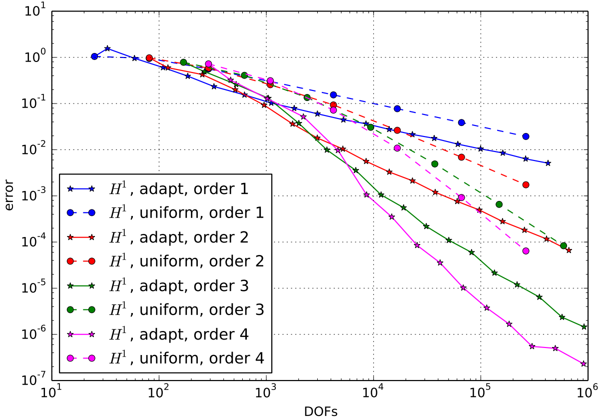
Iteration 13, mesh and solution

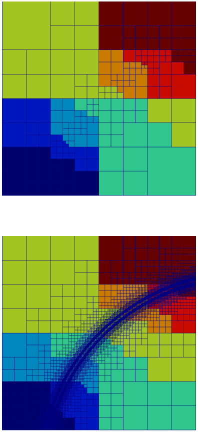
- Adaptivity tested for element orders 1-4 (showed order 1)
- Guided by exact solution, using H^1 semi-norm for error calculation



Iteration 18, mesh and solution.

Adaptivity in 2D on 8 subdomains

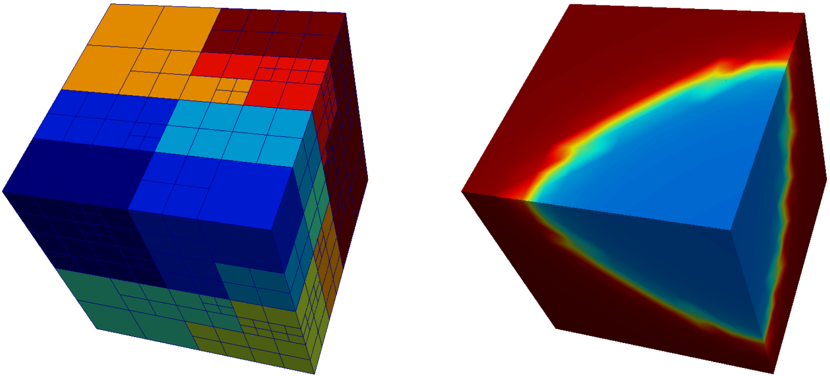




Convergence of adaptivity in 2D, 8 subdomains

Adaptivity in 3D on 8 subdomains

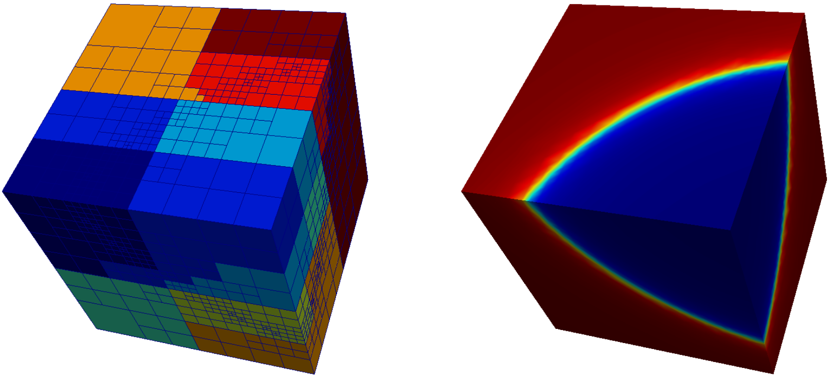
- Adaptivity tested for element orders 1-4 (showed order 1)
- Guided by exact solution, using H^1 semi-norm for error calculation



Iteration 3, mesh and solution

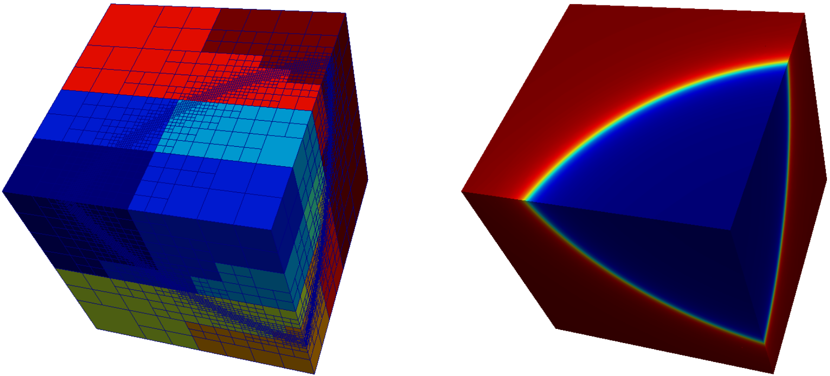
Adaptivity in 3D on 8 subdomains

- Adaptivity tested for element orders 1-4 (showed order 1)
- Guided by exact solution, using H^1 semi-norm for error calculation



Iteration 5, mesh and solution

- Adaptivity tested for element orders 1-4 (showed order 1)
- Guided by exact solution, using H^1 semi-norm for error calculation



Iteration 8, mesh and solution

Conclusions

Development of parallel FEM with AMR

- level-1 hanging nodes simple to handle and interface with DD solver
- disconnected and loosely coupled subdomains handled by detecting components of subdomain mesh

[P. K., J. Šístek, *Coupling parallel adaptive mesh refinement with a nonoverlapping domain decomposition solver*, Advances in Engineering Software 110, 34–54, 2017]

Future work

- improve the performance of the preconditioner by better subdomain shapes; Hilbert curves
- currently each refinement changes all subdomains – not optimal
- multiple subdomains per compute node
- try to keep most of the subdomains intact and re-use part of the preconditioner work
- distribute created or changed subdomains to ensure load balancing
- promising starting point for *embedded domain FEM*

Conclusions

Development of parallel FEM with AMR

- level-1 hanging nodes simple to handle and interface with DD solver
- disconnected and loosely coupled subdomains handled by detecting components of subdomain mesh

[P. K., J. Šístek, *Coupling parallel adaptive mesh refinement with a nonoverlapping domain decomposition solver*, Advances in Engineering Software 110, 34–54, 2017]

Future work

- improve the performance of the preconditioner by better subdomain shapes; Hilbert curves
- currently each refinement changes all subdomains – not optimal
- multiple subdomains per compute node
- try to keep most of the subdomains intact and re-use part of the preconditioner work
- distribute created or changed subdomains to ensure load balancing
- promising starting point for *embedded domain FEM*

Thank you for your attention.

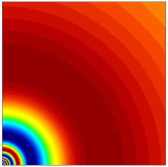
Pavel Kůs
kus@math.cas.cz

Jakub Šístek
sistek@math.cas.cz

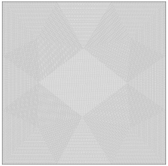
Institute of Mathematics, Czech Academy of Sciences, Prague

Mesh refinements – extra

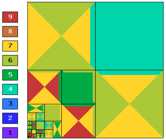
- We want to focus the computational effort to troubled areas (e.g. singularities of electrostatic field, boundary layers in flow simulations, etc.)
- Algorithmic complexity of the software grows
- Different approaches
  - complete re-meshing
  - change of vertices positions(r-adaptivity)
  - element refinements (h-adaptivity)
  - different polynomial orders (p-adaptivity)
  - combination of both (hp-adaptivity)



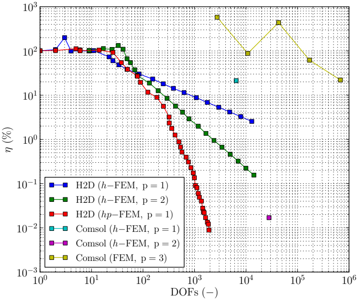
benchmark



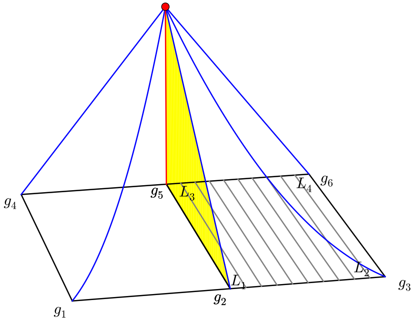
uniform mesh

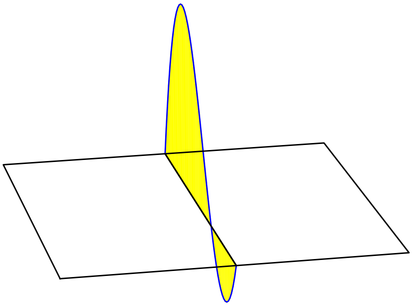


hp mesh

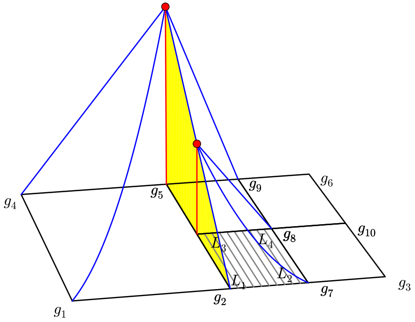


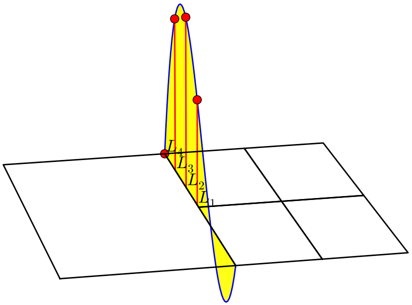
Hanging nodes – extra





- No global degrees of freedom assigned to hanging nodes
- Contributions to the local local stiffness matrix and RHS have to be adjusted to ensure continuity
- Works for higher-order basis functions as well
- Works for 2D and 3D for 1-irregular mesh and elements of same order
- Not numbering hanging nodes natural for p4est and BDDCML





- No global degrees of freedom assigned to hanging nodes
- Contributions to the local local stiffness matrix and RHS have to be adjusted to ensure continuity
- Works for higher-order basis functions as well
- Works for 2D and 3D for 1-irregular mesh and elements of same order
- Not numbering hanging nodes natural for p4est and BDDCML

An abstract problem

$$u \in U : a(u,v) = \langle f, v \rangle \quad \forall v \in U$$

- $a(\cdot,\cdot)$ symmetric positive definite form on U
- $\langle \cdot,\cdot \rangle$ is inner product on Hilbert space U
- U is finite dimensional space (typically finite element space)

Matrix form

$$u \in \mathbb{R}^n : Au = f$$

- A symmetric positive definite matrix
- A large, sparse, condition number $\kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}} = \mathcal{O}(1/h^2)$

Solving the linear system – extra

An abstract problem

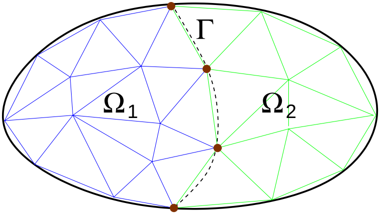
$$u \in U : a(u, v) = \langle f, v \rangle \quad \forall v \in U$$

- $a(\cdot, \cdot)$ symmetric positive definite form on U
- $\langle \cdot, \cdot \rangle$ is inner product on Hilbert space U
- U is finite dimensional space (typically finite element space)

Matrix form

$$u \in \mathbb{R}^n : Au = f$$

- A symmetric positive definite matrix
- A large, sparse, condition number $\kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}} = \mathcal{O}(1/h^2)$

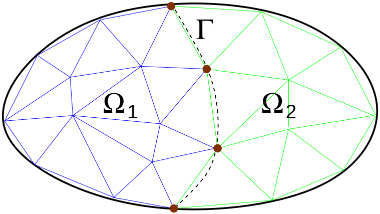


- Ω_1, Ω_2 …subdomains (substructures), do not overlap
- Γ …**interface**
- The goal is to do as much work as possible locally

Reduced (Schur complement) problem on interface Γ

$$Su_\Gamma = g$$

- S …Schur complement matrix
- S much smaller than A
- solved by PCG
- S never constructed, only its action on vector used



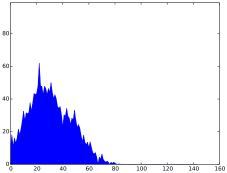
- Ω_1, Ω_2 ... subdomains (substructures), do not overlap
- Γ ... **interface**
- The goal is to do as much work as possible locally

Reduced (Schur complement) problem on interface Γ

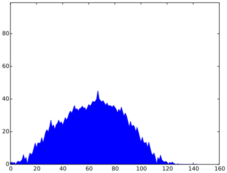
$$Su_\Gamma = g$$

- S ... Schur complement matrix
- S much smaller than A
- solved by PCG
- S never constructed, only its action on vector used

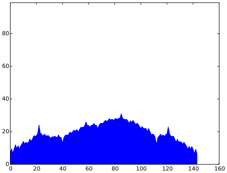
Which elements to refine – extra



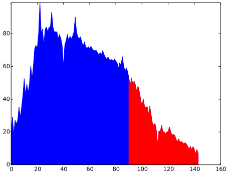
Processor 1



Processor 2



Processor 3



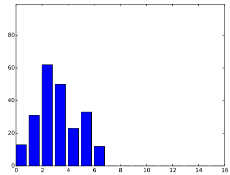
Global sum

x axis: Estimated element error
y axis: Number of elements

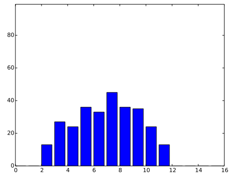
Goal: refine prescribed fraction of elements in each step

It might be too expensive to communicate error estimate of all elements to all processors

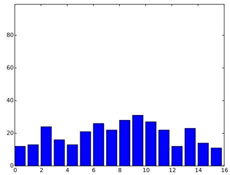
1. The first step is to communicate globally maximal element error



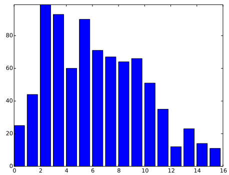
Processor 1



Processor 2



Processor 3

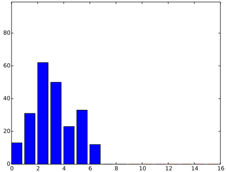


Global sum

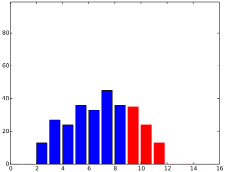
x axis: Estimated element error
y axis: Number of elements

2. On each processor, count elements in moderate number of error intervals (the same “bins” on all processors)
3. Communicate to the remaining processors - this can be done

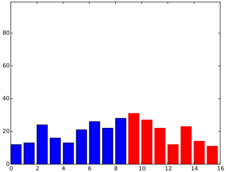
Which elements to refine – extra



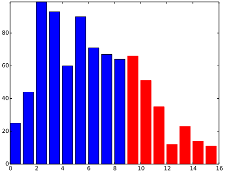
Processor 1



Processor 2



Processor 3



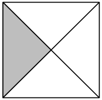
Global sum

x axis: Estimated element error
y axis: Number of elements

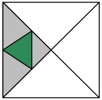
4. Using global data, determine error threshold leading to refinement of (approximately) given fraction of elements
5. Refine elements of estimated error larger than threshold

It can be none or all on some processors, but globally as required

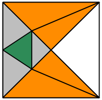
Continuity of basis functions – extra



marked element

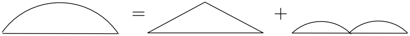


refinement
1-irregularity rule

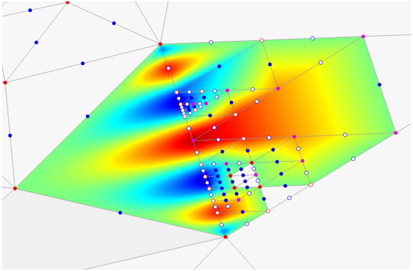


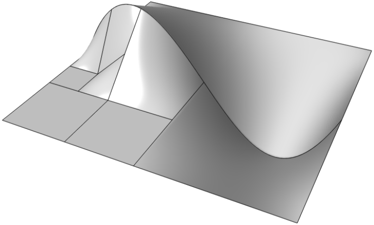
refinements forced
by the previous step

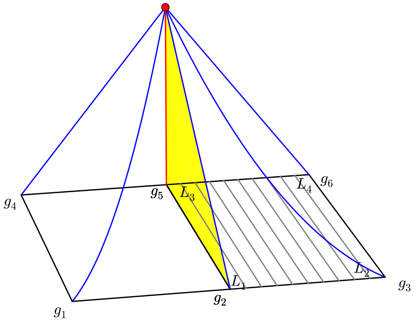
- Standard H^1-conforming elements
- continuity of basis functions has to be enforced
- "gluing" of basis functions
- for hanging nodes, combinations of shape functions



- For higher-order hanging nodes, many shape functions may contribute







$$(L_1, L_2, L_3, L_4) \leftrightarrow (g_2, g_3, g_5, g_6)$$

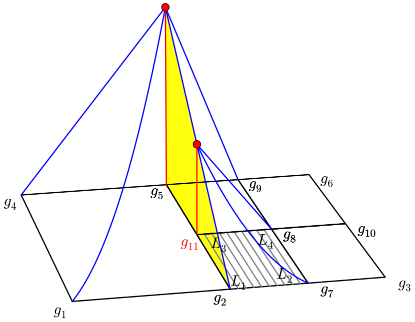
$$G_K = \begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$$

$$A_K = \begin{bmatrix} a(L_1, L_1) & \dots & a(L_1, L_4) \\ \vdots & \ddots & \vdots \\ a(L_4, L_1) & \dots & a(L_4, L_4) \end{bmatrix}$$

$$A = \sum_{K \in \mathcal{T}_k} G_K^T A_K G_K$$

Regular element

- Matrix G_K represents the relationship between local and global DOFs
- Local stiffness matrix distributed to the Global one
- Similarly for right-hand side



$$(L_1, L_2, L_3, L_4) \leftrightarrow (g_2, g_7, \mathbf{g_{11}}, g_8)$$

$$A = \sum_{K \in \mathcal{T}_k} G_K^T A_K G_K$$

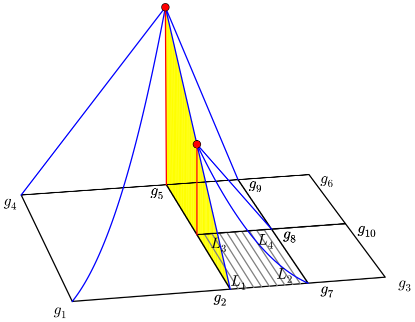
$$\tilde{A} = \begin{bmatrix} A \\ C \end{bmatrix}$$

$$g_{11} = \frac{g_2 + g_5}{2}$$

$$C = \left[0, \tfrac{1}{2}, 0, 0, \tfrac{1}{2}, 0, 0, 0, 0, 0, -1\right]$$

First option – constraints added to global matrix

- There are global degrees of freedom assigned to hanging nodes
- Matrix A assembled as in the regular case
- Solution would be discontinuous $\rightarrow$ the global system has to be extended by constraints C
- The matrix $\tilde{A}$ is rectangular, has to be modified before actual solution – various techniques



$$(L_1, L_2, L_3, L_4)^T = T_K(g_2, g_7, \mathbf{g_5}, g_8)^T$$

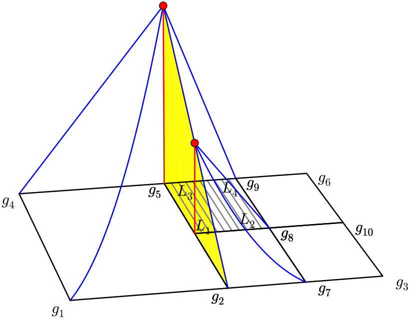
$$T_K = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ \mathbf{1/2} & 0 & \mathbf{1/2} & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$$

$$\bar{A}_K = T_K^T A_K T_K$$

$$A = \sum_{K \in \mathcal{T}_k} G_K^T \bar{A}_K G_K$$

Second option – local change of basis

- No global degrees of freedom assigned to hanging nodes
- Matrix T_K used to modify the local stiffness matrix and RHS
- Corresponds to the construction of globally continuous basis function (depicted the one corresponding to g_5)



$$(L_1, L_2, L_3, L_4)^T = T_K(\mathbf{g_2}, g_8, g_5, g_9)^T$$

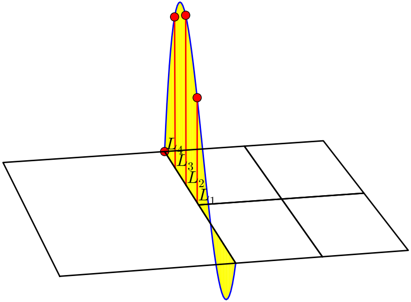
$$T_K = \begin{bmatrix} \mathbf{1/2} & 0 & \mathbf{1/2} & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$$

$$\bar{A}_K = T_K^T A_K T_K$$

$$A = \sum_{K \in \mathcal{T}_k} G_K^T \bar{A}_K G_K$$

Second option – local change of basis

- No global degrees of freedom assigned to hanging nodes
- Matrix T_K used to modify the local stiffness matrix and RHS
- Corresponds to the construction of globally continuous basis function (depicted the one corresponding to g_5)



$$T_K = \begin{bmatrix} \cdot & \cdot & v(L_1) & \cdot & \cdots \\ \cdot & \cdot & v(L_2) & \cdot & \cdots \\ \cdot & \cdot & v(L_3) & \cdot & \cdots \\ \cdot & \cdot & v(L_4) & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$\bar{A}_K = T_K^T A_K T_K$$

$$A = \sum_{K \in \mathcal{T}_k} G_K^T \bar{A}_K G_K$$

Second option – local change of basis

- Works for higher-order basis functions as well
- Works for 2D and 3D for 1-irregular mesh and elements of same order
- Not numbering hanging nodes natural for p4est $\rightarrow$ we use this approach

Fempar vs. BDDCML – extra

We use **non-overlapping** domain decomposition method, namely **BDDC**, a Balancing Domain Decomposition by Constraints. Experiments with two different libraries:

- **Fempar library**
  - A FORTRAN library for the development of Finite Element Multiphysics PARallel solvers
  - Developed at CIMNE, Barcelona, Spain by the group of Santiago Badia
  - Scales to hundreds of thousands of processor cores
  - Large project, includes all FEM machinery (space discretization, integration, assembling, physic-based preconditioners, . . .)
  - A lot of the code has to be aware of the hanging nodes
- **BDDCML library**
  - A FORTRAN library, BDDC **Multi Level**
  - Developed at IM AS CR by Jakub Šístek
  - Only linear algebra solver
  - Receives discrete system and some geometry information
  - It is much easier to deal with hanging nodes on the interface

Fempar vs. BDDCML – extra

We use **non-overlapping** domain decomposition method, namely **BDDC**, a Balancing Domain Decomposition by Constraints. Experiments with two different libraries:

- **Fempar library**
  - A FORTRAN library for the development of Finite Element Multiphysics PARallel solvers
  - Developed at CIMNE, Barcelona, Spain by the group of Santiago Badia
  - Scales to hundreds of thousands of processor cores
  - Large project, includes all FEM machinery (space discretization, integration, assembling, physic-based preconditioners, . . .)
  - A lot of the code has to be aware of the hanging nodes
- **BDDCML library**
  - A FORTRAN library, BDDC **Multi Level**
  - Developed at IM AS CR by Jakub Šístek
  - Only linear algebra solver
  - Receives discrete system and some geometry information
  - It is much easier to deal with hanging nodes on the interface

Selection of constraints – extra

Algorithm for generating constraints

1. classify interface to faces, edges and vertices — **components independent**
2. **for each subdomain**
   for each component
   for each face
   - select nodes on interface shared with neighbouring subdomain (generally larger set than the face under consideration) and detect components
   - select (in 3D) three nodes as corners from each such set as
     - pick arbitrary node of the set
     - find the first corner as the most remote node from the arbitrary node
     - find the second corner as the most remote node from the first corner
     - find the third corner as the node maximizing the are of the triangle
3. select corners as union of vertices and face-based selection
4. remove corners from edges and faces
5. use arithmetic averages on edges and faces

- extension on face-based selection of corners [Šístek et al. (2011)]
- amenable for parallelization

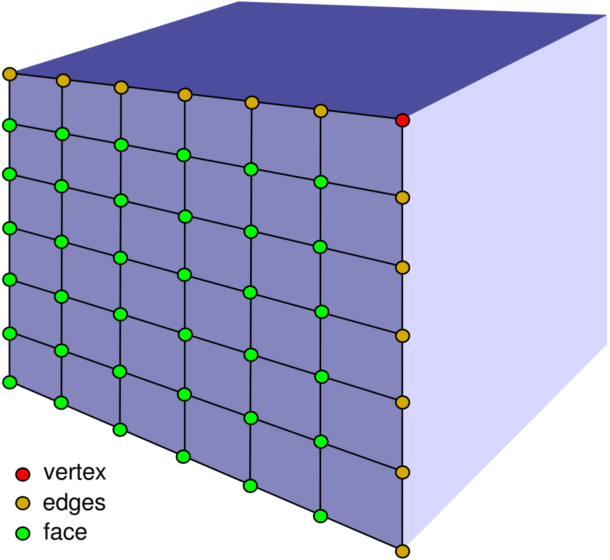
Selection of constraints – extra

Algorithm for generating constraints

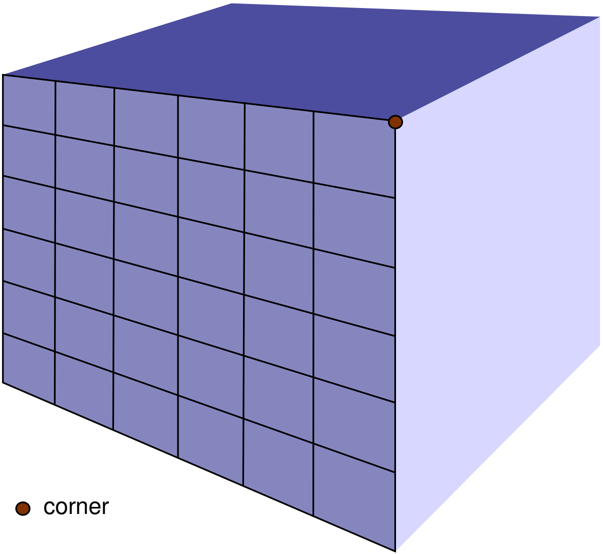
1. classify interface to faces, edges and vertices — **components independent**
2. **for each subdomain**
   for each component
   for each face
   - select nodes on interface shared with neighbouring subdomain (generally larger set than the face under consideration) and detect components
   - select (in 3D) three nodes as corners from each such set as
     - pick arbitrary node of the set
     - find the first corner as the most remote node from the arbitrary node
     - find the second corner as the most remote node from the first corner
     - find the third corner as the node maximizing the are of the triangle
3. select corners as union of vertices and face-based selection
4. remove corners from edges and faces
5. use arithmetic averages on edges and faces

- extension on face-based selection of corners [Šístek et al. (2011)]
- amenable for parallelization

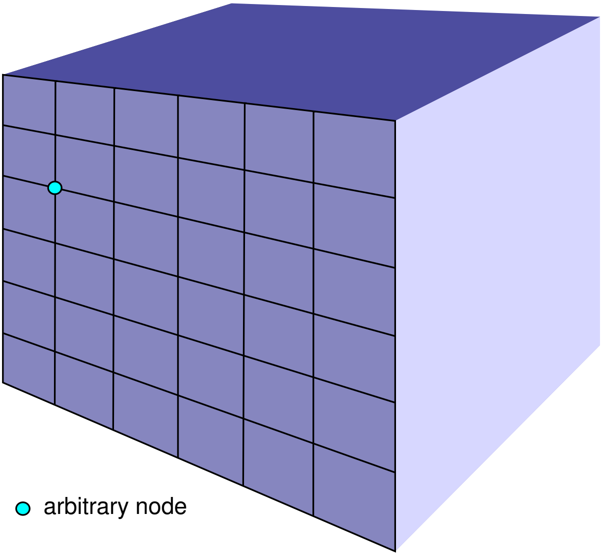
Selection of corners – extra



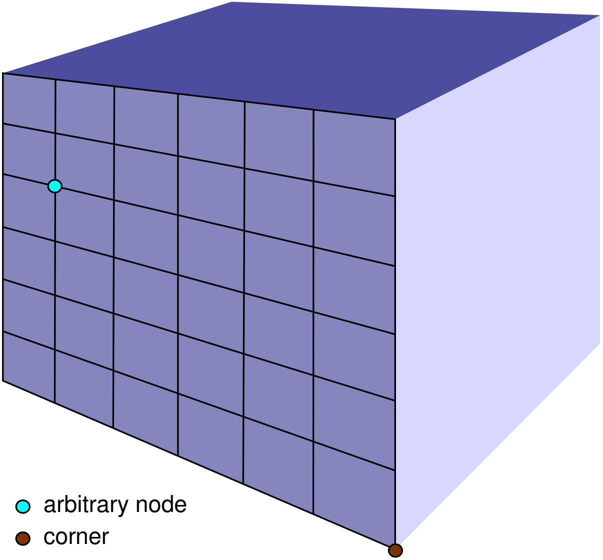
Selection of corners – extra



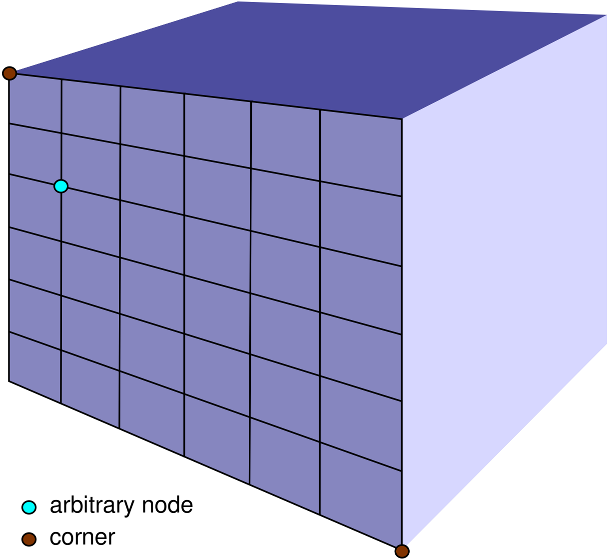
Selection of corners – extra



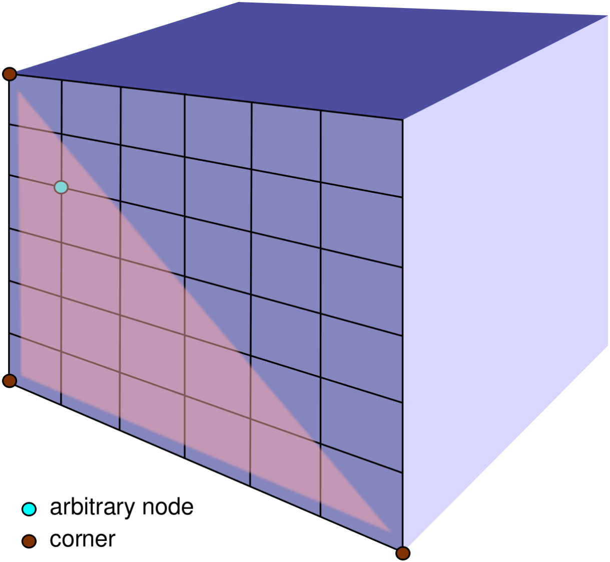
Selection of corners – extra



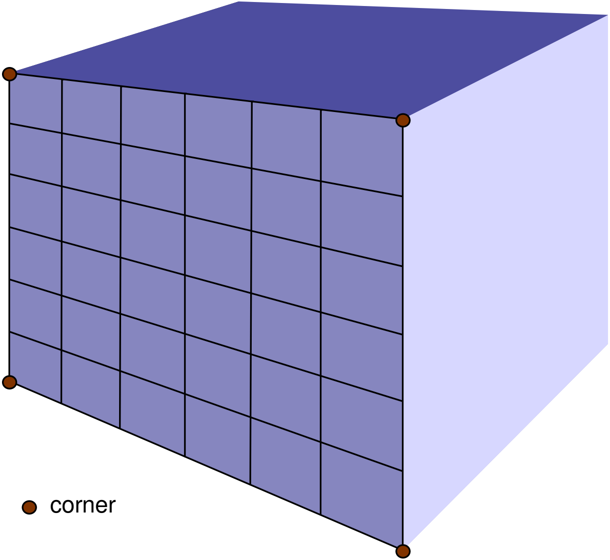
Selection of corners – extra



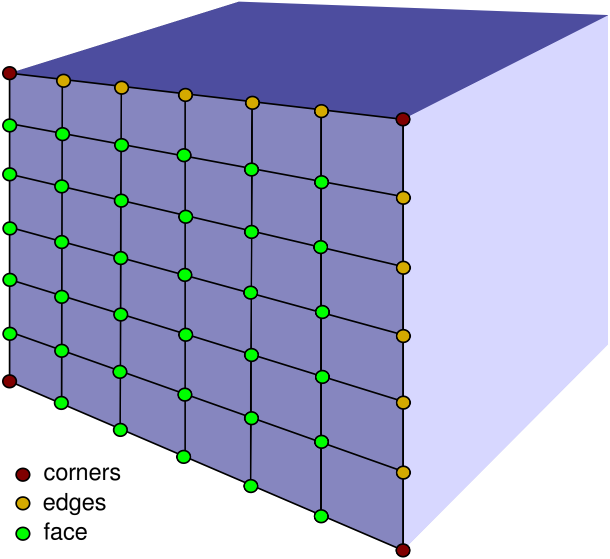
Selection of corners – extra



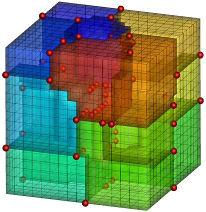
Selection of corners – extra

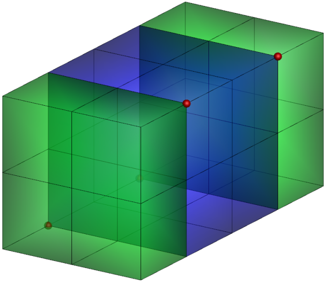


Selection of corners – extra

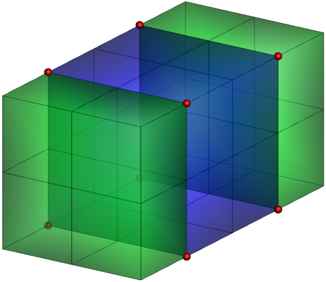


- geometric information useful for the solver
- component detection crucial for robustness of corner selection
- amenable for parallelization





components not detected



components detected

Parallel scaling tests – extra

How to set up the scaling test for adaptivity?

- different trajectories of adaptive computations for changing number of cores
- what time to measure — total solution time, time of the last problem from adaptive loop?
- setup of weak scaling tests unclear

Strong scaling tests on the final problem

- refinements are prescribed and always the same, not an adaptive run
- investigate the behaviour of DD on these nonstandard meshes

Parallel scaling tests – extra

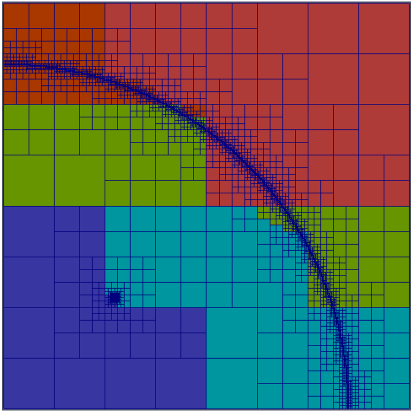
How to set up the scaling test for adaptivity?

- different trajectories of adaptive computations for changing number of cores
- what time to measure — total solution time, time of the last problem from adaptive loop?
- setup of weak scaling tests unclear

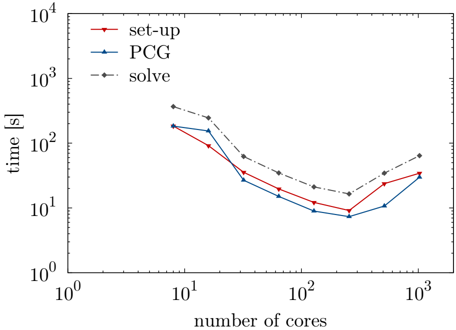
Strong scaling tests on the final problem

- refinements are prescribed and always the same, not an adaptive run
- investigate the behaviour of DD on these nonstandard meshes

Same **2D** mesh with **28M DOFs**, increasing number of subdomains



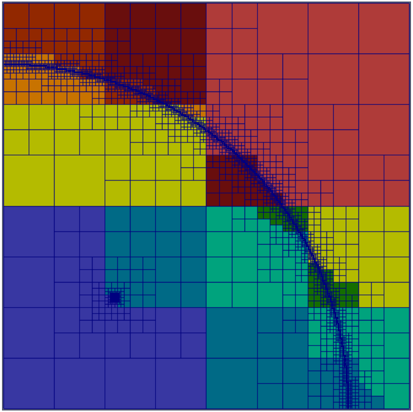
Illustrative mesh, 5000 DOFs,
5 subdomains



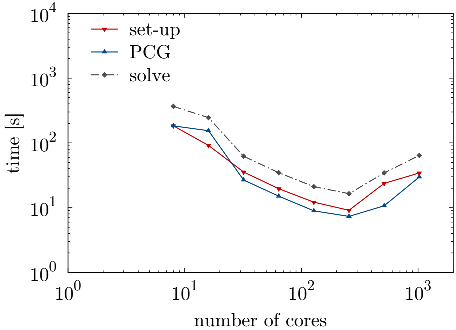
- run on *Salomon@IT4I*
- using 2-level method affects scaling
- should be improved by multi-level

Strong scaling in 2D (preliminary results)

Same **2D** mesh with **28M DOFs**, increasing number of subdomains



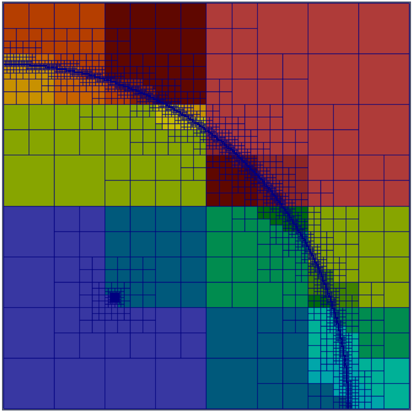
Illustrative mesh, 5000 DOFs,
10 subdomains



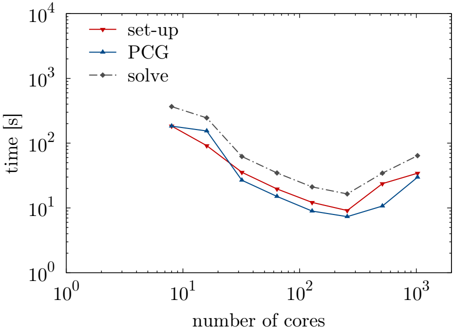
- run on *Salomon@IT4I*
- using 2-level method affects scaling
- should be improved by multi-level

Strong scaling in 2D (preliminary results)

Same **2D** mesh with **28M DOFs**, increasing number of subdomains



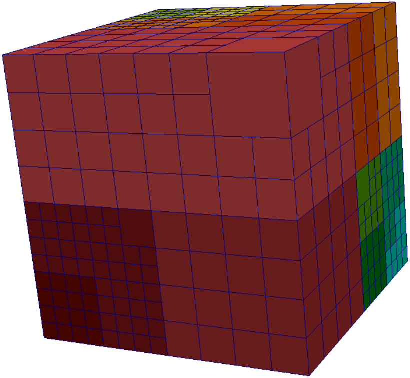
Illustrative mesh, 5000 DOFs,
20 subdomains.



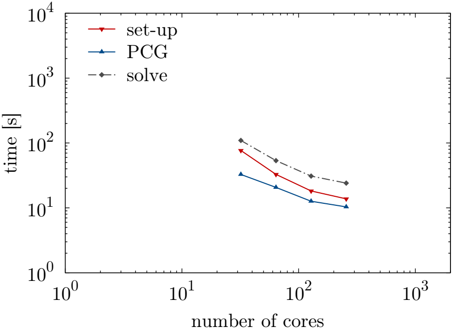
- run on *Salomon@IT4I*
- using 2-level method affects scaling
- should be improved by multi-level

Strong scaling in 3D (preliminary results)

Same **3D** mesh with **15M DOFs**, increasing number of subdomains



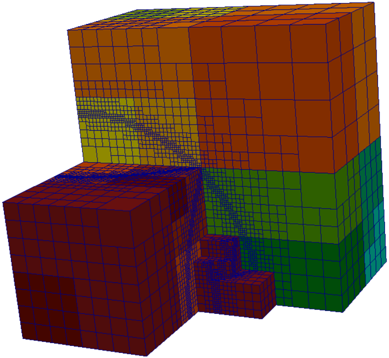
Illustrative mesh, 100K DOFs, 20 subdomains, shown 1 – 20



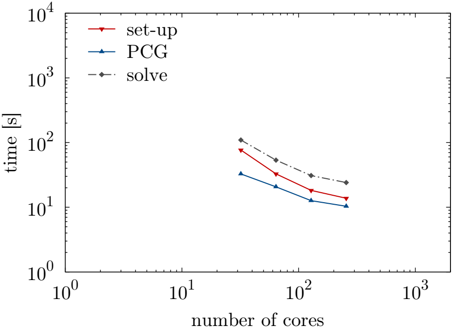
- run on *Salomon@IT4I*
- using 2-level method affects scaling
- 30–47 PCG iterations
- some issues not fully understood

Strong scaling in 3D (preliminary results)

Same **3D** mesh with **15M DOFs**, increasing number of subdomains

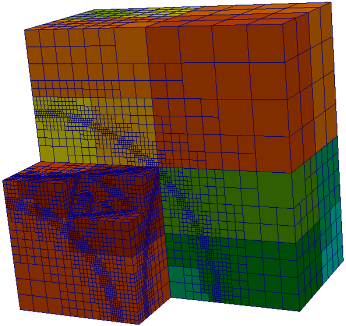


Illustrative mesh, 100K DOFs, 20 subdomains, shown 1 – 18

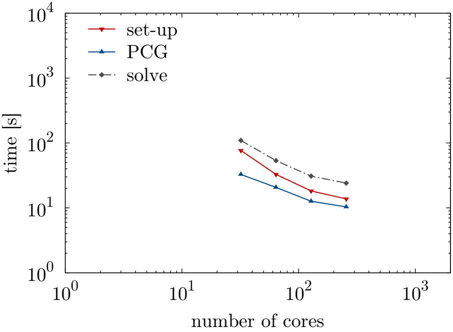


- run on *Salomon@IT4I*
- using 2-level method affects scaling
- 30–47 PCG iterations
- some issues not fully understood

Same **3D** mesh with **15M DOFs**, increasing number of subdomains



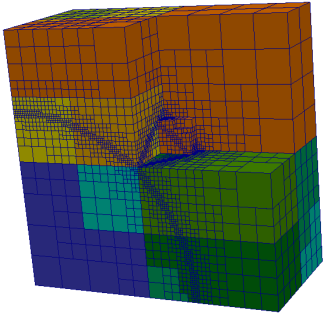
Illustrative mesh, 100K DOFs, 20 subdomains, shown 1 – 16



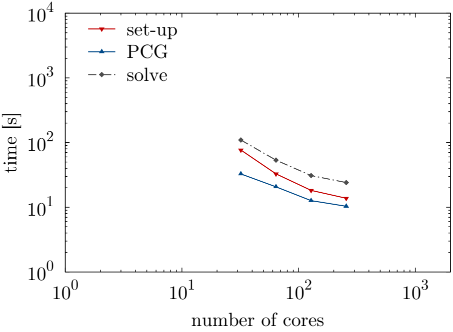
- run on *Salomon@IT4I*
- using 2-level method affects scaling
- 30–47 PCG iterations
- some issues not fully understood

Strong scaling in 3D (preliminary results)

Same **3D** mesh with **15M DOFs**, increasing number of subdomains

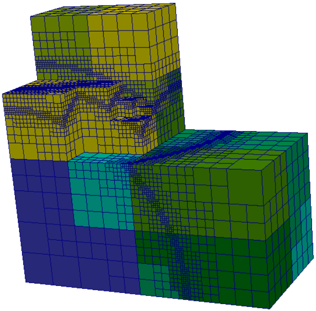


Illustrative mesh, 100K DOFs, 20 subdomains, shown 1 – 14

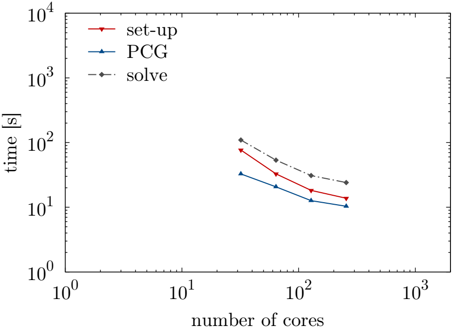


- run on *Salomon@IT4I*
- using 2-level method affects scaling
- 30–47 PCG iterations
- some issues not fully understood

Same **3D** mesh with **15M DOFs**, increasing number of subdomains

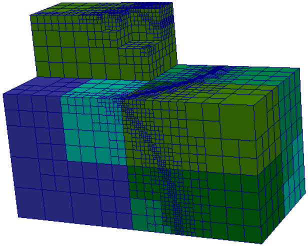


Illustrative mesh, 100K DOFs, 20 subdomains, shown 1 – 12

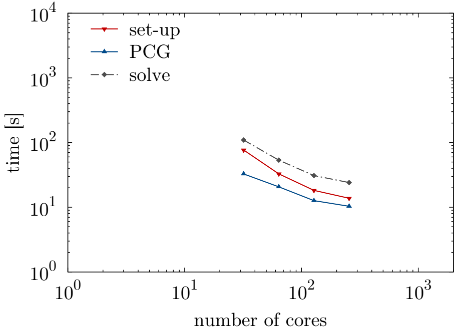


- run on *Salomon@IT4I*
- using 2-level method affects scaling
- 30–47 PCG iterations
- some issues not fully understood

Same **3D** mesh with **15M DOFs**, increasing number of subdomains

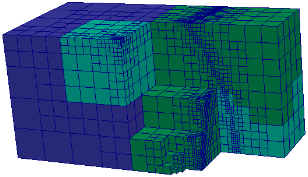


Illustrative mesh, 100K DOFs, 20 subdomains, shown 1 – 10

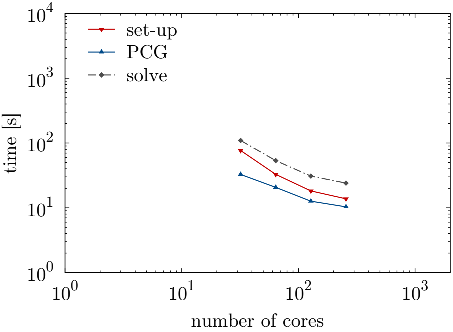


- run on *Salomon@IT4I*
- using 2-level method affects scaling
- 30–47 PCG iterations
- some issues not fully understood

Same **3D** mesh with **15M DOFs**, increasing number of subdomains



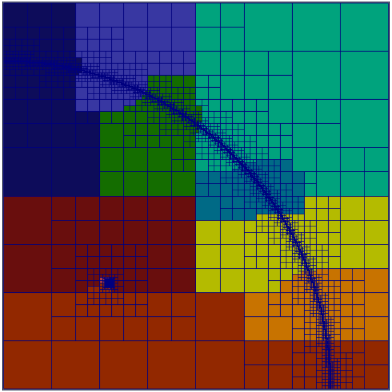
Illustrative mesh, 100K DOFs, 20 subdomains, shown 1 – 8.



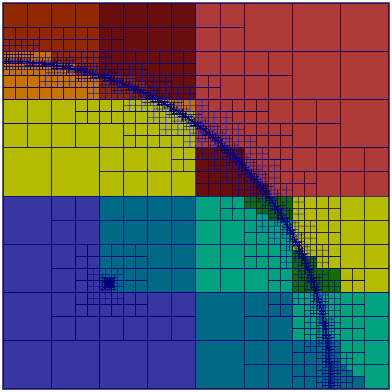
- run on *Salomon@IT4I*
- using 2-level method affects scaling
- 30–47 PCG iterations
- some issues not fully understood

p4est vs. Metis

The same mesh partitioned from Z-curve (p4est) and graph (Metis)



subdomains by p4est



subdomains by Metis

- no significant differences in numbers of iterations
- more tests required in this direction