Efficient solution techniques for isogeometric analysis

Matthias Möller

Department of Applied Mathematics
Delft University of Technology, NL

SFB 1313 Lecture Series

22-02-2022
About me

- Associate Professor of Numerical Analysis at DIAM/TU Delft
- PhD and PostDoc at the Chair of Applied Mathematics and Numerics/TU Dortmund

Research interests

- Finite element and isogeometric analysis
- Adaptive high-resolution schemes for flow problems
- Fast solution techniques for (non-)linear problems
- High-performance and quantum-accelerated computing
- Scientific machine learning
About me

• Associate Professor of Numerical Analysis at DIAM/TU Delft
• PhD and PostDoc at the Chair of Applied Mathematics and Numerics/TU Dortmund

Research interests

• Finite element and isogeometric analysis
• Adaptive high-resolution schemes for flow problems
• Fast solution techniques for (non-)linear problems
• High-performance and quantum-accelerated computing
• Scientific machine learning

⇒ MS-12: Scientific machine learning in computational mechanics
  9th GACM Colloquium on Computational Mechanics in Essen, September 21-23, 2022
The IGA team

Jochen Hinz (EPFL)  Roel Tielen (ASML)  Hugo Verhelst (TUD)  Andrzej Jaeschke (Łódź)

Collaborations
Göddeke (U Stuttgart), Elgeti/Helmig (RWTH Aachen, TU Vienna), Mantzaflaris (INRIA), Gauger (TU K’lautern), Jüttler (JKU), Simeon (TU K’lautern), ...

Funding
EU-H2020 MOTOR (GA 678727), NWO FlexFloat starting 2022 (⇒ will open soon)
Isogeometric Analysis

Following examples may be mentioned: shell buckling analysis is very sensitive to geometric imperfections, boundary layer phenomena and lift and drag are sensitive to precise geometry of aerodynamic and hydrodynamic configurations, and sliding contact between bodies cannot be accurately represented without precise geometric descriptions. Automatic adaptive mesh refinement has not been as widely adopted in industry as one might assume from the extensive academic literature because mesh refinement requires access to the exact geometry, and thus it also requires seamless and automatic communication with CAD, which simply does not exist. Without accurate geometry and mesh adaptivity, convergence and precision results are in many cases impossible.

Deficiencies in current engineering analysis procedures also preclude successful application of important pace setting technologies.
My personal ‘top 3 features’ of IGA

1. Unified mathematical approach towards geometry modelling and PDE analysis

\[ \mathbf{x}(\xi, \eta) = \sum_{i,j} x_{i,j} N^P_i(\xi) N^q_j(\eta) \]

\[ u(\xi, \eta) = \sum_{i,j} u_{i,j} N^P_i(\xi) N^q_j(\eta) \]

with B-spline basis functions \( N^P_i \) of order \( p \).

- PoU, local support, non-negative
- Geometry-preserving refinement
- Generic extension to high order
- Operations can be expressed at SpMV's
My personal ‘top 3 features’ of IGA

2 ‘Meshing’ + design optimization becomes one global optimization problem

My personal ‘top 3 features’ of IGA

3. $C^{p-1}$-continuity enables direct simulation of higher-order PDEs

H.M. Verhelst, https://github.com/gismo/gsKLShear (v22.1)
My personal ‘top 3 features’ of IGA

3 $C^{p-1}$-continuity enables direct simulation of higher-order PDEs

My personal ‘top 3 features’ of IGA

3 $C^{p-1}$-continuity enables higher-order material point method

But ...

**IGA also has its challenges**

- automatic BRep-CAD-to-VRep-analysis workflows (we really don’t care)
- efficient $C^1$ multi-patch coupling (Delft, Linz, ...)
- efficient assembly of linear and multi-linear forms (INRIA, Pavia, ...)
- efficient solution of linear systems of equations (Delft, Linz, ...)
- ...
State of the art in IGA solvers

Direct solvers
- Performance study [Collier et al. 2012]
- Refined IGA [Garcia et al. 2018]

Preconditioning techniques
- Schwarz methods [da Veiga et al. 2012 & 2013]
- Sylvester equation [Sangalli & Tani 2016]
- Nonsymmetric systems [Tani 2017]
- BPX [Cho & Vásquez 2018]
- Fast diagonalization [Montardini et al. 2019]
- Space-time IGA [Hofer et al. 2019]
- Schwarz methods [Cho 2020]
- Directional splitting [Calo et al. 2021]
- Kronecker product [Loli et al. 2021]

p-multigrid techniques
- (Block-)ILUT smoother [Tielen et al. 2018, 2020]
- Multiplicative Schwarz smoother [de la Riva 2020]

h-multigrid techniques
- Full multigrid [Hofreither 2016]
- THB-splines [Hofreither et al. 2017]
- Symbol-based [Donatelli 2017]
- Boundary correction [Hofreither et al. 2017]
- Subspace corrected mass smoother [Takacs 2017]
- Multiplicative Schwarz smoother [de la Riva 2018]
- Biharmonic equation [Sogn et al. 2019]
- Immersed IGA [de Prenter et al. 2020]
- Bilaplacian equation [de la Riva et al. 2020]
- (Non-)conforming multipatch [Takacs 2020]

Transient problems
- Parallel splitting solvers [Puzyrev et al. 2019]
- Space-time solvers [Langer et al. 2016]
- Space-time solvers [Loli et al. 2020]
- Space-time least-squares [Montardini et al. 2020]
- MGRIT-IGA [Tielen et al. 2021]
State of the art in IGA solvers

Direct solvers
- Performance study [Collier et al. 2012]
- Refined IGA [Garcia et al. 2018]
## State of the art in IGA solvers

### Direct solvers
- Performance study [Collier et al. 2012]
- Refined IGA [Garcia et al. 2018]

### Preconditioning techniques
- Schwarz methods [da Veiga et al. 2012 & 2013]
- Sylvester equation [Sangalli & Tani 2016]
- Nonsymmetric systems [Tani 2017]
- BPX [Cho & Vásquez 2018]
- Fast diagonalization [Montardini et al. 2019]
- Space-time IGA [Hofer et al. 2019]
- Schwarz methods [Cho 2020]
- Directional splitting [Calo et al. 2021]
- Kronecker product [Loli et al. 2021]
State of the art in IGA solvers

<table>
<thead>
<tr>
<th>Direct solvers</th>
<th>\textit{h}-multigrid techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{itemize}</td>
<td>\begin{itemize}</td>
</tr>
<tr>
<td>\item Performance study [Collier \textit{et al.} 2012]</td>
<td>\item Full multigrid [Hofreither 2016]</td>
</tr>
<tr>
<td>\item Refined IGA [Garcia \textit{et al.} 2018]</td>
<td>\item THB-splines [Hofreither \textit{et al.} 2017]</td>
</tr>
<tr>
<td>\end{itemize}</td>
<td>\item Symbol-based [Donatelli 2017]</td>
</tr>
<tr>
<td>\end{itemize}</td>
<td>\item Boundary correction [Hofreither \textit{et al.} 2017]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Preconditioning techniques</th>
<th>\item Subspace corrected mass smoother [Takacs 2017]</th>
</tr>
</thead>
<tbody>
<tr>
<td>\item Schwarz methods [da Veiga \textit{et al.} 2012 &amp; 2013]</td>
<td>\item Multiplicative Schwarz smoother [de la Riva 2018]</td>
</tr>
<tr>
<td>\item Sylvester equation [Sangalli &amp; Tani 2016]</td>
<td>\item Biharmonic equation [Sogn \textit{et al.} 2019]</td>
</tr>
<tr>
<td>\item Nonsymmetric systems [Tani 2017]</td>
<td>\item Immersed IGA [de Prenter \textit{et al.} 2020]</td>
</tr>
<tr>
<td>\item BPX [Cho &amp; Vásquez 2018]</td>
<td>\item Bilaplacian equation [de la Riva \textit{et al.} 2020]</td>
</tr>
<tr>
<td>\item Fast diagonalization [Montardini \textit{et al.} 2019]</td>
<td>\item (Non-)conforming multipatch [Takacs 2020]</td>
</tr>
<tr>
<td>\item Space-time IGA [Hofer \textit{et al.} 2019]</td>
<td></td>
</tr>
</tbody>
</table>
### State of the art in IGA solvers

#### Direct solvers
- Performance study [Collier *et al.* 2012]
- Refined IGA [Garcia *et al.* 2018]

#### Preconditioning techniques
- Schwarz methods [da Veiga *et al.* 2012 & 2013]
- Sylvester equation [Sangalli & Tani 2016]
- Nonsymmetric systems [Tani 2017]
- BPX [Cho & Vásquez 2018]
- Fast diagonalization [Montardini *et al.* 2019]
- Space-time IGA [Hofer *et al.* 2019]
- Schwarz methods [Cho 2020]
- Directional splitting [Calo *et al.* 2021]
- Kronecker product [Loli *et al.* 2021]

#### $h$-multigrid techniques
- Full multigrid [Hofreither 2016]
- THB-splines [Hofreither *et al.* 2017]
- Symbol-based [Donatelli 2017]
- Boundary correction [Hofreither *et al.* 2017]
- Subspace corrected mass smoother [Takacs 2017]
- Multiplicative Schwarz smoother [de la Riva 2018]
- Biharmonic equation [Sogn *et al.* 2019]
- Immersed IGA [de Prenter *et al.* 2020]
- Bilaplacian equation [de la Riva *et al.* 2020]
- (Non-)conforming multipatch [Takacs 2020]

#### $p$-multigrid techniques
- (Block-)ILUT smoother [Tielen *et al.* 2018, 2020]
- Multiplicative Schwarz smoother [de la Riva 2020]
## State of the art in IGA solvers

### Direct solvers
- Performance study [Collier et al. 2012]
- Refined IGA [Garcia et al. 2018]

### Preconditioning techniques
- Schwarz methods [da Veiga et al. 2012 & 2013]
- Sylvester equation [Sangalli & Tani 2016]
- Nonsymmetric systems [Tani 2017]
- BPX [Cho & Vásquez 2018]
- Fast diagonalization [Montardini et al. 2019]
- Space-time IGA [Hofer et al. 2019]
- Schwarz methods [Cho 2020]
- Directional splitting [Calo et al. 2021]
- Kronecker product [Loli et al. 2021]

### p-multigrid techniques
- (Block-)ILUT smoother [Tielen et al. 2018, 2020]
- Multiplicative Schwarz smoother [de la Riva 2020]

### h-multigrid techniques
- Full multigrid [Hofreither 2016]
- THB-splines [Hofreither et al. 2017]
- Symbol-based [Donatelli 2017]
- Boundary correction [Hofreither et al. 2017]
- Subspace corrected mass smoother [Takacs 2017]
- Multiplicative Schwarz smoother [de la Riva 2018]
- Biharmonic equation [Sogn et al. 2019]
- Immersed IGA [de Prenter et al. 2020]
- Bilaplacian equation [de la Riva et al. 2020]
- (Non-)conforming multipatch [Takacs 2020]

### Transient problems
- Parallel splitting solvers [Puzyrev et al. 2019]
- Space-time solvers [Langer et al. 2016]
- Space-time solvers [Loli et al. 2020]
- Space-time least-squares [Montardini et al. 2020]
- MGRIT-IGA [Tielen et al. 2021]
State of the art in IGA solvers

### Direct solvers
- Performance study [Collier et al. 2012]
- Refined IGA [Garcia et al. 2018]

### Preconditioning techniques
- Schwarz methods [da Veiga et al. 2012 & 2013]
- Sylvester equation [Sangalli & Tani 2016]
- Nonsymmetric systems [Tani 2017]
- BPX [Cho & Vásquez 2018]
- Fast diagonalization [Montardini et al. 2019]
- Space-time IGA [Hofer et al. 2019]
- Schwarz methods [Cho 2020]
- Directional splitting [Calo et al. 2021]
- Kronecker product [Loli et al. 2021]

### h-multigrid techniques
- Full multigrid [Hofreither 2016]
- THB-splines [Hofreither et al. 2017]
- Symbol-based [Donatelli 2017]
- Boundary correction [Hofreither et al. 2017]
- Subspace corrected mass smoother [Takacs 2017]
- Multiplicative Schwarz smoother [de la Riva 2018]
- Biharmonic equation [Sogn et al. 2019]
- Immersed IGA [de Prenter et al. 2020]
- Bilaplacian equation [de la Riva et al. 2020]
- (Non-)conforming multipatch [Takacs 2020]

### p-multigrid techniques
- (Block-)ILUT smoother [Tielen et al. 2018, 2020]
- Multiplicative Schwarz smoother [de la Riva 2020]

### Transient problems
- Parallel splitting solvers [Puzyrev et al. 2019]
- Space-time solvers [Langer et al. 2016]
- Space-time solvers [Loli et al. 2020]
- Space-time least-squares [Montardini et al. 2020]
- MGRIT-IGA [Tielen et al. 2021]
Outline

1 Motivation and problem formulations

2 Part I: Multigrid methods for IGA
   - Introduction to $h$- and $p$-multigrid
   - ILUT smoother for single-patch IGA
   - Block-ILUT smoother for multi-patch IGA

3 Part II: Multigrid reduction in time (MGRIT)
   - Introduction to MGRIT
   - MGRIT-IGA

4 Conclusions
Model problems

Part I: Convection-diffusion-reaction equation (CDR-Eq)

\[-\nabla \cdot (\mathbb{D} \nabla u(x)) + \mathbf{v} \cdot \nabla u(x) + ru(x) = f\]
\[u(x) = g\]
\[x \in \Omega\]

Part II: Heat equation (Heat-Eq)

\[\partial_t u(x, t) - \kappa \Delta u(x, t) = f\]
\[u(x, t) = g\]
\[x \in \Gamma, t \in [0, T]\]
\[u(x, 0) = u^0(x)\]
\[x \in \Omega\]

\(d\)-dimensional connected Lipschitz domain \(\Omega \subset \mathbb{R}^d\), its boundary \(\Gamma = \partial \Omega\), load vector \(f \in L^2(\Omega)\), Dirichlet boundary conditions \(g\), diffusion tensor \(\mathbb{D}\) and coefficient \(\kappa\), resp., divergence-free velocity field \(\mathbf{v}\), source term \(r\), and \(u^0\) initial conditions
Variational formulation

**CDR-Eq:** Find $u \in H^1_g(\Omega)$ such that

$$a(w, u) = l(w) \quad \forall w \in H^1_0(\Omega)$$

**Heat-Eq:** Given $u^n \in H^1_g(\Omega)$ find $u^{n+1} \in H^1_g(\Omega)$ such that

$$\langle w, u^{n+1} \rangle + \Delta t k(w, u^{n+1}) = \langle w, u^n \rangle + l(w) \quad \forall w \in H^1_0(\Omega)$$
Variational formulation

**CDR-Eq:** Find $u \in \mathcal{H}_g^1(\Omega)$ such that

$$a(w, u) = l(w) \quad \forall w \in \mathcal{H}_0^1(\Omega)$$

**Heat-Eq:** Given $u^n \in \mathcal{H}_g^1(\Omega)$ find $u^{n+1} \in \mathcal{H}_g^1(\Omega)$ such that

$$\langle w, u_{n+1} \rangle + \Delta t k(w, u^{n+1}) = \langle w, u^n \rangle + l(w) \quad \forall w \in \mathcal{H}_0^1(\Omega)$$

with (bi-)linear forms defined as

$$a(w, u) := \int_{\Omega} \nabla w \cdot (\mathbb{D} \nabla u) + w (\mathbf{v} \cdot \nabla u + ru) \, dx$$

$$\langle w, u \rangle := \int_{\Omega} w u \, dx$$

$$k(w, u) := \kappa \int_{\Omega} \nabla u \cdot \nabla u \, dx$$

$$l(w) := \langle w, f \rangle$$
Algebraic equations

**CDR-Eq:** Find \( u_{h,p} \in \mathcal{V}_{h,p} \) such that

\[
A_{h,p} u_{h,p} = f_{h,p}
\]

**Heat-Eq:** Find \( u_{h,p}^{n+1} \in \mathcal{V}_{h,p} \) such that

\[
[M_{h,p} + \Delta t \ K_{h,p}] \ u_{h,p}^{n+1} = M_{h,p} u_{h,p}^n + f_{h,p}
\]
Algebraic equations

**CDR-Eq:** Find $u_{h,p} \in \mathcal{V}_{h,p}$ such that

$$A_{h,p} u_{h,p} = f_{h,p}$$

**Heat-Eq:** Find $u_{h,p}^{n+1} \in \mathcal{V}_{h,p}$ such that

$$[ M_{h,p} + \Delta t K_{h,p} ] u_{h,p}^{n+1} = M_{h,p} u_{h,p}^{n} + f_{h,p}$$

The unknown solution vector is given by

$$u_{h,p}^{n} = \sum_{j=1}^{N_{b}} u_{j}^{n} \varphi_{j}(x), \quad \text{where } u_{j}^{n} \text{ is the basis coefficient corresponding to } \varphi_{j} \in \mathcal{V}_{h,p}$$
Algebraic equations

**CDR-Eq:** Find $u_{h,p} \in \mathcal{V}_{h,p}$ such that

$$A_{h,p} u_{h,p} = f_{h,p}$$

**Heat-Eq:** Find $u_{h,p}^{n+1} \in \mathcal{V}_{h,p}$ such that

$$[M_{h,p} + \Delta t K_{h,p}] u_{h,p}^{n+1} = M_{h,p} u_{h,p}^{n} + f_{h,p}$$

The unknown solution vector is given by

$$u_{h,p}^{n} = \sum_{j=1}^{N_b} u_{j}^{n} \varphi_{j}(x), \quad \text{where } u_{j}^{n} \text{ is the basis coefficient corresponding to } \varphi_{j} \in \mathcal{V}_{h,p}$$

and the system matrices and right-hand side vector are defined as

$$A_{h,p} = \{a(\varphi_{i}, \varphi_{j})\}_{i,j}, \quad K_{h,p} = \{k(\varphi_{i}, \varphi_{j})\}_{i,j}, \quad M_{h,p} = \{\langle \varphi_{i}, \varphi_{j} \rangle\}_{i,j}, \quad f_{h,p} = \{l(\varphi_{i})\}_{i}$$
Ansatz spaces

**FEA:** element-wise ‘pull-back’

\[ \mathcal{V}_{h,p} = \{ v \in C^0(\bar{\Omega}) : v|_{T_k} \in \mathbb{Q}_p \circ F_{k}^{-1}, \forall T_k \in \mathcal{T}_h \} \]

\[ v|_{\Gamma} = 0 \}

with \( \mathbb{Q}_p \) the space of polynomials of degree \( p \) or less
Ansatz spaces

**FEA:** element-wise ‘pull-back’

\[ \mathcal{V}_{h,p} = \{ v \in C^0(\bar{\Omega}) : v|_{T_k} \in \mathbb{Q}_p \circ F_k^{-1}, \forall T_k \in \mathcal{T}_h \}
\]

\[ v|_{\Gamma} = 0 \}

with \( \mathbb{Q}_p \) the space of polynomials of degree \( p \) or less

**IGA:** patch-wise ‘pull-back’

\[ \mathcal{V}_{h,p} = \text{span}\{ \hat{\varphi}_j \circ F^{-1}_\ell \} \]

with \( \hat{\varphi}_j \) the \( j^{\text{th}} \) B-spline basis function
Ansatz spaces

**FEA:** element-wise ‘pull-back’

\[
\mathcal{V}_{h,p} = \{ v \in C^0(\bar{\Omega}) : v|_{T_k} \in \mathbb{Q}_p \circ F_k^{-1}, \forall T_k \in \mathcal{T}_h \\
v|_{\Gamma} = 0 \}
\]

with \( \mathbb{Q}_p \) the space of polynomials of degree \( p \) or less

**IGA:** patch-wise ‘pull-back’

\[
\mathcal{V}_{h,p} = \text{span}\{ \hat{\varphi}_j \circ F^{-1}_\ell \}
\]

with \( \hat{\varphi}_j \) the \( j^{th} \) B-spline basis function

Think of IGA patches as macro elements

B-spline illustration taken from: H.Nguyen-XuanaLoc et al., DOI: 10.1016/j.tafmec.2014.07.008
**Condition number**

<table>
<thead>
<tr>
<th></th>
<th>SEM-NI</th>
<th>IGA-$C^0$</th>
<th>IGA-$C^{p-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{K}(M)$</td>
<td>$\sim p^d$</td>
<td>$\sim p^{-d/2}4pd$</td>
<td>(h = 1/p)</td>
</tr>
<tr>
<td>$\mathcal{K}(K)$</td>
<td>$\sim h^{-2}p^3$</td>
<td>$\sim p^{-d/2}4dp$</td>
<td>(\sim (\frac{4}{p})^{d/h}4pd(hp)^{-d/2})</td>
</tr>
</tbody>
</table>

From: P. Gervasio, L. Dedè, O. Chanon, and A. Quarteroni, DOI: 10.1007/s10915-020-01204-1
Sparsity pattern: 2d single patch, $p = 1$

IGA-$C^0$

ref$_h = 0$

ref$_h = 1$

ref$_h = 2$

ref$_h = 3$

IGA-$C^{p-1}$
Sparsity pattern: 2d single patch, $p = 2$

ref$_h = 0$

ref$_h = 1$

ref$_h = 2$

ref$_h = 3$
Sparsity pattern: 2d single patch, $p = 3$

IGA-$C^0$

ref$_h = 0$

ref$_h = 1$

ref$_h = 2$

ref$_h = 3$

IGA-$C^{p-1}$
Sparsity pattern: 2d multi-patch IGA-$C^{p-1}$, $\text{ref}_h = 3$

Four-patch geometry with $C^0$ coupling of conforming degrees of freedom.
Sparsity pattern: 2d multi-patch IGA-$C^{p-1}$, $\text{ref}_h = 3$

$p = 1$

$p = 2$

$p = 3$

Four-patch geometry with $C^0$ coupling of conforming degrees of freedom.
Sketch of our solution strategy

- Coarsening in $p$ reduces the stencil but not so much the number of unknowns
  - $p$-multigrid with direct projection $\mathcal{V}_{h,p} \supset \mathcal{V}_{h,1}$
  - note that spaces are not nested ($\mathcal{V}_{h,p} \not\supset \mathcal{V}_{h,p-1} \not\supset \ldots$)
  - ILUT smoother at single-patch level

\[ \log_{10} h \]

\[ h = 1/p \]

\[ \sim (\frac{1}{h})^{d/h} 4^{d/p}(hp)^{-d/2} \]

\[ \sim e^{pd} \]

\[ \sim (\frac{1}{h})^{d/h} p^{-d/2} h^{-d/2} - 1.4 dp \]

\[ \sim p e^{dp} \]

\[ \sim h^{-2}p \]
Sketch of our solution strategy

- Coarsening in $p$ reduces the stencil but not so much the number of unknowns
  - $p$-multigrid with direct projection $\mathcal{V}_{h,p} \searrow \mathcal{V}_{h,1}$
  - note that spaces are not nested ($\mathcal{V}_{h,p} \not\supset \mathcal{V}_{h,p-1} \not\supset \ldots$)
  - ILUT smoother at single-patch level

- For $p = 1$, IGA-$C^0$ reduces to FEA with Lagrange finite elements
  - $h$-multigrid with established smoothers and coarse-grid solvers

\[ \text{ref}_{h} = 3 \quad \text{ref}_{h} = 2 \quad \text{ref}_{h} = 1 \quad \text{ref}_{h} = 0 \]
Sketch of our solution strategy

- Coarsening in $p$ reduces the stencil but not so much the number of unknowns
  - $p$-multigrid with **direct projection** $V_{h,p} \searrow V_{h,1}$
  - note that spaces are not nested ($V_{h,p} \not\supset V_{h,p-1} \not\supset \ldots$)
  - **ILUT smoother** at single-patch level

- For $p = 1$, IGA-$C^0$ reduces to FEA with Lagrange finite elements
  - $h$-multigrid with established smoothers and coarse-grid solvers

- Exploit the block structure of multi-patch topologies by using a **block-ILUT smoother**
Sketch of our solution strategy

• Coarsening in $p$ reduces the stencil but not so much the number of unknowns
  • $p$-multigrid with direct projection $V_{h,p} \downarrow V_{h,1}$
  • note that spaces are not nested ($V_{h,p} \not\supset V_{h,p-1} \not\supset \ldots$)
  • ILUT smoother at single-patch level

• For $p = 1$, IGA-$C^0$ reduces to FEA with Lagrange finite elements
  • $h$-multigrid with established smoothers and coarse-grid solvers

• Exploit the block structure of multi-patch topologies by using a block-ILUT smoother

• robust with respect to $h$, $p$, $N_p$, and ‘the PDE’
• computational efficient throughout all problem sizes
• applicable to locally refined THB-splines
• good spatial solver for transient problems (Part II)
The complete multigrid cycle

IGA-$C^{p-1}$

1. $p = 3$
2. $p = 2$
3. $p = 1$

IGA-$C^0$

1. $p = 1$
2. $p = 1$
3. $p = 1$

$p$-multigrid

$h$-multigrid

▲ (Block-)ILUT  ● Gauss-Seidel  ■ direct solve
The complete multigrid algorithm – the outer $p$-multigrid part

1. Starting from $u^{(0,0)}_{h,p}$ apply $\nu_1$ pre-smoothing steps:

$$u^{(0,m)}_{h,p} := u^{(0,m-1)}_{h,p} + S_{h,p} \left( f_{h,p} - A_{h,p} u^{(0,m-1)}_{h,p} \right), \quad m = 0, 1, \ldots, \nu_1$$
The complete multigrid algorithm – the outer \( p \)-multigrid part

1. Starting from \( u_{h,p}^{(0,0)} \) apply \( \nu_1 \) pre-smoothing steps:

\[
u_{h,p}^{(0,m)} := u_{h,p}^{(0,0)} + S_{h,p} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,m-1)} \right), \quad m = 0, 1, \ldots, \nu_1
\]

2. Restrict the residual onto \( V_{h,1} \):

\[
    r_{h,1} = I_{h,p}^{h,1} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,\nu_1)} \right), \quad I_{h,p}^{h,1} := M_{h,1}^{-1} M_{h,p,1}
\]

with \( M_{h,p,1} = \{(\varphi_i, \psi_j)\}_{i,j} \), where \( \varphi_i \in V_{h,p} \) and \( \psi_j \in V_{h,1} \)
The complete multigrid algorithm – the outer $p$-multigrid part

1. Starting from $u_{h,p}^{(0,0)}$ apply $\nu_1$ pre-smoothing steps:

$$u_{h,p}^{(0,m)} := u_{h,p}^{(0,m-1)} + S_{h,p} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,m-1)} \right), \quad m = 0, 1, \ldots, \nu_1$$

2. Restrict the residual onto $V_{h,1}$:

$$r_{h,1} = I_{h,p}^{h,1} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,\nu_1)} \right), \quad I_{h,p}^{h,1} := M_{h,1}^{-1} M_{h,p,1}$$

with $M_{h,p,1} = \{(\varphi_i, \psi_j)\}_{i,j}$, where $\varphi_i \in V_{h,p}$ and $\psi_j \in V_{h,1}$

3. Solve the residual equation with an $h$-multigrid method:

$$A_{h,1} e_{h,1} = r_{h,1}$$
The complete multigrid algorithm – the outer $p$-multigrid part

1. Starting from $u_{h,p}^{(0,0)}$ apply $\nu_1$ pre-smoothing steps:
   
   \[
   u_{h,p}^{(0,m)} := u_{h,p}^{(0,m-1)} + S_{h,p} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,m-1)} \right), \quad m = 0, 1, \ldots, \nu_1
   \]

2. Restrict the residual onto $\mathcal{V}_{h,1}$:
   
   \[
   r_{h,1} = I_{h,p}^{h,1} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,\nu_1)} \right), \quad I_{h,p}^{h,1} := M_{h,1}^{-1} M_{h,p,1}
   \]
   with $M_{h,p,1} = \{(\varphi_i, \psi_j)\}_{i,j}$, where $\varphi_i \in \mathcal{V}_{h,p}$ and $\psi_j \in \mathcal{V}_{h,1}$

3. Solve the residual equation with an $h$-multigrid method:
   
   \[
   A_{h,1} e_{h,1} = r_{h,1}
   \]

4. Project the error onto $\mathcal{V}_{h,p}$ and update the solution:
   
   \[
   u_{h,p}^{(0,\nu_1)} := u_{h,p}^{(0,\nu_1)} + I_{h,1}^{h,p} (e_{h,1}), \quad I_{h,1}^{h,p} := M_{h,p}^{-1} M_{h,1,p}
   \]
The complete multigrid algorithm – the outer $p$-multigrid part

1. Starting from $u_{h,p}^{(0,0)}$ apply $\nu_1$ pre-smoothing steps:

   \[ u_{h,p}^{(0,m)} := u_{h,p}^{(0,m-1)} + S_{h,p} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,m-1)} \right), \quad m = 0, 1, \ldots, \nu_1 \]

2. Restrict the residual onto $\mathcal{V}_{h,1}$:

   \[ r_{h,1} = I_{h,1}^{h,1} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,\nu_1)} \right), \quad I_{h,p}^{h,1} := M_{h,1}^{-1} M_{h,p,1} \]

   with $M_{h,p,1} = \{(\varphi_i, \psi_j)\}_{i,j}$, where $\varphi_i \in \mathcal{V}_{h,p}$ and $\psi_j \in \mathcal{V}_{h,1}$

3. Solve the residual equation with an $h$-multigrid method:

   \[ A_{h,1} e_{h,1} = r_{h,1} \]

4. Project the error onto $\mathcal{V}_{h,p}$ and update the solution:

   \[ u_{h,p}^{(0,\nu_1)} := u_{h,p}^{(0,\nu_1)} + I_{h,1}^{h,p} (e_{h,1}), \quad I_{h,1}^{h,p} := M_{h,p}^{-1} M_{h,1,p} \]

5. Apply $\nu_2$ post-smoothing steps as in 1.

   1. to obtain $u_{h,p}^{(1,0)} := u_{h,p}^{(0,\nu_1+\nu_2)}$ and repeat steps 1.–5. until $\| r_{h,p}^{(k)} \| < \text{tol} \| r_{h,p}^{(0)} \|$ for some tolerance parameter $\text{tol}$. 
The complete multigrid algorithm – the outer $p$-multigrid part

1. Starting from $u_{h,p}^{(0,0)}$ apply $\nu_1$ pre-smoothing steps:

   \[ u_{h,p}^{(0,m)} := u_{h,p}^{(0,m-1)} + S_{h,p} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,m-1)} \right), \quad m = 0, 1, \ldots, \nu_1 \]

2. Restrict the residual onto $V_{h,1}$:

   \[ r_{h,1} = I_{h,1}^{h,1} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,\nu_1)} \right), \quad I_{h,1}^{h,1} := M_{h,1}^{-1} M_{h,p,1} \quad \text{mass lumping} \]

   with $M_{h,p,1} = \{(\varphi_i, \psi_j)\}_{i,j}$, where $\varphi_i \in V_{h,p}$ and $\psi_j \in V_{h,1}$

3. Solve the residual equation with an $h$-multigrid method:

   \[ A_{h,1} e_{h,1} = r_{h,1} \]

4. Project the error onto $V_{h,p}$ and update the solution:

   \[ u_{h,p}^{(0,\nu_1)} := u_{h,p}^{(0,\nu_1)} + I_{h,1}^{h,p} (e_{h,1}), \quad I_{h,1}^{h,p} := M_{h,p,1}^{-1} M_{h,1,p} \quad \text{mass lumping (B-splines!)} \]

5. Apply $\nu_2$ post-smoothing steps as in 1. to obtain $u_{h,p}^{(1,0)} := u_{h,p}^{(0,\nu_1+\nu_2)}$ and repeat steps 1.–5. until $\| r_{h,p}^{(k)} \| < \text{tol} \| r_{h,p}^{(0)} \|$ for some tolerance parameter tol.
The complete multigrid algorithm – the outer $p$-multigrid part

1. Starting from $u_{h,p}^{(0,0)}$ apply $\nu_1$ pre-smoothing steps:

   $$ u_{h,p}^{(0,m)} := u_{h,p}^{(0,m-1)} + S_{h,p} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,m-1)} \right), \quad m = 0, 1, \ldots, \nu_1 $$

2. Restrict the residual onto $\mathcal{V}_{h,1}$:

   $$ r_{h,1} = I_{h,p}^{h,1} \left( f_{h,p} - A_{h,p} u_{h,p}^{(0,\nu_1)} \right), \quad I_{h,p}^{h,1} := M_{h,1}^{-1} M_{h,p,1} \quad \text{mass lumping} $$

   with $M_{h,p,1} = \{(\varphi_i, \psi_j)\}_{i,j}$, where $\varphi_i \in \mathcal{V}_{h,p}$ and $\psi_j \in \mathcal{V}_{h,1}$

3. Solve the residual equation with an $h$-multigrid method:

   $$ A_{h,1} e_{h,1} = r_{h,1} $$

4. Project the error onto $\mathcal{V}_{h,p}$ and update the solution:

   $$ u_{h,p}^{(0,\nu_1)} := u_{h,p}^{(0,\nu_1)} + I_{h,p}^{h,1} (e_{h,1}), \quad I_{h,p}^{h,1} := M_{h,p}^{-1} M_{h,1,p} \quad \text{mass lumping (B-splines!)} $$

5. Apply $\nu_2$ post-smoothing steps as in 1. to obtain $u_{h,p}^{(1,0)} := u_{h,p}^{(0,\nu_1+\nu_2)}$ and repeat steps 1.–5. until $\| r_{h,p}^{(k)} \| < \text{tol} \| r_{h,p}^{(0)} \|$ for some tolerance parameter tol.
The complete multigrid algorithm – the inner $h$-multigrid part

3.1. Starting from $u_{h,1}^{(k,0)}$ apply $\nu_1$ pre-smoothing steps:

$$u_{h,1}^{(k,m)} := u_{h,1}^{(k,m-1)} + S_{h,1} \left( f_{h,1} - A_{h,1} u_{h,1}^{(k,m-1)} \right), \quad m = 0, 1, \ldots, \nu_1$$

3.2. Restrict the residual onto $V_{2h,1}$:

$$r_{2h,1} = I_{h,1}^{2h,1} \left( f_{h,1} - A_{h,1} u_{h,1}^{(k,\nu_1)} \right), \quad I_{h,1}^{2h,1} \text{ linear interpolation}$$

3.3. Solve the residual equation by applying $h$-multigrid recursively or the coarse-grid solver:

$$A_{2h,1} e_{2h,1} = r_{2h,1}$$

3.4. Project the error onto $V_{h,1}$ and update the solution:

$$u_{h,1}^{(k,\nu_1)} := u_{h,1}^{(k,\nu_1)} + I_{2h,1}^{h,1} (e_{2h,1}), \quad I_{2h,1}^{h,1} := \frac{1}{2} \left( I_{h,1}^{2h,1} \right)^\top$$

3.5. Apply $\nu_2$ post-smoothing steps as in 3.1. to obtain $u_{h,1}^{(k+1,0)} := u_{h,1}^{(k,\nu_1+\nu_2)}$ and repeat steps 3.1.–3.5. according to the $h$-multigrid cycle (V- or W-cycle).
## Multigrid components

<table>
<thead>
<tr>
<th></th>
<th>$h$-multigrid</th>
<th>$p$-multigrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>restriction operator</td>
<td>$I_{h,1}^{2h,1}$ linear interpolation</td>
<td>$I_{h,1}^{h,p} := M_{h,p}^{-1} M_{h,1,p}$</td>
</tr>
<tr>
<td>prolongation operator</td>
<td>$I_{2h,1}^{h,1} := \frac{1}{2} \left( I_{h,1}^{2h,1} \right)^\top$</td>
<td>$I_{h,p}^{h,1} := M_{h,1}^{-1} M_{h,p,1}$</td>
</tr>
</tbody>
</table>
**Multigrid components**

<table>
<thead>
<tr>
<th></th>
<th>(h)-multigrid</th>
<th>(p)-multigrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>restriction operator</td>
<td>(I_{h,1}^{2h,1}) linear interpolation</td>
<td>(I_{h,1}^{h,p} := M_{h,p}^{-1} M_{h,1,p})</td>
</tr>
<tr>
<td>prolongation operator</td>
<td>(I_{2h,1}^{h,1} := \frac{1}{2} \left(I_{h,1}^{2h,1}\right)^\top)</td>
<td>(I_{h,p}^{h,1} := M_{h,1}^{-1} M_{h,p,1})</td>
</tr>
<tr>
<td>smoothing operator</td>
<td>incomplete LU factorization of (A_{h,p} \approx L_{h,p} U_{h,p}), whereby all elements smaller than (10^{-13}) are dropped and the amount of non-zero entries per row are kept constant</td>
<td></td>
</tr>
</tbody>
</table>

---

Y. Saad, ILUT: A dual threshold incomplete LU factorization, DOI: 10.1002/nla.1680010405
## Multigrid components

<table>
<thead>
<tr>
<th></th>
<th>$h$-multigrid</th>
<th>$p$-multigrid</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>restriction operator</strong></td>
<td>$I_{2h,1}^{h,1}$ linear interpolation</td>
<td>$I_{h,1}^{h,p} := M_{h,p}^{-1} M_{h,1,p}$</td>
</tr>
<tr>
<td><strong>prolongation operator</strong></td>
<td>$I_{2h,1}^{h,1} := \frac{1}{2} \left( I_{h,1}^{2h,1} \right)^\top$</td>
<td>$I_{h,1}^{h,p} := M_{h,1}^{-1} M_{h,p,1}$</td>
</tr>
<tr>
<td><strong>smoothing operator</strong></td>
<td>incomplete LU factorization of $A_{h,p} \approx L_{h,p} U_{h,p}$, whereby all elements smaller than $10^{-13}$ are dropped and the amount of non-zero entries per row are kept constant</td>
<td></td>
</tr>
<tr>
<td><strong>$A_{h,p}$ operator</strong></td>
<td></td>
<td>rediscretization</td>
</tr>
</tbody>
</table>

Y. Saad, ILUT: A dual threshold incomplete LU factorization, DOI: 10.1002/nla.1680010405
Spectrum of the iteration matrix: Poisson on quarter annulus, $p = 2$

Spectrum of the iteration matrix: Poisson on quarter annulus, $p = 3$

Spectrum of the iteration matrix: Poisson on quarter annulus, $p = 4$

**Numerical examples**

**#1:** Poisson’s equation on a quarter annulus domain with radii 1 and 2

<table>
<thead>
<tr>
<th></th>
<th>$p = 2$</th>
<th></th>
<th>$p = 3$</th>
<th></th>
<th>$p = 4$</th>
<th></th>
<th>$p = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ILUT</td>
<td>GS</td>
<td>ILUT</td>
<td>GS</td>
<td>ILUT</td>
<td>GS</td>
<td>ILUT</td>
</tr>
<tr>
<td>$h = 2^{-6}$</td>
<td>4</td>
<td>30</td>
<td>3</td>
<td>62</td>
<td>3</td>
<td>176</td>
<td>3</td>
</tr>
<tr>
<td>$h = 2^{-7}$</td>
<td>4</td>
<td>29</td>
<td>3</td>
<td>61</td>
<td>3</td>
<td>172</td>
<td>3</td>
</tr>
<tr>
<td>$h = 2^{-8}$</td>
<td>5</td>
<td>30</td>
<td>3</td>
<td>60</td>
<td>3</td>
<td>163</td>
<td>3</td>
</tr>
<tr>
<td>$h = 2^{-9}$</td>
<td>5</td>
<td>32</td>
<td>3</td>
<td>61</td>
<td>3</td>
<td>163</td>
<td>3</td>
</tr>
</tbody>
</table>

---

Numerical examples

**#2:** CDR equation with \( \mathbb{D} = \begin{pmatrix} 1.2 & -0.7 \\ -0.4 & 0.9 \end{pmatrix} \), \( \mathbf{v} = (0.4, -0.2)^\top \), and \( r = 0.3 \) on the unit square domain

<table>
<thead>
<tr>
<th>( h = 2^{-6} )</th>
<th>( p = 2 )</th>
<th>( p = 3 )</th>
<th>( p = 4 )</th>
<th>( p = 5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILUT</td>
<td>GS</td>
<td>ILUT</td>
<td>GS</td>
<td>ILUT</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>3</td>
</tr>
</tbody>
</table>
Computational efficiency: $p$- vs. $h$-multigrid

Comparison with $h$-multigrid method with subspace corrected mass smoother [Takacs, 2017]
Computational efficiency: \( p \)- vs. \( h \)-multigrid

Comparison with \( h \)-multigrid method with subspace corrected mass smoother [Takacs, 2017]
Computational efficiency: $\{h,p\}$-multigrid + $\{ILUT,SCMS\}$-smoother
Numerical examples: THB splines

#3: Poisson’s equation on the unit square domain

<table>
<thead>
<tr>
<th></th>
<th>$p = 2$</th>
<th>$p = 3$</th>
<th>$p = 4$</th>
<th>$p = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ILUT</td>
<td>GS</td>
<td>ILUT</td>
<td>GS</td>
</tr>
<tr>
<td>$h = 2^{-4}$</td>
<td>6 17 8 47</td>
<td>7 177 10 1033</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$h = 2^{-5}$</td>
<td>6 16 7 44</td>
<td>8 182 7 923</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$h = 2^{-6}$</td>
<td>6 17 5 43</td>
<td>6 201 12 1009</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Block ILUT

Exact LU decomposition of the block matrix $A$

$$
\begin{bmatrix}
A_{11} & \ldots & A_{1\Gamma_1} \\
\vdots & \ddots & \vdots \\
A_{Np_1} & \ldots & A_{Np \Gamma_1}
\end{bmatrix}
= \begin{bmatrix}
L_1 & \ldots & L_{Np} \\
\vdots & \ddots & \vdots \\
B_1 & \ldots & B_{Np}
\end{bmatrix}
\begin{bmatrix}
U_1 & \ldots & C_1 \\
\vdots & \ddots & \vdots \\
U_{Np} & \ldots & C_{Np}
\end{bmatrix},
$$

with

$$
A_{\ell \ell} = L_{\ell} U_{\ell}, \quad B_\ell = A_{\ell \Gamma} U_{\ell}^{-1}, \quad C_\ell = L_{\ell}^{-1} A_{\Gamma \ell}, \quad S = A_{\Gamma \Gamma} - \sum_{\ell=1}^{N_p} B_\ell C_\ell
$$
Block ILUT

Approximate LU decomposition of the block matrix $A$

\[
\begin{bmatrix}
A_{11} & A_{1\Gamma} \\
\vdots & \ddots & \ddots \\
A_{1\Gamma} & \cdots & A_{N_p \Gamma} & A_{N_p \Gamma}
\end{bmatrix}
\approx
\begin{bmatrix}
\tilde{L}_1 \\
\vdots \\
\tilde{L}_{N_p}
\end{bmatrix}
\begin{bmatrix}
\tilde{U}_1 & \cdots & \tilde{C}_1 \\
\vdots & \ddots & \vdots \\
\tilde{U}_{N_p} & \cdots & \tilde{C}_{N_p}
\end{bmatrix}
\]

with

\[
A_{\ell\ell} = L_\ell U_\ell, \quad B_\ell = A_{\ell\Gamma} U_\ell^{-1}, \quad C_\ell = L_\ell^{-1} A_{\ell\Gamma}, \quad S = A_{\Gamma\Gamma} - \sum_{\ell=1}^{N_p} B_\ell C_\ell
\]

Let us replace $L_\ell$ and $U_\ell$ by their (local) ILUT factorizations (compute in parallel!)

\[
A_{\ell\ell} \approx \tilde{L}_\ell \tilde{U}_\ell, \quad \tilde{B}_\ell = A_{\ell\Gamma} \tilde{U}_\ell^{-1}, \quad \tilde{C}_\ell = \tilde{L}_\ell^{-1} A_{\ell\Gamma}, \quad \tilde{S} = A_{\Gamma\Gamma} - \sum_{\ell=1}^{N_p} \tilde{B}_\ell \tilde{C}_\ell
\]
**Numerical examples:**  *Block-ILUT vs. global ILUT*

#1: Poisson’s equation on the quarter annulus domain with radii 1 and 2

<table>
<thead>
<tr>
<th></th>
<th>$p = 2$</th>
<th></th>
<th>$p = 3$</th>
<th></th>
<th>$p = 4$</th>
<th></th>
<th>$p = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># patches</td>
<td></td>
<td># patches</td>
<td></td>
<td># patches</td>
<td></td>
<td># patches</td>
</tr>
<tr>
<td>$h = 2^{-5}$</td>
<td>4 16 64</td>
<td>$h = 2^{-6}$</td>
<td>4 16 64</td>
<td>$h = 2^{-7}$</td>
<td>4 16 64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3(5) 4(7) 4(9)</td>
<td>3(5) 3(7) 4(11)</td>
<td>2(4) 2(6) 4(11)</td>
<td>2(4) 2(6) 4(11)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3(5) 3(5) 4(7)</td>
<td>3(5) 3(7) 4(10)</td>
<td>3(6) 2(7) 3(11)</td>
<td>3(5) 3(7) 3(10)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3(5) 3(5) 3(5)</td>
<td>3(5) 3(6) 3(8)</td>
<td>3(5) 2(6) 3(10)</td>
<td>3(5) 3(7) 3(10)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Numbers in parentheses correspond to global ILUT

---

Numerical examples: \textit{Block-ILUT vs. global ILUT}

\#2: CDR equation with $D = \begin{pmatrix} 1.2 & -0.7 \\ -0.4 & 0.9 \end{pmatrix}$, $v = (0.4, -0.2)^\top$, and $r = 0.3$ on the unit square domain

\begin{tabular}{c|ccc|ccc|ccc|ccc|ccc|ccc|ccc|ccc|ccc|}
 & \multicolumn{3}{c|}{\# patches} & \multicolumn{3}{c|}{\# patches} & \multicolumn{3}{c|}{\# patches} & \multicolumn{3}{c|}{\# patches} & \multicolumn{3}{c|}{\# patches} & \multicolumn{3}{c|}{\# patches} & \multicolumn{3}{c|}{\# patches} \\
\hline
$h = 2^{-5}$ & 4(6) & 4(8) & 7(11) & 3(6) & 3(8) & 5(15) & 2(6) & 3(8) & 5(15) & 2(5) & 2(7) & 4(14) & 3(6) & 3(8) & 5(15) & 2(6) & 3(8) & 5(15) & 2(5) & 2(7) & 4(14) \\
$h = 2^{-6}$ & 4(6) & 4(7) & 5(8) & 3(6) & 3(8) & 4(10) & 3(7) & 3(9) & 4(13) & 3(7) & 3(8) & 3(13) & 3(6) & 3(8) & 4(10) & 3(7) & 3(9) & 4(13) & 3(7) & 3(8) & 3(13) \\
$h = 2^{-7}$ & 4(6) & 4(6) & 4(7) & 3(6) & 3(7) & 3(8) & 2(7) & 3(7) & 3(10) & 4(6) & 3(8) & 3(12) & 4(6) & 3(8) & 3(12) & 4(6) & 3(8) & 3(12) & 4(6) & 3(8) & 3(12) \\
\end{tabular}

Numbers in parentheses correspond to global ILUT

---

Numerical examples: *Block-ILUT vs. global ILUT*

**#4: Poisson’s equation on the Yeti footprint**

<table>
<thead>
<tr>
<th></th>
<th>$p = 2$</th>
<th>$p = 3$</th>
<th>$p = 4$</th>
<th>$p = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>block</td>
<td>global</td>
<td>block</td>
<td>global</td>
</tr>
<tr>
<td>$h = 2^{-3}$</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>$h = 2^{-4}$</td>
<td>4</td>
<td>8</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>$h = 2^{-5}$</td>
<td>4</td>
<td>8</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

---

Outline

1 Motivation and problem formulations

2 Part I: Multigrid methods for IGA
   - Introduction to $h$- and $p$-multigrid
   - ILUT smoother for single-patch IGA
   - Block-ILUT smoother for multi-patch IGA

3 Part II: Multigrid reduction in time (MGRIT)
   - Introduction to MGRIT
   - MGRIT-IGA

4 Conclusions

- robust with respect to $h$, $p$, $N_p$, and ‘the PDE’
- computational efficient throughout all problem sizes
- applicable to locally refined THB-splines
- Good spatial solver for transient problems (Part II)
Part II: Multigrid reduction in time (MGRIT)

\[ \Delta t_C = m \Delta t_F \]

Sketch of the MGRIT algorithm

**Heat-Eq:** Find $u_{h,p}^{n+1} \in \mathcal{V}_{h,p}$ such that

$$[M_{h,p} + \Delta t F K_{h,p}] u_{h,p}^{n+1} = M_{h,p} u_{h,p}^n + f_{h,p}$$
Sketch of the MGRIT algorithm

Heat-Eq: Find $u_{h,p}^{n+1} \in \mathcal{V}_{h,p}$ such that

$$\left[ M_{h,p} + \Delta t_F K_{h,p} \right] u_{h,p}^{n+1} = M_{h,p} u_{h,p}^n + f_{h,p}$$

Writing out the above two-level scheme for all time levels yields

$$A_{h,p} u_{h,p} = \begin{bmatrix} I_{h,p} \\ -\Psi_{h,p} M_{h,p} & I_{h,p} \\ \vdots & \vdots & \ddots \\ -\Psi_{h,p} M_{h,p} & I_{h,p} \end{bmatrix} \begin{bmatrix} u_{h,p}^0 \\ u_{h,p}^1 \\ \vdots \\ u_{h,p}^{N_t} \end{bmatrix} = \Delta t_F \begin{bmatrix} \Psi_{h,p} f_{h,p} \\ \Psi_{h,p} f_{h,p} \\ \vdots \\ \Psi_{h,p} f_{h,p} \end{bmatrix}$$

with

$$\Psi_{h,p} = \left[ M_{h,p} + \Delta t_F K_{h,p} \right]^{-1}$$

---

Sketch of the MGRIT algorithm, cont’d

Reordering of $A_{h,p}$ into (F)ine and (C)oarse time levels yields

$$
\begin{bmatrix}
A_{FF} & A_{FC} \\
A_{CF} & A_{CC}
\end{bmatrix} =
\begin{bmatrix}
I_F & 0 \\
A_{CF}A_{FF}^{-1} & I_C
\end{bmatrix}
\begin{bmatrix}
A_{FF} & 0 \\
0 & S
\end{bmatrix}
\begin{bmatrix}
I_F & A_{FF}^{-1}A_{FC} \\
0 & I_C
\end{bmatrix}
$$

---

Sketch of the MGRIT algorithm, cont’d

Reordering of $A_{h,p}$ into (F)ine and (C)oarse time levels yields

$$
\begin{bmatrix}
A_{FF} & A_{FC} \\
A_{CF} & A_{CC}
\end{bmatrix} =
\begin{bmatrix}
I_F & 0 \\
A_{CF} A_{FF}^{-1} & I_C
\end{bmatrix}
\begin{bmatrix}
A_{FF} & 0 \\
0 & S
\end{bmatrix}
\begin{bmatrix}
I_F & A_{FF}^{-1} A_{FC} \\
0 & I_C
\end{bmatrix}
$$

with block-diagonal fine-level system matrix

$$A_{FF} = I_{N_t/m, N_t/m} \bigotimes \begin{pmatrix}
I_{h,p} \\
-\Psi_{h,p} M_{h,p} & I_{h,p} \\
\vdots & \ddots & \ddots \\
-\Psi_{h,p} M_{h,p} & \cdots & I_{h,p}
\end{pmatrix}
\begin{array}{c}
\scriptstyle m \times m \text{ blocks}
\end{array}$$

---

Sketch of the MGRIT algorithm, cont'd

Reordering of $A_{h,p}$ into (F)ine and (C)oarse time levels yields

$$
\begin{bmatrix}
A_{FF} & A_{FC} \\
A_{CF} & A_{CC}
\end{bmatrix}
= 
\begin{bmatrix}
I_F & 0 \\
A_{CF}A_{FF}^{-1} & I_C
\end{bmatrix}
\begin{bmatrix}
A_{FF} & 0 \\
0 & S
\end{bmatrix}
\begin{bmatrix}
I_F & A_{FF}^{-1}A_{FC} \\
0 & I_C
\end{bmatrix}
$$

with block-diagonal fine-level system matrix $A_{FF}$

$$A_{FF} = I_{N_t/m,N_t/m} \otimes 
\begin{pmatrix}
I_{h,p} \\
-\Psi_{h,p} M_{h,p} & I_{h,p} \\
& \ddots \\
& & -\Psi_{h,p} M_{h,p} & I_{h,p}
\end{pmatrix}
$$

$m \times m$ blocks

and the Schur complement $S = A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}$

---

Sketch of the MGRIT algorithm, cont’d

Approximate the Schur complement

$$S = \begin{bmatrix}
I & -\left(\Psi_{h,p} M_{h,p}\right)^m & \cdots & -\left(\Psi_{h,p} M_{h,p}\right)^m \\
-(\Psi_{h,p} M_{h,p})^m & I & \cdots & -\left(\Psi_{h,p} M_{h,p}\right)^m \\
\vdots & \ddots & \ddots & \vdots \\
-(\Psi_{h,p} M_{h,p})^m & \cdots & -(\Psi_{h,p} M_{h,p})^m & I
\end{bmatrix} \approx \begin{bmatrix}
I & -\Phi_{h,p} M_{h,p} & \cdots & \cdots \\
-\Phi_{h,p} M_{h,p} & I & \cdots & -\Phi_{h,p} M_{h,p} \\
\vdots & \ddots & \ddots & \vdots \\
\cdots & \cdots & -\Phi_{h,p} M_{h,p} & I
\end{bmatrix}$$

with coarse integrator

$$\Phi_{h,p} = [M_{h,p} + \Delta t_C K_{h,p}]^{-1}$$
The MGRIT-IGA V-cycle

\[

dl = 0 \quad \bullet \quad \Delta t
\]
\[
dl = 1 \quad \Delta tm
\]
\[
dl = 2 \quad \Delta tm^2
\]
\[
dl = 3 \quad \Delta tm^3
\]
\[
dl = 4 \quad \Delta tm^4
\]

● relaxation  ■ exact solve  \↓ restriction  \↑ interpolation
MGRIT-IGA implementation

**G+Smo:** Geometry plus Simulation Modules
- open-source cross-platform IGA library written in C++
- dimension-independent code development using templates
- building on Eigen C++ library for linear algebra

**XBraid:** Parallel Multigrid in Time
- open-source implementation of the optimal-scaling multigrid solver in MPI/C with C++ interface
- extendable by overloading callback functions

Try it yourself

https://github.com/gismo/gismo/tree/xbraid/extensions/gsXBraid
Numerical examples: *Strong scaling of MGRIT-IGA*

**#5:** Heat-Eq with $h = 2^{-6}$ spatial resolution solved for $N_t = 10,000$ time steps with backward Euler method on 128 Xeon Gold 6130 CPUs (2.10GHz, 96GB, 16 cores)

![Graph showing CPU times for different parallel configurations](image-url)

Numerical examples: Speed-up of MGRIT-IGA

#5: Heat-Eq with $h = 2^{-6}$ spatial resolution solved for $N_t = 10,000$ time steps with backward Euler method on 128 Xeon Gold 6130 CPUs (2.10GHz, 96GB, 16 cores)

---

Numerical examples: Weak scaling of MGRIT-IGA

#5: Heat-Eq with $h = 2^{-6}$ spatial resolution solved for $N_t = \text{cores}/64 \cdot 1.000$ time steps with backward Euler method on 128 Xeon Gold 6130 CPUs (2.10GHz, 96GB, 16 cores)

---

Do we really need $p$-multigrid or would a standard solver be good enough?
Do we really need $p$-multigrid or would a standard solver be good enough? No!
Do we really need $p$-multigrid or would a standard solver be good enough? No!

**CPU times**

- **CG solver on 3 × 2 cores**
  - $p = 2$: $N_t = 250$: 11, $N_t = 500$: 20, $N_t = 1000$: 40, $N_t = 2000$: 76
  - $p = 3$: $N_t = 250$: 30, $N_t = 500$: 59, $N_t = 1000$: 123, $N_t = 2000$: 250
  - $p = 4$: $N_t = 250$: 95, $N_t = 500$: 193, $N_t = 1000$: 421, $N_t = 2000$: 855
  - $p = 5$: $N_t = 250$: 292, $N_t = 500$: 590, $N_t = 1000$: 1312, $N_t = 2000$: 2743

- **$p$-mg-ILUT on 3 × 2 cores**
  - $p = 2$: $N_t = 250$: 26, $N_t = 500$: 27, $N_t = 1000$: 48, $N_t = 2000$: 82
  - $p = 4$: $N_t = 250$: 45, $N_t = 500$: 62, $N_t = 1000$: 92, $N_t = 2000$: 146
  - $p = 5$: $N_t = 250$: 79, $N_t = 500$: 109, $N_t = 1000$: 142, $N_t = 2000$: 190
MGRIT-IGA + $p$-multigrid with (block-)ILUT smoother

- robust with respect to $h$, $p$, $N_p$, and ‘the PDE’
- computational efficient throughout all problem sizes
- applicable to locally refined THB-splines
- good strong and weak scaling in no. of cores and $N_t$
Conclusion

MGRIT-IGA + $p$-multigrid with (block-)ILUT smoother

- robust with respect to $h$, $p$, $N_p$, and ‘the PDE’
- computational efficient throughout all problem sizes
- applicable to locally refined THB-splines
- good strong and weak scaling in no. of cores and $N_t$

What’s next?

- MGRIT-IGA with THB-splines and adaptive refinement in time
- extension to nonlinear PDEs and higher-order time integrators
Further reading


R. Tielen, M. Möller and C. Vuik: *Combining p-multigrid and multigrid reduced in time methods to obtain a scalable solver for Isogeometric Analysis*, arXiv:2107.05337

Thank you for your attention!