Towards a parallel domain decomposition solver for immersed boundary finite element method

Pavel Kůs[1]
joint work with
Jakub Šístek[1], Fehmi Cirak[2], and Eky Febrianto[2]

[1]Institute of Mathematics, Czech Academy of Sciences, Prague
[2]Department of Engineering, University of Cambridge

May 26, 2021

Outline

Introduction

Parallel adaptive mesh refinement

Multilevel BDDC method

Immersed boundary FEM

Numerical results

Conclusion and outlooks

Outline

Introduction

Parallel adaptive mesh refinement

Multilevel BDDC method

Immersed boundary FEM

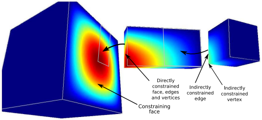
Numerical results

Conclusion and outlooks

Motivations and ingredients

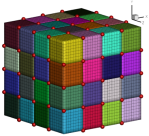
1. Adaptivity and higher order finite elements

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

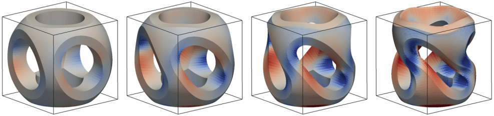


2. Nonoverlapping domain decomposition and parallel computing

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



3. Immersed boundary FEM



[T. Rüberg, F. Cirak, and J.M. Garcia-Aznar, *An unstructured immersed finite element method for nonlinear solid mechanics*, Advanced Modeling and Simulation in Engineering Sciences, 2016.]

Outline

Introduction

Parallel adaptive mesh refinement

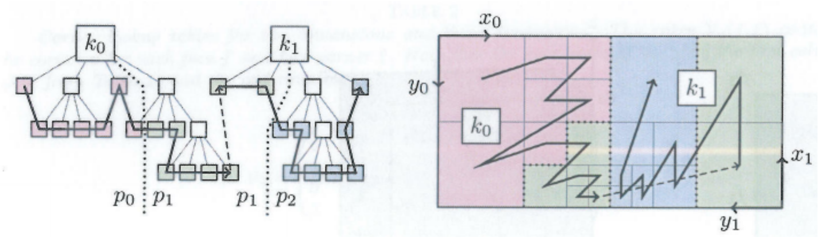
Multilevel BDDC method

Immersed boundary FEM

Numerical results

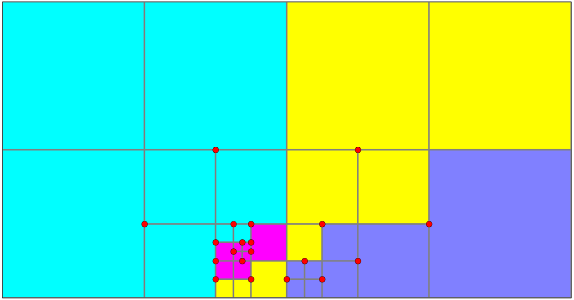
Conclusion and outlooks

The *p4est* library



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.

Nonstandard properties of the mesh

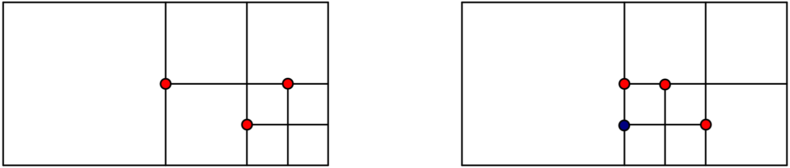


1. **Hanging nodes**
   - Hanging nodes have to be eliminated
   - They can also appear at the subdomain interface
2. **Shape of the subdomains**
   - Subdomains might be disconnected or only loosely coupled (e.g. by one node in elasticity)

Adding rules to the game

Assumptions on the mesh

- only level-1 hanging nodes allowed
- 2:1 rule
- equal order shape functions, i.e. no hp, but higher p fine



O.K. — not O.K.

Outline

Introduction

Parallel adaptive mesh refinement

Multilevel BDDC method

Immersed boundary FEM

Numerical results

Conclusion and outlooks

Solving the linear system

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 U
- U is finite dimensional space (typically finite element functions)

Matrix form

$$u \in U : Au = f$$

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

Solving the linear system

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 U
- U is finite dimensional space (typically finite element functions)

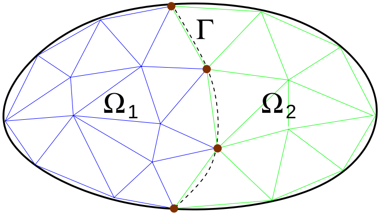
Matrix form

$$u \in U : Au = f$$

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

Nonoverlapping DD methods

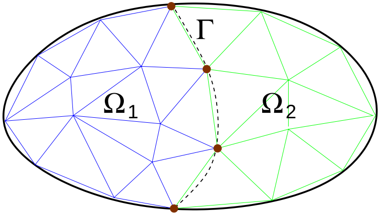
- idea goes back to **substructuring** – a trick used in seventies to fit larger FE problems into memory



  - Ω_1, Ω_2 ... subdomains (substructures)
  - Γ ... interface
- unknowns at interface are shared by more subdomains, remaining (interior) unknowns belong to a single subdomain
- the first step is reduction of the problem to the **interface** Γ

Nonoverlapping DD methods

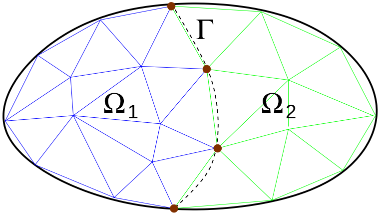
- idea goes back to **substructuring** – a trick used in seventies to fit larger FE problems into memory



  - Ω_1, Ω_2 ... subdomains (substructures)
  - Γ ... interface
- unknowns at interface are shared by more subdomains, remaining (interior) unknowns belong to a single subdomain
- the first step is reduction of the problem to the **interface** Γ

Nonoverlapping DD methods

- idea goes back to **substructuring** – a trick used in seventies to fit larger FE problems into memory



  - Ω_1, Ω_2 ... subdomains (substructures)
  - Γ ... interface
- unknowns at interface are shared by more subdomains, remaining (interior) unknowns belong to a single subdomain
- the first step is reduction of the problem to the **interface** Γ

Formation of the interface problem

- recall the matrix problem

$$Au = f$$

- reorder unknowns so that those at interior u_o^1 and u_o^2 are first, then interface u_Γ

$$\begin{bmatrix} A_{oo}^1 & & A_{o\Gamma}^1 \\ & A_{oo}^2 & A_{o\Gamma}^2 \\ A_{\Gamma o}^1 & A_{\Gamma o}^2 & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_o^1 \\ u_o^2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_o^1 \\ f_o^2 \\ f_\Gamma \end{bmatrix}$$

- eliminate interior unknowns – subdomain by subdomain = **in parallel**

$$\begin{bmatrix} A_{oo}^1 & & A_{o\Gamma}^1 \\ & A_{oo}^2 & A_{o\Gamma}^2 \\ & & S \end{bmatrix} \begin{bmatrix} u_o^1 \\ u_o^2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_o^1 \\ f_o^2 \\ g \end{bmatrix}$$

$$S = \sum_{assembly} A_{\Gamma\Gamma}^i - A_{\Gamma o}^i (A_{oo}^i)^{-1} A_{o\Gamma}^i = \sum_{assembly} S^i$$

$$g = \sum_{assembly} f_\Gamma^i - A_{\Gamma o}^i (A_{oo}^i)^{-1} f_o^i = \sum_{assembly} g^i$$

- recall the matrix problem

$$Au = f$$

- reorder unknowns so that those at interior u_o^1 and u_o^2 are first, then interface u_Γ

$$\begin{bmatrix} A_{oo}^1 & & A_{o\Gamma}^1 \\ & A_{oo}^2 & A_{o\Gamma}^2 \\ A_{\Gamma o}^1 & A_{\Gamma o}^2 & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_o^1 \\ u_o^2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_o^1 \\ f_o^2 \\ f_\Gamma \end{bmatrix}$$

- eliminate interior unknowns – subdomain by subdomain = **in parallel**

$$\begin{bmatrix} A_{oo}^1 & & A_{o\Gamma}^1 \\ & A_{oo}^2 & A_{o\Gamma}^2 \\ & & S \end{bmatrix} \begin{bmatrix} u_o^1 \\ u_o^2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_o^1 \\ f_o^2 \\ g \end{bmatrix}$$

$$S = \sum_{assembly} A_{\Gamma\Gamma}^i - A_{\Gamma o}^i (A_{oo}^i)^{-1} A_{o\Gamma}^i = \sum_{assembly} S^i$$

$$g = \sum_{assembly} f_\Gamma^i - A_{\Gamma o}^i (A_{oo}^i)^{-1} f_o^i = \sum_{assembly} g^i$$

Formation of the interface problem

- recall the matrix problem

$$Au = f$$

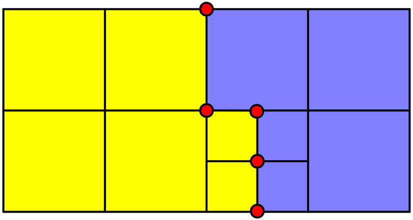
- reorder unknowns so that those at interior u_o^1 and u_o^2 are first, then interface u_Γ

$$\begin{bmatrix} A_{oo}^1 & & A_{o\Gamma}^1 \\ & A_{oo}^2 & A_{o\Gamma}^2 \\ A_{\Gamma o}^1 & A_{\Gamma o}^2 & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_o^1 \\ u_o^2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_o^1 \\ f_o^2 \\ f_\Gamma \end{bmatrix}$$

- eliminate interior unknowns – subdomain by subdomain = **in parallel**

$$\begin{bmatrix} A_{oo}^1 & & A_{o\Gamma}^1 \\ & A_{oo}^2 & A_{o\Gamma}^2 \\ & & S \end{bmatrix} \begin{bmatrix} u_o^1 \\ u_o^2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_o^1 \\ f_o^2 \\ g \end{bmatrix}$$

$$\begin{aligned} S &= \sum_{assembly} A_{\Gamma\Gamma}^i - A_{\Gamma o}^i (A_{oo}^i)^{-1} A_{o\Gamma}^i = \sum_{assembly} S^i \\ g &= \sum_{assembly} f_\Gamma^i - A_{\Gamma o}^i (A_{oo}^i)^{-1} f_o^i = \sum_{assembly} g^i \end{aligned}$$



- interface Γ

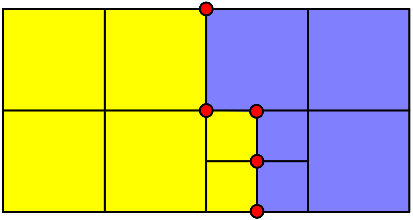
Reduced (Schur complement) problem on interface Γ

$$Su_\Gamma = g$$

$$S = \sum_{assembly} A^i_{\Gamma\Gamma} - A^i_{\Gamma o}(A^i_{oo})^{-1}A^i_{o\Gamma} = \sum_{assembly} S^i$$

$$g = \sum_{assembly} f^i_\Gamma - A^i_{\Gamma o}(A^i_{oo})^{-1}f^i_o = \sum_{assembly} g^i$$

- solved by PCG



- interface Γ

Reduced (Schur complement) problem on interface Γ

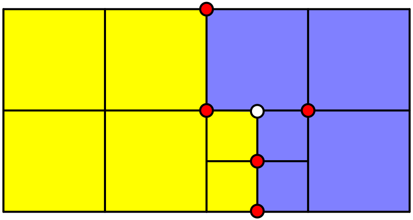
$$Su_\Gamma = g$$

$$S = \sum_{assembly} A^i_{\Gamma\Gamma} - A^i_{\Gamma o}(A^i_{oo})^{-1} A^i_{o\Gamma} = \sum_{assembly} S^i$$

$$g = \sum_{assembly} f^i_\Gamma - A^i_{\Gamma o}(A^i_{oo})^{-1} f^i_o = \sum_{assembly} g^i$$

- solved by PCG

Iterative substructuring



- interface Γ

Reduced (Schur complement) problem on interface Γ

$$Su_\Gamma = g$$

$$S = \sum_{assembly} A^i_{\Gamma\Gamma} - A^i_{\Gamma o}(A^i_{oo})^{-1} A^i_{o\Gamma} = \sum_{assembly} S^i$$

$$g = \sum_{assembly} f^i_\Gamma - A^i_{\Gamma o}(A^i_{oo})^{-1} f^i_o = \sum_{assembly} g^i$$

- solved by PCG

A practical algorithm of iterative substructuring

- In **setup**:
  1. factorize matrix A_{oo} (block diagonal = in parallel)
  2. form condensed right-hand side by solving

$$A_{oo}h = f_o,$$

and inserting $g = f_\Gamma - A_{\Gamma o}h$.

- In **each iteration**, for given p construct Sp as

$$\begin{bmatrix} A_{oo} & A_{o\Gamma} \\ A_{\Gamma o} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} w \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ Sp \end{bmatrix}$$

  1. Solve (in parallel) *discrete Dirichlet problem*

$$A_{oo}w = -A_{o\Gamma}p$$

  2. Get Sp (in parallel) as

$$Sp = A_{\Gamma o}w + A_{\Gamma\Gamma}p$$

- **After iterations**, for given u_Γ, resolve (in parallel) interior unknowns by back-substitution in

$$A_{oo}u_o = f_o - A_{o\Gamma}u_\Gamma$$

A practical algorithm of iterative substructuring

- In **setup**:
  1. factorize matrix A_{oo} (block diagonal = in parallel)
  2. form condensed right-hand side by solving

$$A_{oo}h = f_o,$$

  and inserting $g = f_\Gamma - A_{\Gamma o}h$.

- In **each iteration**, for given p construct Sp as

$$\begin{bmatrix} A_{oo} & A_{o\Gamma} \\ A_{\Gamma o} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} w \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ Sp \end{bmatrix}$$

  1. Solve (in parallel) *discrete Dirichlet problem*

$$A_{oo}w = -A_{o\Gamma}p$$

  2. Get Sp (in parallel) as

$$Sp = A_{\Gamma o}w + A_{\Gamma\Gamma}p$$

- **After iterations**, for given u_Γ, resolve (in parallel) interior unknowns by back-substitution in

$$A_{oo}u_o = f_o - A_{o\Gamma}u_\Gamma$$

A practical algorithm of iterative substructuring

- In **setup**:
  1. factorize matrix A_{oo} (block diagonal = in parallel)
  2. form condensed right-hand side by solving

$$A_{oo}h = f_o,$$

  and inserting $g = f_\Gamma - A_{\Gamma o}h$.

- In **each iteration**, for given p construct Sp as

$$\begin{bmatrix} A_{oo} & A_{o\Gamma} \\ A_{\Gamma o} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} w \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ Sp \end{bmatrix}$$

  1. Solve (in parallel) *discrete Dirichlet problem*

$$A_{oo}w = -A_{o\Gamma}p$$

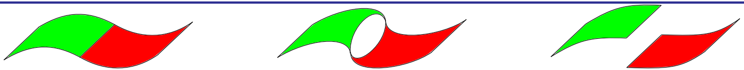
  2. Get Sp (in parallel) as

$$Sp = A_{\Gamma o}w + A_{\Gamma\Gamma}p$$

- **After iterations**, for given u_Γ, resolve (in parallel) interior unknowns by back-substitution in

$$A_{oo}u_o = f_o - A_{o\Gamma}u_\Gamma$$

The BDDC preconditioner



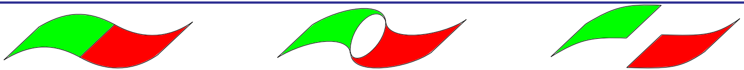
U	$\subset$	$\widetilde{W}$	$\subset$	W
continuous at all nodes at interface		continuous at selected coarse dofs		no continuity at interface

- 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 $\widetilde{W}$

Variational form of $M_{BDDC} : r \longmapsto u$

$$w \in \widetilde{W} : \quad a(w,z) = \langle r, Ez \rangle \quad \forall z \in \widetilde{W}$$
$$u = Ew$$

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 $\widetilde{W}$

Variational form of $M_{BDDC} : r \longmapsto u$

$$w \in \widetilde{W} : \quad a(w,z) = \langle r, Ez \rangle \quad \forall z \in \widetilde{W}$$
$$u = Ew$$

The multilevel BDDC preconditioner

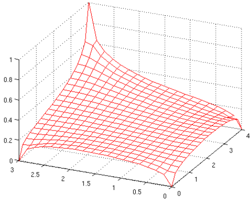
Local energy minimization problems

On each subdomain – **coarse degrees of freedom** – basis functions Ψ^i – prescribed values of coarse degrees of freedom, minimal energy elsewhere,

$$\begin{bmatrix} A^i & C^{iT} \\ C^i & 0 \end{bmatrix} \begin{bmatrix} \Psi^i \\ \Lambda^i \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}.$$

- A^i ... local subdomain stiffness matrix
- C^i ... matrix of constraints – selects unknowns into coarse degrees of freedom

Matrix of **coarse problem** A_C assembled from local matrices $A_{Ci} = \Psi^{iT} A^i \Psi^i = -\Lambda^i$.



coarse basis fun.

Disconnected and loosely coupled subdomains

Assumption on solvability of local saddle-point systems

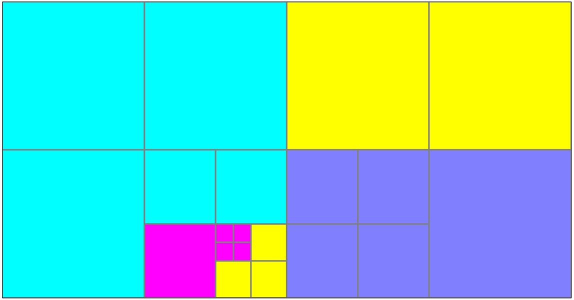
$$\mathrm{null}\, A^i \cap \mathrm{null}\, C^i = \{\mathbf{0}\}$$

- invertibility of
$$\begin{bmatrix} A^i & C^{iT} \\ C^i & 0 \end{bmatrix}$$
follows, see e.g. [Benzi, Golub, Liesen (2005)]
- satisfied if enough constraints on continuity in $\widetilde{W}$ are selected

Disconnected and loosely coupled subdomains

- more constrains needed in C^i
- more coarse basis functions Ψ^i

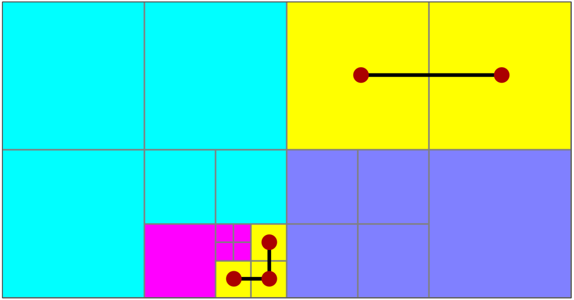
Disconnected and loosely coupled subdomains



- local nullspaces more complicated
- more constrains needed for each subdomain
- detect **graph components** of subdomain mesh

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

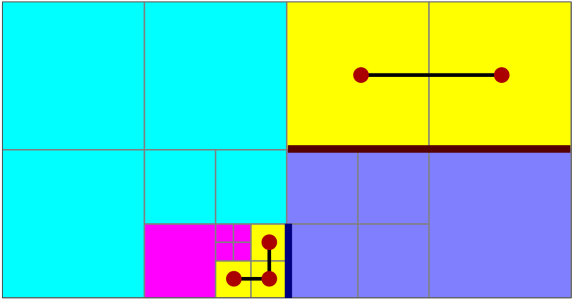
Disconnected and loosely coupled subdomains



- local nullspaces more complicated
- more constrains needed for each subdomain
- detect **graph components** of subdomain mesh

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

Disconnected and loosely coupled subdomains



- local nullspaces more complicated
- more constrains needed for each subdomain
- detect **graph components** of subdomain mesh

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

One step of BDDC

Get residual at interface nodes $r_\Gamma^{(k)} = g - Su_\Gamma^{(k)}$ and produce **preconditioned residual** $z_\Gamma^{(k)} = M_{BDDC} r_\Gamma^{(k)}$ by

1. 1. Distribution of residual

subdomain problems | (global) coarse problem

for $i = 1, \dots, N$

$$r^i = E^{iT} r_\Gamma^{(k)}$$

$$r_C = \sum_{i=1}^{N} R_C^{iT} \Psi^{i*T} E^{iT} r_\Gamma^{(k)}$$

2. Correction of solution

$$\begin{bmatrix} A^i & C^{iT} \\ C^i & 0 \end{bmatrix} \begin{bmatrix} z^i \\ \mu^i \end{bmatrix} = \begin{bmatrix} r^i \\ 0 \end{bmatrix}$$

$$A_C u_C = r_C$$

3. Combination of subdomain and coarse corrections

$$z_\Gamma^{(k)} = \sum_{i=1}^{N} E^i \left(\Psi^i R_C^i u_C + z^i \right)$$

One step of BDDC

Get residual at interface nodes $r_\Gamma^{(k)} = g - Su_\Gamma^{(k)}$ and produce **preconditioned residual** $z_\Gamma^{(k)} = M_{BDDC} r_\Gamma^{(k)}$ by

1. 1. Distribution of residual

subdomain problems

for $i = 1, \ldots, N$

$$r^i = E^{iT} r_\Gamma^{(k)}$$

(global) coarse problem

$$r_C = \sum_{i=1}^{N} R_C^{iT} \Psi^{i*T} E^{iT} r_\Gamma^{(k)}$$

2. Correction of solution

$$\begin{bmatrix} A^i & C^{iT} \\ C^i & 0 \end{bmatrix} \begin{bmatrix} z^i \\ \mu^i \end{bmatrix} = \begin{bmatrix} r^i \\ 0 \end{bmatrix}$$

$$A_C u_C = r_C$$

3. Combination of subdomain and coarse corrections

$$z_\Gamma^{(k)} = \sum_{i=1}^{N} E^i \left(\Psi^i R_C^i u_C + z^i \right)$$

One step of BDDC

Get residual at interface nodes $r_\Gamma^{(k)} = g - S u_\Gamma^{(k)}$ and produce **preconditioned residual** $z_\Gamma^{(k)} = M_{BDDC} r_\Gamma^{(k)}$ by

1. 1. Distribution of residual

subdomain problems

for $i = 1, \ldots, N$

$$r^i = E^{iT} r_\Gamma^{(k)}$$

(global) coarse problem

$$r_C = \sum_{i=1}^{N} R_C^{iT} \Psi^{i*T} E^{iT} r_\Gamma^{(k)}$$

2. Correction of solution

$$\begin{bmatrix} A^i & C^{iT} \\ C^i & 0 \end{bmatrix} \begin{bmatrix} z^i \\ \mu^i \end{bmatrix} = \begin{bmatrix} r^i \\ 0 \end{bmatrix}$$

$$A_C u_C = r_C$$

3. Combination of subdomain and coarse corrections

$$z_\Gamma^{(k)} = \sum_{i=1}^{N} E^i \left(\Psi^i R_C^i u_C + z^i \right)$$

One step of BDDC

Get residual at interface nodes $r_\Gamma^{(k)} = g - Su_\Gamma^{(k)}$ and produce **preconditioned residual** $z_\Gamma^{(k)} = M_{BDDC} r_\Gamma^{(k)}$ by

1. 1. Distribution of residual

   subdomain problems

   for $i = 1, \dots, N$

   $$r^i = E^{iT} r_\Gamma^{(k)}$$

   (global) coarse problem

   $$r_C = \sum_{i=1}^{N} R_C^{iT} \Psi^{i*T} E^{iT} r_\Gamma^{(k)}$$

2. Correction of solution

   $$\begin{bmatrix} A^i & C^{iT} \\ C^i & 0 \end{bmatrix} \begin{bmatrix} z^i \\ \mu^i \end{bmatrix} = \begin{bmatrix} r^i \\ 0 \end{bmatrix}$$

   $$A_C u_C = r_C$$

3. Combination of subdomain and coarse corrections

   $$z_\Gamma^{(k)} = \sum_{i=1}^{N} E^i \left(\Psi^i R_C^i u_C + z^i \right)$$

Outline

Introduction

Parallel adaptive mesh refinement

Multilevel BDDC method

Immersed boundary FEM

Numerical results

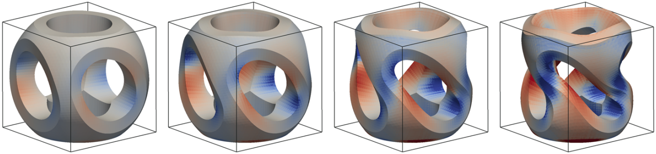
Conclusion and outlooks

Immersed boundary FEM

- an approach to avoiding mesh generation in FEM simulations
- robustness for geometric resolution through implicit description of the boundary

Nonstandard features compared to FEM

- weak enforcement of Dirichlet boundary conditions
- nonstandard quadrature rules
- ill-conditioning of stiffness matrices due to cut cells



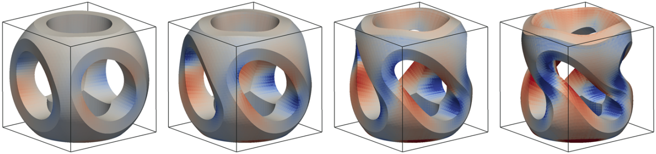
T. Rüberg, F. Cirak, and J.M. Garcia-Aznar, *An unstructured immersed finite element method for nonlinear solid mechanics*, Advanced Modeling and Simulation in Engineering Sciences, 2016.

Immersed boundary FEM

- an approach to avoiding mesh generation in FEM simulations
- robustness for geometric resolution through implicit description of the boundary

Nonstandard features compared to FEM

- weak enforcement of Dirichlet boundary conditions
- nonstandard quadrature rules
- ill-conditioning of stiffness matrices due to cut cells



T. Rüberg, F. Cirak, and J.M. Garcia-Aznar, *An unstructured immersed finite element method for nonlinear solid mechanics*, Advanced Modeling and Simulation in Engineering Sciences, 2016.

Weak enforcement of the boundary conditions

$$u \in H^1(\Omega): \quad a(u,v) = l(v) \quad \forall v \in H^1(\Omega)$$

Penalty method

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega + \int_{\Gamma_D} \gamma u v \, \mathrm{d}\Gamma$$

$$l(v) = \int_\Omega f v \, \mathrm{d}\Omega + \int_{\Gamma_N} \bar{t} v \, \mathrm{d}\Gamma + \int_{\Gamma_D} \gamma \bar{u} v \, \mathrm{d}\Gamma$$

Nitsche method

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega - \int_{\Gamma_D} (\mathbf{n} \cdot \nabla u) v \, \mathrm{d}\Gamma - \int_{\Gamma_D} (\mathbf{n} \cdot \nabla v) u \, \mathrm{d}\Gamma + \int_{\Gamma_D} \gamma u v \, \mathrm{d}\Gamma$$

$$l(v) = \int_\Omega f v \, \mathrm{d}\Omega + \int_{\Gamma_N} \bar{t} v \, \mathrm{d}\Gamma - \int_{\Gamma_D} (\mathbf{n} \cdot \nabla v) \bar{u} \, \mathrm{d}\Gamma + \int_{\Gamma_D} \gamma \bar{u} v \, \mathrm{d}\Gamma$$

- $\gamma = \gamma_0 \alpha / h$

Weak enforcement of the boundary conditions

$$u \in H^1(\Omega): \quad a(u,v) = l(v) \quad \forall v \in H^1(\Omega)$$

Penalty method

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \; \mathrm{d}\Omega + \int_{\Gamma_D} \gamma u v \; \mathrm{d}\Gamma$$

$$l(v) = \int_\Omega f v \; \mathrm{d}\Omega + \int_{\Gamma_N} \bar{t} v \; \mathrm{d}\Gamma + \int_{\Gamma_D} \gamma \bar{u} v \; \mathrm{d}\Gamma$$

Nitsche method

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \; \mathrm{d}\Omega - \int_{\Gamma_D} (\mathbf{n} \cdot \nabla u) v \; \mathrm{d}\Gamma - \int_{\Gamma_D} (\mathbf{n} \cdot \nabla v) u \; \mathrm{d}\Gamma + \int_{\Gamma_D} \gamma u v \; \mathrm{d}\Gamma$$

$$l(v) = \int_\Omega f v \; \mathrm{d}\Omega + \int_{\Gamma_N} \bar{t} v \; \mathrm{d}\Gamma - \int_{\Gamma_D} (\mathbf{n} \cdot \nabla v) \bar{u} \; \mathrm{d}\Gamma + \int_{\Gamma_D} \gamma \bar{u} v \; \mathrm{d}\Gamma$$

- $\gamma = \gamma_0 \alpha / h$

Implementation

Parallel FEM solver with AMR and embedded domains

- experimental in-house code
- implicit geometry description
- C++ with 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

Outline

Introduction

Parallel adaptive mesh refinement

Multilevel BDDC method

Immersed boundary FEM

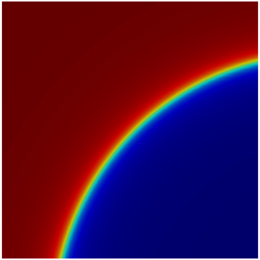
Numerical results

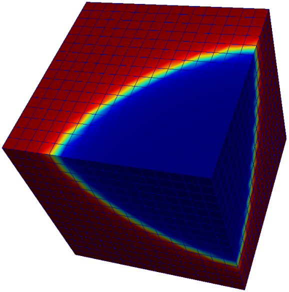
Conclusion and outlooks

Adaptive mesh refinement

Internal layer benchmark

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

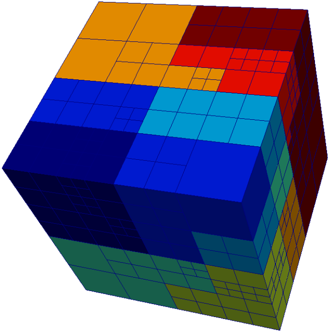


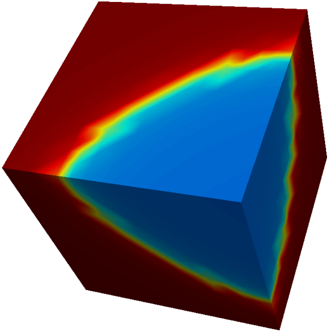


- solution exhibits sharp internal layer
- r is a distance from a given point
- s controls "steepness" of the layer

Adaptivity in 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

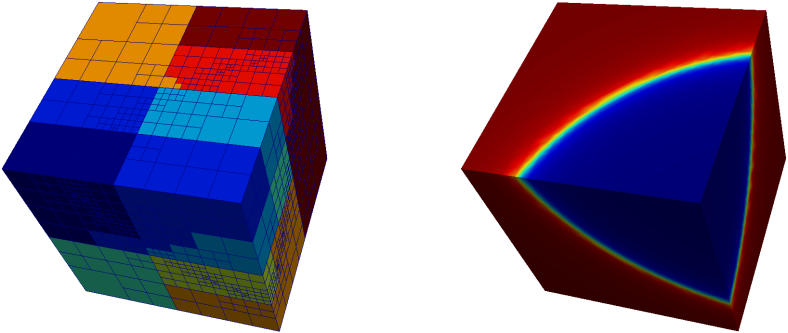




Iteration 3, mesh and solution

Adaptivity in 3D on 8 subdomains

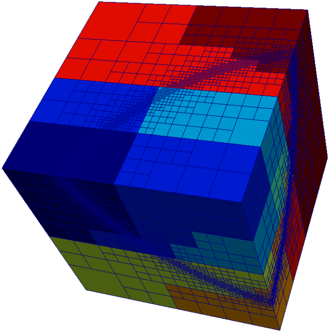
- Adaptivity tested for element orders 1-4 (showed order 1)
- Guided by exact solution, using H^1 semi-norm for error calculation

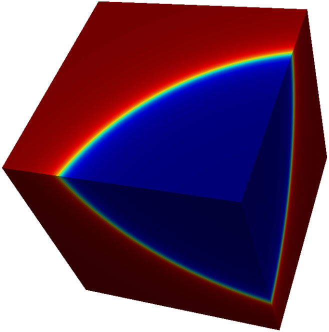


Iteration 5, mesh and solution

Adaptivity 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 8, mesh and solution

Adaptivity in 3D, 3-level BDDC, linear elements

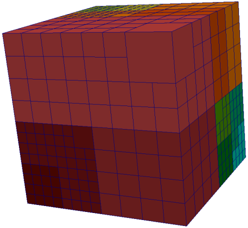
Adaptive mesh refinement

subs.	size	loc. size	PCG its.	time set-up [s]	time PCG [s]
2048/46	4913	2	**9**	2.8	1.1
2048/46	8594	4	**29**	0.55	1.6
2048/46	$2.5 \cdot 10^4$	12	**53**	0.6	3
2048/46	$1.3 \cdot 10^5$	63	**60**	0.67	3.5
2048/46	$7.0 \cdot 10^5$	342	**54**	0.89	3.5
2048/46	$3.0 \cdot 10^6$	1445	**56**	1.6	4.8
2048/46	$1.4 \cdot 10^7$	6623	**55**	2.9	10
2048/46	$6.4 \cdot 10^7$	$3.1 \cdot 10^4$	**55**	10	33
2048/46	$2.9 \cdot 10^8$	$1.4 \cdot 10^5$	**56**	61	130
2048/46	$1.3 \cdot 10^9$	$6.3 \cdot 10^5$	**51**	565	521

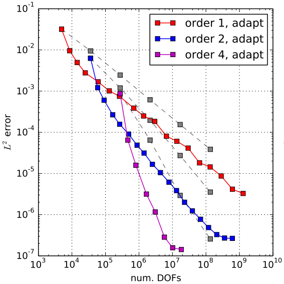
- run on *Salomon@IT4I*

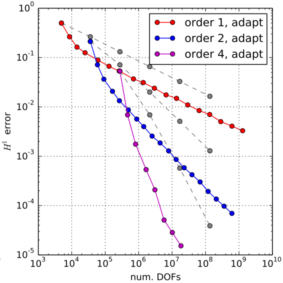
P. Kůs, J. Šístek. Coupling parallel adaptive mesh refinement with a nonoverlapping domain decomposition solver. Advances in Engineering Software, 110:34–54, 2017.

Adaptivity in 3D, 3-level BDDC, convergence order

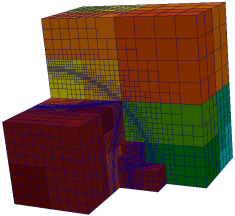


Illustrative mesh,
100K DOFs,
20 subdomains,
shown 1 – 20

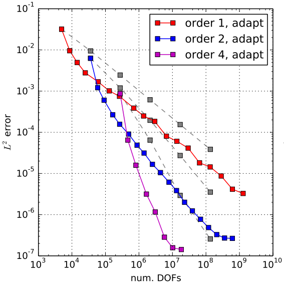


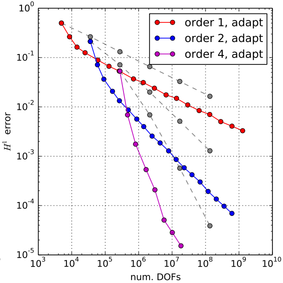


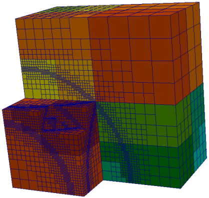
Adaptivity in 3D, 3-level BDDC, convergence order



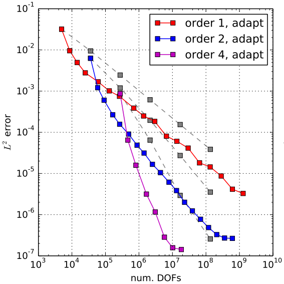
Illustrative mesh,
100K DOFs,
20 subdomains,
shown 1 – 18

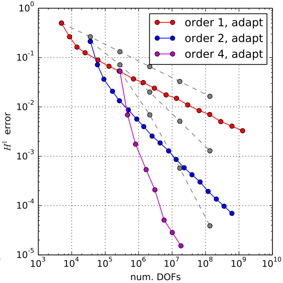


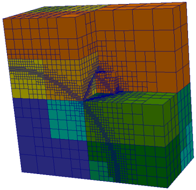




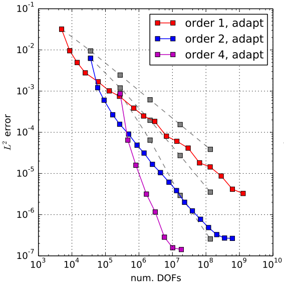
Illustrative mesh,
100K DOFs,
20 subdomains,
shown 1 – 16

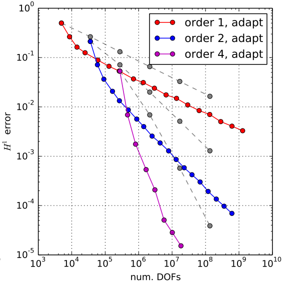




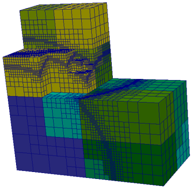


Illustrative mesh,
100K DOFs,
20 subdomains,
shown 1 – 14

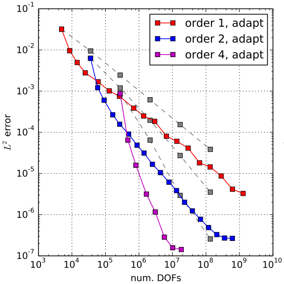


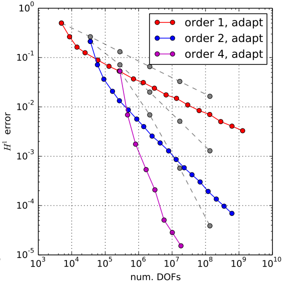


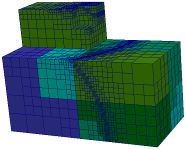
Adaptivity in 3D, 3-level BDDC, convergence order



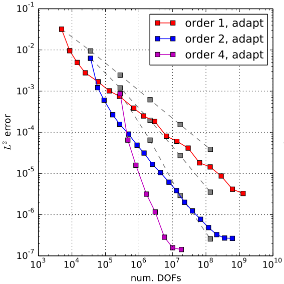
Illustrative mesh,
100K DOFs,
20 subdomains,
shown 1 – 12

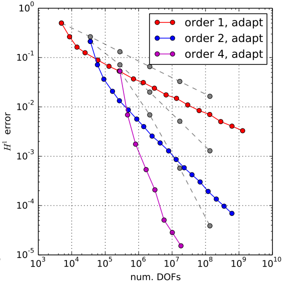


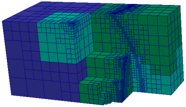




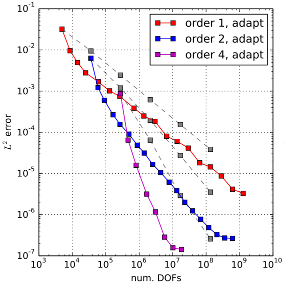
Illustrative mesh,
100K DOFs,
20 subdomains,
shown 1 – 10

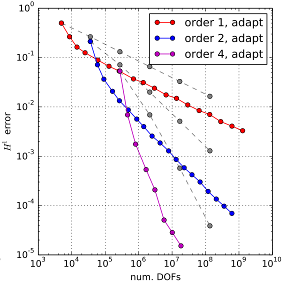






Illustrative mesh,
100K DOFs,
20 subdomains,
shown 1 – 8.





Outline

Introduction

Parallel adaptive mesh refinement

Multilevel BDDC method

Immersed boundary FEM

Numerical results

Conclusion and outlooks

Our use of direct solver

Two matrix factorizations

- 50% runtime (of the solver itself) spent in matrix factorizations, another 30% in back substitions (in iterations).
- We factorize two matrices on each subdomain. First A_{oo} with inner DOFs, the other with approx. 20% extra rows and columns, a saddle point problem. The first matrix is a submatrix of the second one.
- For Poisson problem and linear elasticity we use Cholesky and LDL^T decompositions, respectively. LU with pivoting used for non-symmetric problems.

$$\begin{bmatrix} A_{oo} & A_{o\Gamma} & 0 \\ A_{\Gamma o} & A_{\Gamma\Gamma} & C_\Gamma^T \\ 0 & C_\Gamma & 0 \end{bmatrix}$$

- We need to invert A_{oo} and the 3 by 3 block matrix on each subdom.
- For eliptic problems, A_{oo} is positive definite, the first 2 by 2 block matrix is positive semidefinite and the whole 3 by 3 block matrix is a saddle point problem.
- Only the part of C_i corresponding to the interface Γ is nonzero (C_Γ).

Our use of direct solver

Distibution of subdomains

- Currently we run pure MPI with one subdomain per MPI rank, trying to balance the subdomain sizes.
- This is sometimes difficult to achieve when using the parallel adaptivity of the mesh and many subdomain components.
- Having variable number of subdomains per node is appealing from various reasons, but would require a major code refactoring.

Contacts

Thank you for your attention!

Pavel Kůs

kus@math.cas.cz

Institute of Mathematics, Czech Academy of Sciences, Prague

BDDCML library

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