Design-through-analysis

Matthias Möller

Department of Applied Mathematics
Delft University of Technology, NL

Fluids under Control

Prague Summer School 2021, August 23 – 27

About me

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

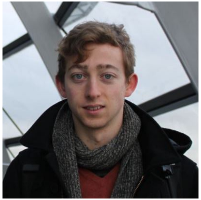
Research interests

- 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

The IGA team

Jochen Hinz (EPFL)

Roel Tielen (TUD)



Hugo Verhelst (TUD)

Andrzej Jaeschke (Łódź)

Collaborations

Elgeti/Helmig (RWTH Aachen), Mantzaflaris (INRIA), Gauger/Sagebaum (TU K'lautern), Brümmer/Utri (TU Dortmund), Jüttler (JKU), Simeon/Shamanskiy (TU K'lautern), ...

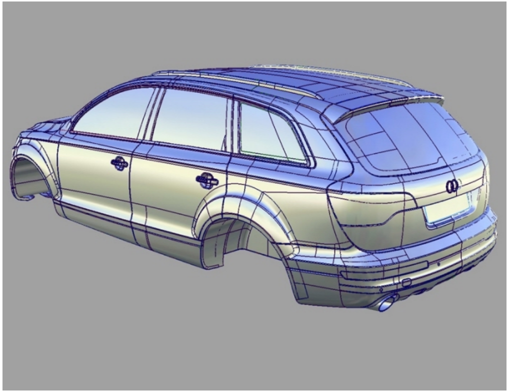
Funding

EU-H2020 MOTOR (GA 678727)

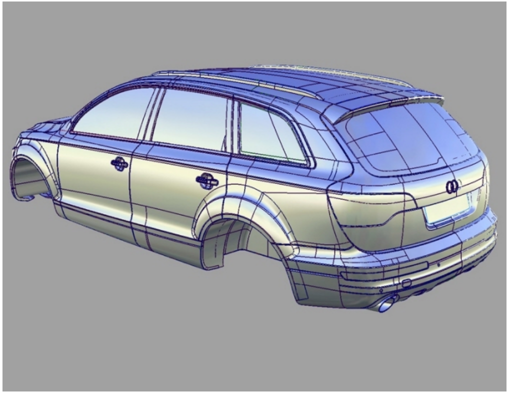
Design-through-Analysis

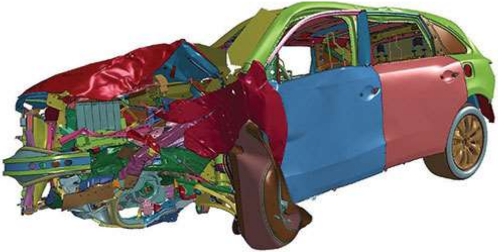
???

Design-through-Analysis

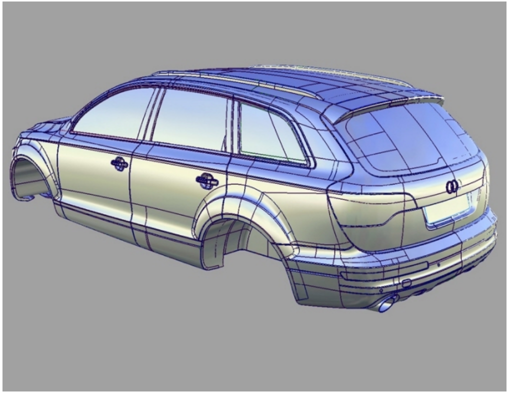


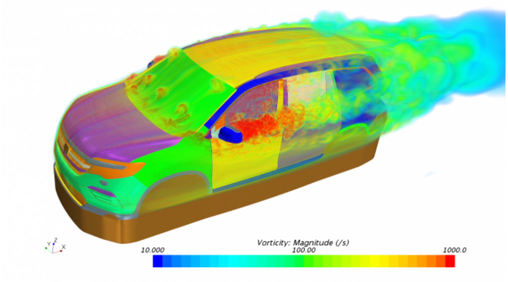
Design-through-Analysis





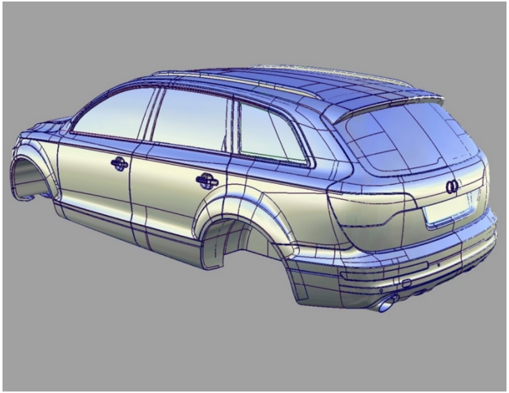
Design-through-Analysis

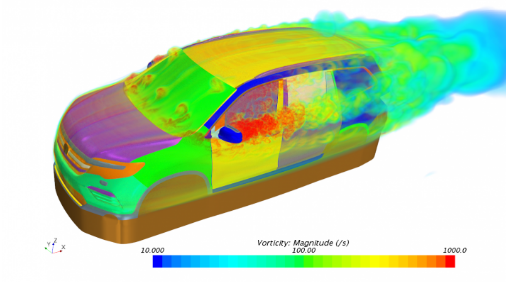




Ideally, we want a quick interaction between design (left) and analysis (right).

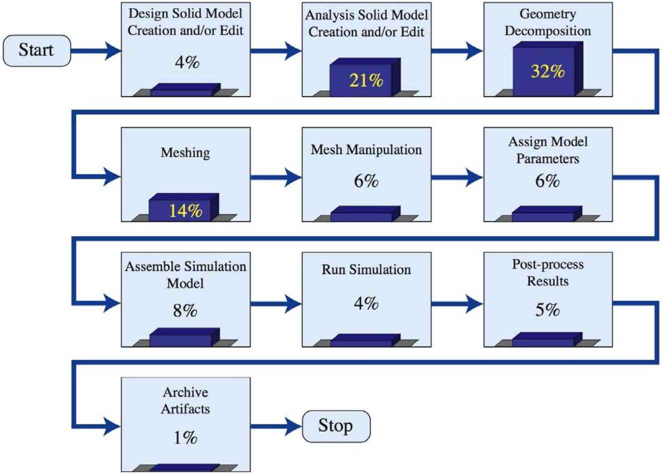
Design-through-Analysis





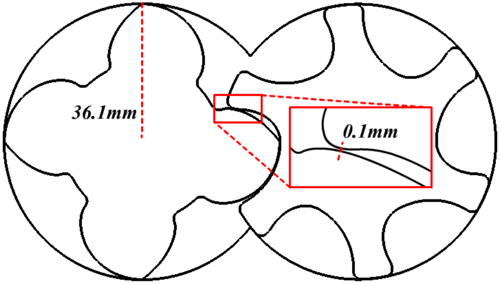
Ideally, we want a quick interaction between design (left) and analysis (right).

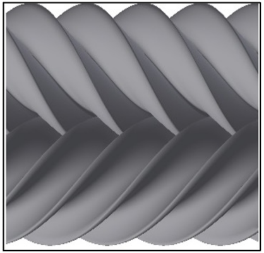
Design-through-Analysis

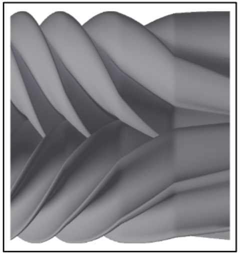


Ted Blacker, Sandia National Laboratories

Design-through-Analysis







We are mainly interested in 'designs' that are created algorithmically based on user-definable design parameters (e.g., wrap angle) and mathematical expressions.

Syllabus: Design-through-Analysis

IGA fundamentals

- Introduction to B-splines
- Geometry modelling and PDE analysis
- Assembly of system matrices
- Multi-patch coupling
- Adaptive spline technologies
- Efficient solution techniques

Analysis-suitable parametrizations

- PDE-based parametrization techniques

Gradient-based design optimization

- Gradient-based design optimization
- Algorithmic differentiation and computational aspects
- Selected applications

IGA fundamentals: Geometry modelling and PDE analysis with B-splines

Finite element analysis with B-spline basis functions

As in parametric finite elements, we transform integrals over the *physical* domain Ω into integrals over the (entire!) *parametric* domain $\hat{\Omega}$ by means of the **integration rule**

$$\int_{\Omega} w(\mathbf{x})\,\mathrm{d}\mathbf{x} = \int_{\hat{\Omega}} w(\mathbf{x}(\boldsymbol{\xi}))\ |\det J(\boldsymbol{\xi})|\ \mathrm{d}\boldsymbol{\xi}$$

with the **Jacobian matrix** given by

$$J(\boldsymbol{\xi}) = \left(\frac{\partial \mathbf{x}_a}{\partial \boldsymbol{\xi}_b}\right)_{a,b=1:2}$$

Finite element analysis with B-spline basis functions

As in parametric finite elements, we transform integrals over the *physical* domain Ω into integrals over the (entire!) *parametric* domain $\hat{\Omega}$ by means of the **integration rule**

$$\int_{\Omega} w(\mathbf{x})\,\mathrm{d}\mathbf{x} = \int_{\hat{\Omega}} w(\mathbf{x}(\boldsymbol{\xi}))\ |\det J(\boldsymbol{\xi})|\ \mathrm{d}\boldsymbol{\xi}$$

with the **Jacobian matrix** given by

$$J(\boldsymbol{\xi}) = \left(\frac{\partial \mathbf{x}_a}{\partial \boldsymbol{\xi}_b}\right)_{a,b=1:2}$$

Example

$$\frac{\partial \mathbf{x}_1}{\partial \boldsymbol{\xi}_1} = \frac{\partial x(\xi,\eta)}{\partial \xi} = \sum_{j=1}^{N_b} \mathrm{x}_j \frac{d}{d\xi} B_{j_\xi}(\xi) B_{j_\eta}(\eta)$$

whereby the **derivative of the univariate B-spline basis function** is given by

$$\frac{d}{dt} B_{i,p}(t) = \frac{p}{t_{i+p} - t_i} B_{i,p-1}(t) - \frac{p}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(t)$$

Finite element analysis with B-spline basis functions, cont'd

Making further use of the **chain rule of differentiation**

$$\nabla_{\mathbf{x}} u(\mathbf{x}) = \nabla_{\boldsymbol{\xi}} u(\boldsymbol{\xi}) \cdot J(\boldsymbol{\xi})^{-1}$$

we obtain the following expression for, e.g.

$$k(w,u) = \kappa \int_{\Omega} \nabla_{\mathbf{x}} w \cdot \nabla_{\mathbf{x}} u \, \mathrm{d}\mathbf{x} = \kappa \int_{\hat{\Omega}} (\nabla_{\boldsymbol{\xi}} w(\boldsymbol{\xi}) \cdot J(\boldsymbol{\xi})^{-1}) \cdot (\nabla_{\boldsymbol{\xi}} u(\boldsymbol{\xi}) \cdot J(\boldsymbol{\xi})^{-1}) \, |\det J(\boldsymbol{\xi})| \, \mathrm{d}\boldsymbol{\xi}$$

Finite element analysis with B-spline basis functions, cont'd

Making further use of the **chain rule of differentiation**

$$\nabla_{\mathbf{x}} u(\mathbf{x}) = \nabla_{\boldsymbol{\xi}} u(\boldsymbol{\xi}) \cdot J(\boldsymbol{\xi})^{-1}$$

we obtain the following expression for, e.g.

$$k(w,u) = \kappa \int_{\Omega} \nabla_{\mathbf{x}} w \cdot \nabla_{\mathbf{x}} u \, \mathrm{d}\mathbf{x} = \kappa \int_{\hat{\Omega}} (\nabla_{\boldsymbol{\xi}} w(\boldsymbol{\xi}) \cdot J(\boldsymbol{\xi})^{-1}) \cdot (\nabla_{\boldsymbol{\xi}} u(\boldsymbol{\xi}) \cdot J(\boldsymbol{\xi})^{-1}) \, |\det J(\boldsymbol{\xi})| \, \mathrm{d}\boldsymbol{\xi}$$

Similar expression can be derived for $a(w,u)$, $\langle w, u \rangle$ and $l(w)$ in the same way as it is done in classical finite element analysis. The main difference consists in the fact that $\hat{\Omega} = [0,1]^2$ denotes the *entire parametric domain* and not a single reference element $\hat{T}$. This requires some extra effort in the assembly of matrices/vectors via *numerical* integration.

Finite element analysis with B-spline basis functions, cont'd

Making further use of the **chain rule of differentiation**

$$\nabla_{\mathbf{x}} u(\mathbf{x}) = \nabla_{\boldsymbol{\xi}} u(\boldsymbol{\xi}) \cdot J(\boldsymbol{\xi})^{-1}$$

we obtain the following expression for, e.g.

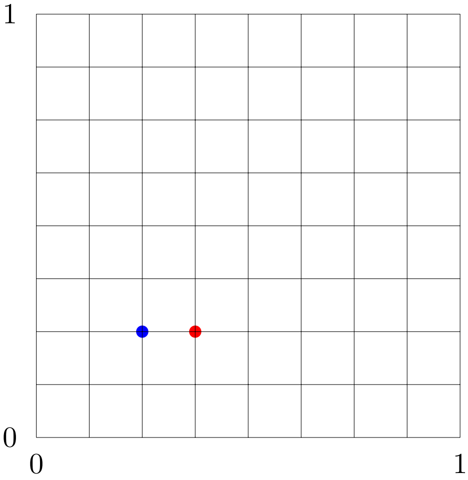
$$k(w,u) = \kappa \int_{\Omega} \nabla_{\mathbf{x}} w \cdot \nabla_{\mathbf{x}} u \, \mathrm{d}\mathbf{x} = \kappa \int_{\hat{\Omega}} (\nabla_{\boldsymbol{\xi}} w(\boldsymbol{\xi}) \cdot J(\boldsymbol{\xi})^{-1}) \cdot (\nabla_{\boldsymbol{\xi}} u(\boldsymbol{\xi}) \cdot J(\boldsymbol{\xi})^{-1}) \, |\det J(\boldsymbol{\xi})| \, \mathrm{d}\boldsymbol{\xi}$$

Similar expression can be derived for $a(w,u)$, $\langle w,u \rangle$ and $l(w)$ in the same way as it is done in classical finite element analysis. The main difference consists in the fact that $\hat{\Omega} = [0,1]^2$ denotes the *entire parametric domain* and not a single reference element $\hat{T}$. This requires some extra effort in the assembly of matrices/vectors via *numerical* integration.

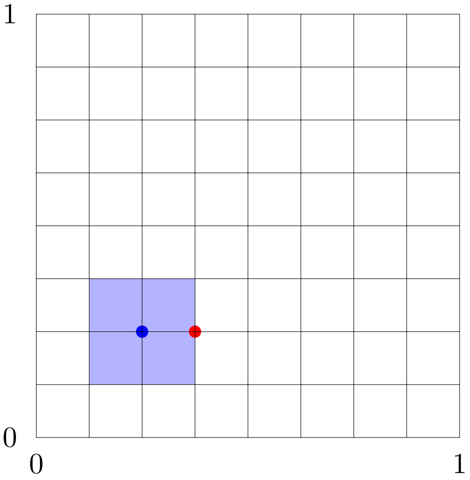
From now on we will refer to the above approach as **isogeometric analysis** (IGA).

T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. CMAME, 194(39):4135–4195, 2005.

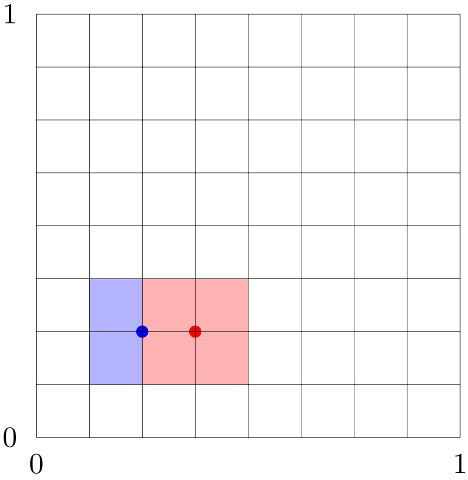
Matrix assembly for standard C^0-FEA



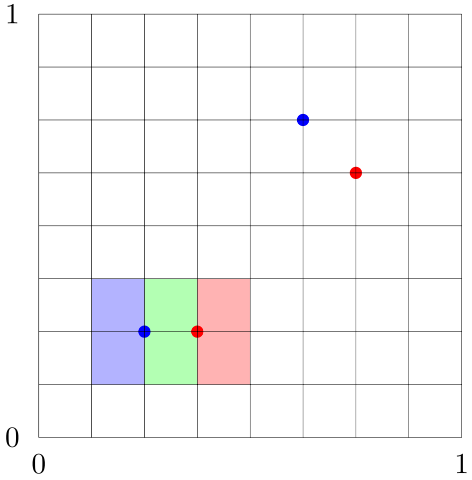
Matrix assembly for standard C^0-FEA



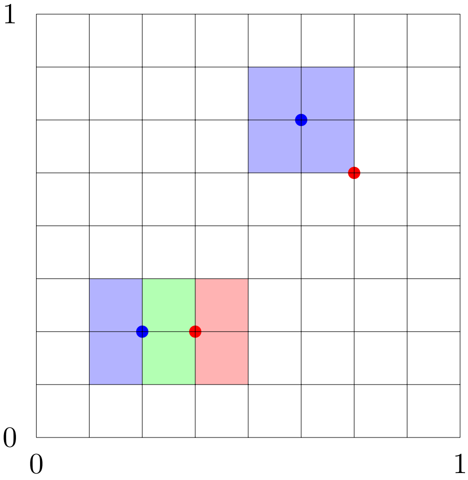
Matrix assembly for standard C^0-FEA



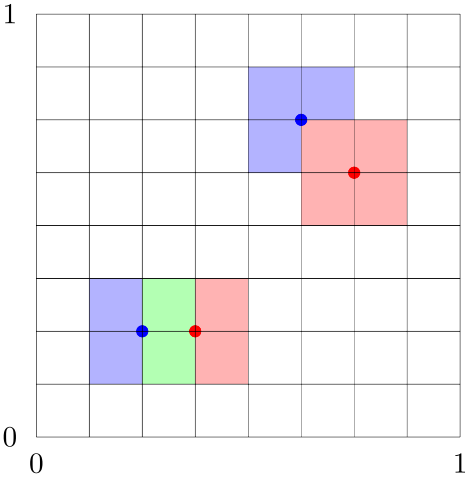
Matrix assembly for standard C^0-FEA



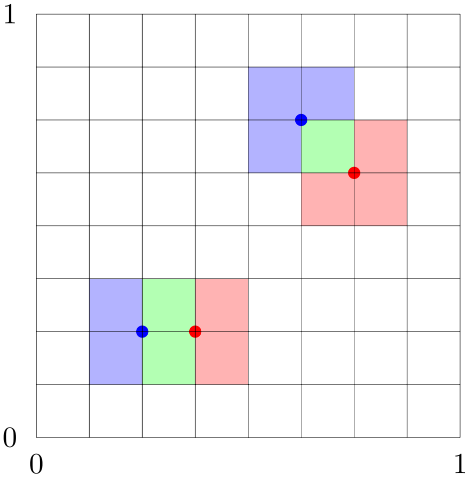
Matrix assembly for standard C^0-FEA



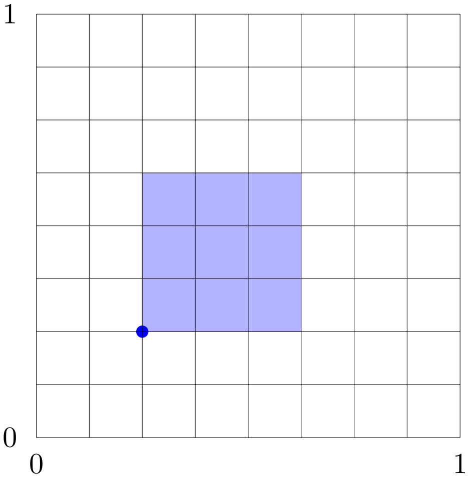
Matrix assembly for standard C^0-FEA



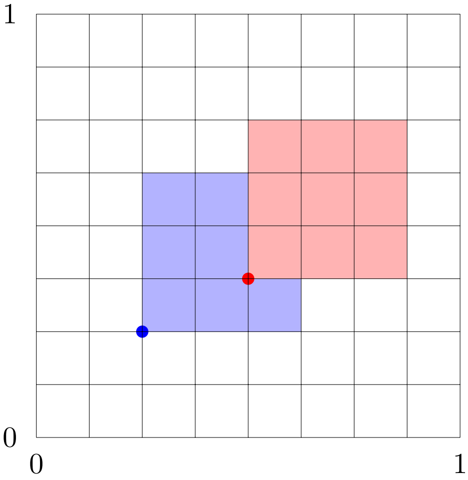
Matrix assembly for standard C^0-FEA



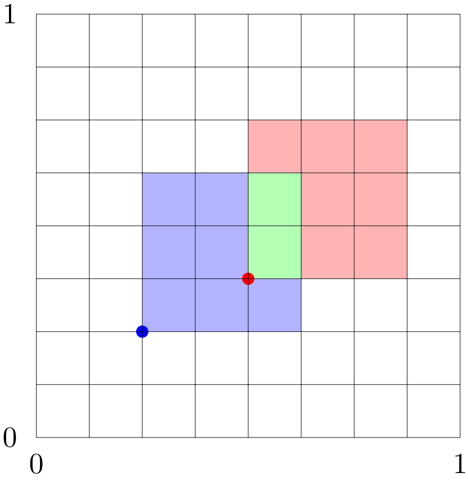
Matrix assembly for IGA



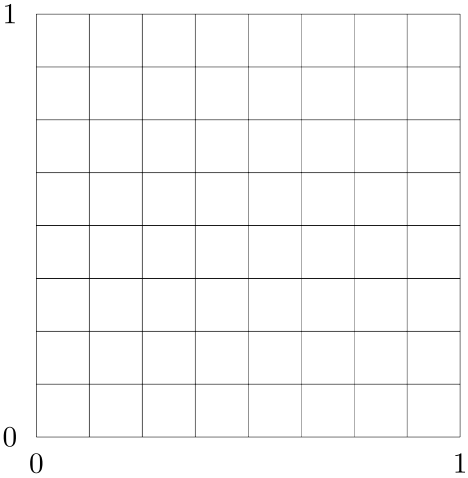
Matrix assembly for IGA



Matrix assembly for IGA



Additional notes



IGA fundamentals: Refinement and adaptive splines

Refinement techniques in IGA

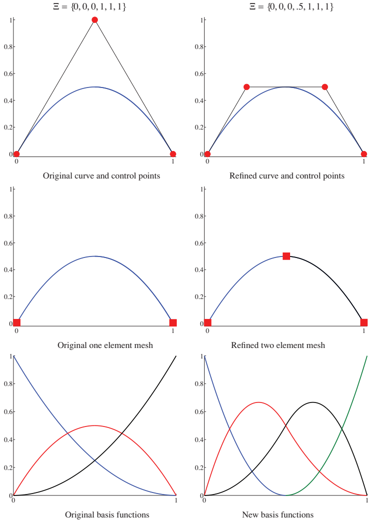
Like in classical FEA, the B-spline space $\mathcal{V}_{h,p}$
can be refined with respect to h and p:

J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric Analysis. Towards Integration of CAD and FEA.

Refinement techniques in IGA

Like in classical FEA, the B-spline space $\mathcal{V}_{h,p}$ can be refined with respect to h and p:

- **Knot insertion** (‘h-refinement’)

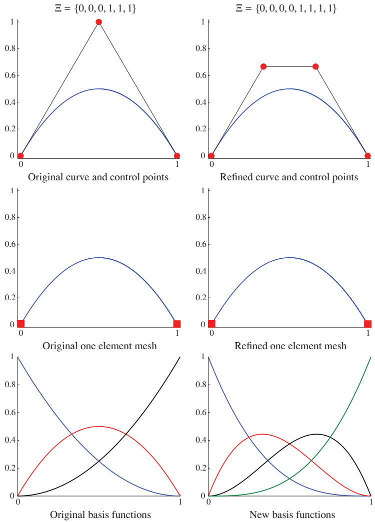


J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric Analysis. Towards Integration of CAD and FEA.

Refinement techniques in IGA

Like in classical FEA, the B-spline space $\mathcal{V}_{h,p}$ can be refined with respect to h and p:

- **Knot insertion** ('h-refinement')
- **Order elevation** ('p-refinement')



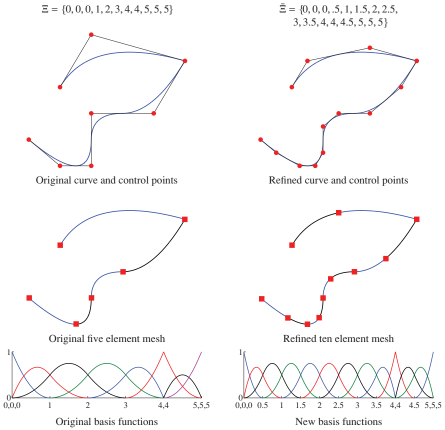
J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric Analysis. Towards Integration of CAD and FEA.

Refinement techniques in IGA

Like in classical FEA, the B-spline space $\mathcal{V}_{h,p}$ can be refined with respect to h and p:

- **Knot insertion** ('h-refinement')
- **Order elevation** ('p-refinement')

In both cases, the represented object (geometry *and* solution) is preserved exactly.



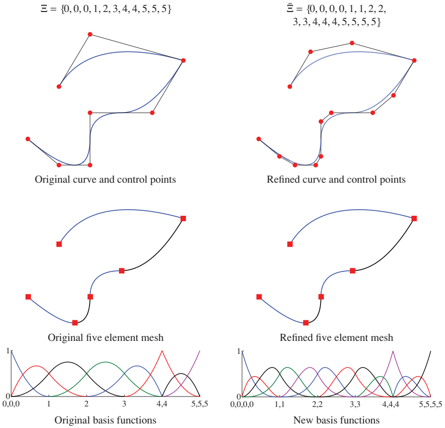
J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric Analysis. Towards Integration of CAD and FEA.

Refinement techniques in IGA

Like in classical FEA, the B-spline space $\mathcal{V}_{h,p}$ can be refined with respect to h and p:

- **Knot insertion** (‘h-refinement’)
- **Order elevation** (‘p-refinement’)

In both cases, the represented object (geometry *and* solution) is preserved exactly.



J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric Analysis. Towards Integration of CAD and FEA.

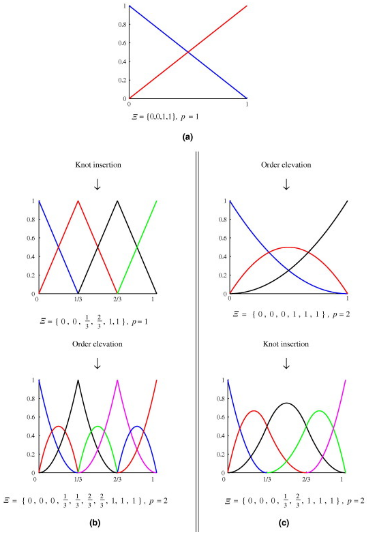
Refinement techniques in IGA

Like in classical FEA, the B-spline space $\mathcal{V}_{h,p}$ can be refined with respect to h and p:

- **Knot insertion** (‘h-refinement’)
- **Order elevation** (‘p-refinement’)

In both cases, the represented object (geometry *and* solution) is preserved exactly.

- k-**refinement** is a unique IGA feature to achieve higher order *and* higher continuity at the same time



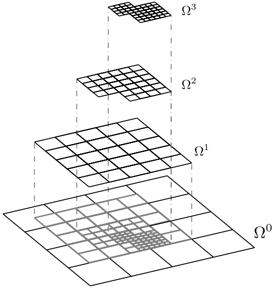
J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric Analysis. Towards Integration of CAD and FEA.

Adaptive spline technologies

- Powell-Sabin splines [Powell, Sabin 1977, Speleers et al. 2012]
- H(ierarchical) B-splines [Forsey, Bartels 1988, Kraft 1997, Vuong et al. 2011]
- T-splines [Sederberg et al. 2003, U.S. patent in 2007 to T-Splines, Inc., now Autodesk]
- Polynomial splines over hierarchical T-meshes [Deng, Chen 2007, Wang et al. 2011]
- U(nstructured)-splines [Thomas et al. 2008, Coreform LLC]
- T(runcated) H(ierarchical) B-splines [Gianelli et al. 2012]
- L(ocally) R(efinable) splines [Dokken et al. 2013]
- ...

A gentle introduction to THB splines

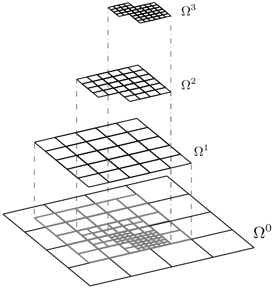
- Let $V^0 \subset V^1 \subset \cdots \subset V^{N-1}$ be a sequence of N nested spline spaces defined on the domain Ω^0.
- Let $\mathcal{B}^\ell$ denote the B-spline basis associated to the space V^ℓ with $\hat{\varphi}_i^\ell \in \mathcal{B}^\ell$, $i = 1, \ldots, N_b^\ell$ being its basis functions.
- Let $\Omega^0 \subseteq \Omega^1 \subseteq \cdots \subseteq \Omega^{N-1}$ be a sequence of nested domains as depicted on the right.



Gianelli *et al.* THB-splines: The truncated basis for hierarchical splines, CAGD 29:485–498, 2012.

A gentle introduction to THB splines

- Let $V^0 \subset V^1 \subset \cdots \subset V^{N-1}$ be a sequence of N nested spline spaces defined on the domain Ω^0.
- Let $\mathcal{B}^\ell$ denote the B-spline basis associated to the space V^ℓ with $\hat{\varphi}_i^\ell \in \mathcal{B}^\ell$, $i = 1, \ldots, N_b^\ell$ being its basis functions.
- Let $\Omega^0 \subseteq \Omega^1 \subseteq \cdots \subseteq \Omega^{N-1}$ be a sequence of nested domains as depicted on the right.



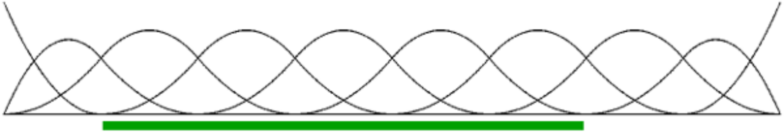
Hierarchical B-spline basis $\mathcal{H} := \mathcal{H}^{N-1}$:

$$\mathcal{H}^0 := \mathcal{B}^0$$
$$\mathcal{H}^{\ell+1} := \left\{ \hat{\varphi}_i^\ell \in \mathcal{H}^\ell \;:\; \text{supp}\ \hat{\varphi}_i^\ell \not\subseteq \Omega^{\ell+1} \right\} \cup \left\{ \hat{\varphi}_i^{\ell+1} \in \mathcal{B}^{\ell+1} \;:\; \text{supp}\ \hat{\varphi}_i^{\ell+1} \subseteq \Omega^{\ell+1} \right\}$$

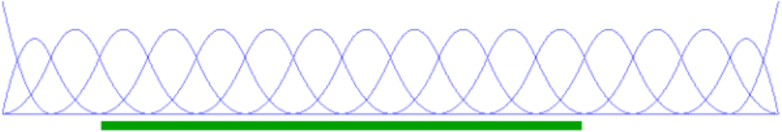
with supp $f := \{\mathbf{x} \,:\, f(\mathbf{x}) \neq 0 \wedge \mathbf{x} \in \Omega^0\}$.

Gianelli *et al.* THB-splines: The truncated basis for hierarchical splines, CAGD 29:485–498, 2012.

A gentle introduction to THB splines, cont'd



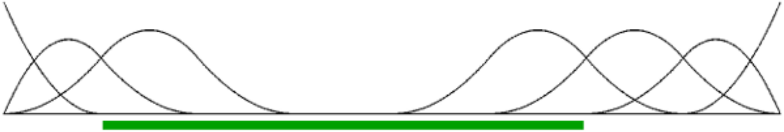
B-splines of level 0



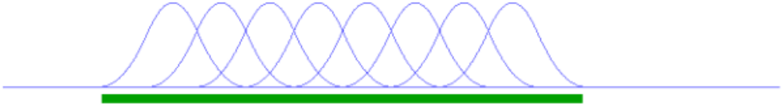
B-splines of level 1

Illustrations taken from `https://gismo.github.io/thbSplineBasis_example.html`

A gentle introduction to THB splines, cont'd



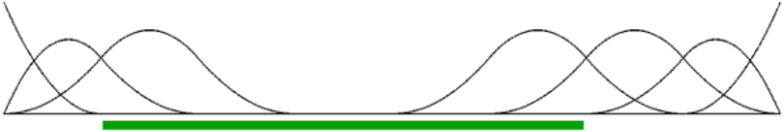
Active B-splines of level 0



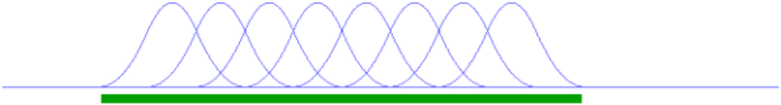
Active B-splines of level 1

Illustrations taken from `https://gismo.github.io/thbSplineBasis_example.html`

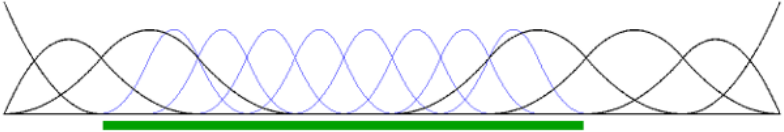
A gentle introduction to THB splines, cont'd



Active B-splines of level 0



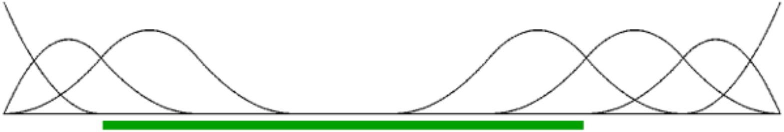
Active B-splines of level 1



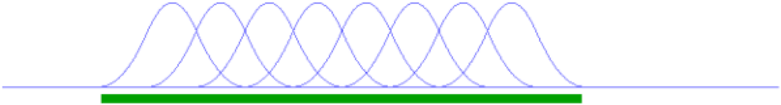
HB-splines (lack the partition-of-unity property)

Illustrations taken from `https://gismo.github.io/thbSplineBasis_example.html`

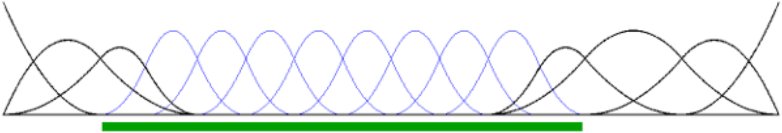
A gentle introduction to THB splines, cont'd



Active B-splines of level 0



Active B-splines of level 1



THB-splines (exhibit the partition-of-unity property)

Illustrations taken from `https://gismo.github.io/thbSplineBasis_example.html`

A gentle introduction to THB splines

***Truncated* Hierarchical B-spline basis** $\mathcal{T} := \mathcal{T}^{N-1}$:

$$\mathcal{T}^0 := \mathcal{B}^0$$

$$\mathcal{T}^{\ell+1} := \left\{ \text{trunc}^{\ell+1}\tau \ : \ \tau \in \mathcal{T}^{\ell} \wedge \text{supp } \tau \not\subseteq \Omega^{\ell+1} \right\} \cup \left\{ \hat{\varphi}_i^{\ell+1} \in \mathcal{B}^{\ell+1} \ : \ \text{supp } \hat{\varphi}_i^{\ell+1} \subseteq \Omega^{\ell+1} \right\}$$

with the **truncation operator** defined as follows:

Let $\tau \in V^{\ell}$ and its representation in terms of the *finer* basis $\mathcal{B}^{\ell+1}$ be given by

$$\tau = \sum_{j=1}^{N_b^{\ell+1}} c_j^{\ell+1}(\tau)\hat{\varphi}_j^{\ell+1}, \qquad c_j^{\ell+1}(\tau) \in \mathbb{R}, \quad \hat{\varphi}_j^{\ell+1} \in \mathcal{B}^{\ell+1}$$

Then

$$\text{trunc}^{\ell+1}\tau := \sum_{\substack{j=1 \\ \text{supp } \hat{\varphi}_j^{\ell+1} \not\subseteq \Omega^{\ell+1}}}^{N_b^{\ell+1}} c_j^{\ell+1}(\tau)\hat{\varphi}_j^{\ell+1}, \qquad c_j^{\ell+1}(\tau) \in \mathbb{R}, \quad \hat{\varphi}_j^{\ell+1} \in \mathcal{B}^{\ell+1}$$

Gianelli *et al.* THB-splines: The truncated basis for hierarchical splines, CAGD 29:485–498, 2012.

IGA fundamentals: Efficient solution techniques

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]

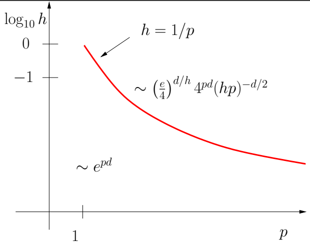
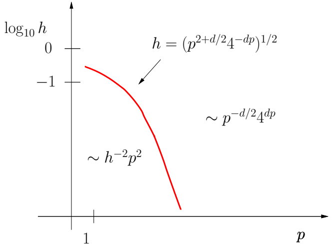
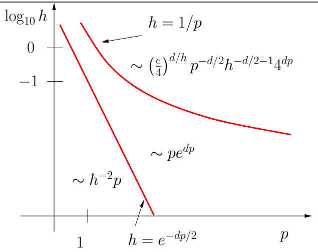
h-multigrid techniques

- Full multigrid [Hofreither 2016]
- THB-splines [Hofreither *et al.* 2017]
- Symbol-based [Donatelli 2017]
- Boundary correction [Hofreither *et al.* 2017]
- Subspace corrected smoother [Takacs *et al.* 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

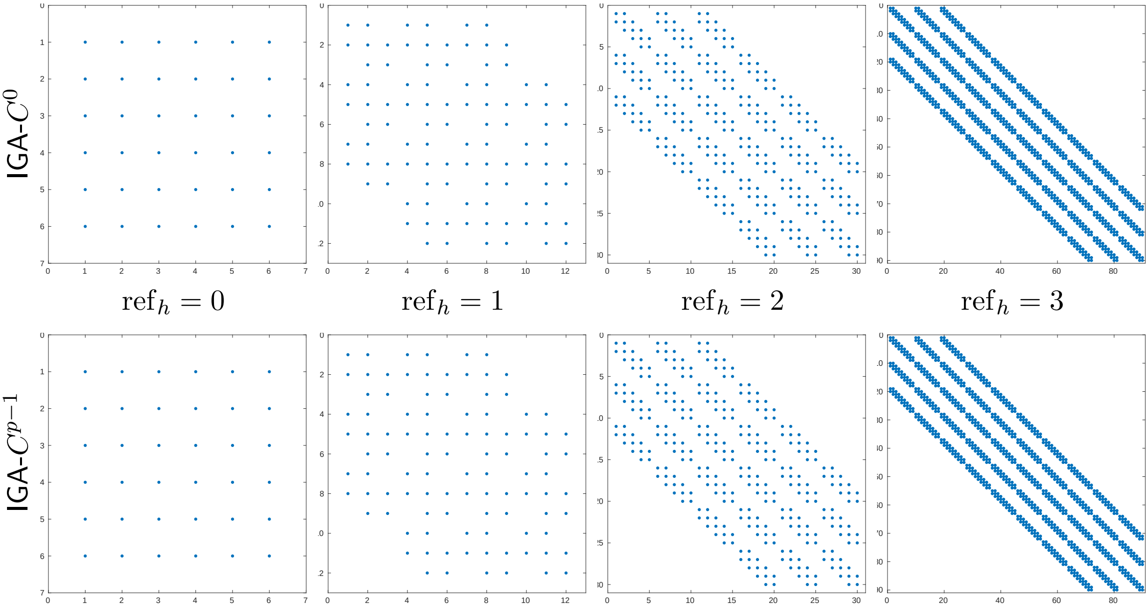
- Parallel splitting solvers [Puzyrev *et al.* 2019]
- Space-time solvers [Langer *et al.* 2016]
- Space-time solvers [Loli *et al.* 2020]
- Space-time least-squares [Montardini *et al.* 2020]
- MGRIT-IGA [Tielen *et al.* 2021]

Condition number

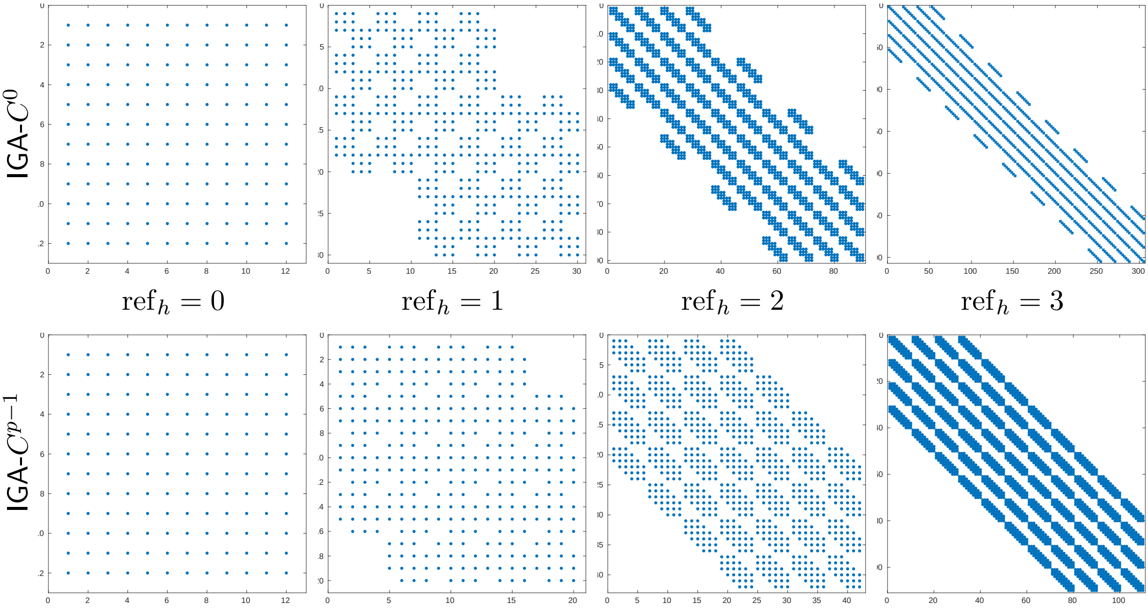
	SEM-NI	IGA-C^0	IGA-C^{p-1}
$\mathcal{K}(M)$	$\sim p^d$	$\sim p^{-d/2}4^{pd}$	
$\mathcal{K}(K)$	$\sim h^{-2}p^3$		

From: P. Gervasio, L. Dedè, O. Chanon, and A. Quarteroni, DOI: 10.1007/s10915-020-01204-1

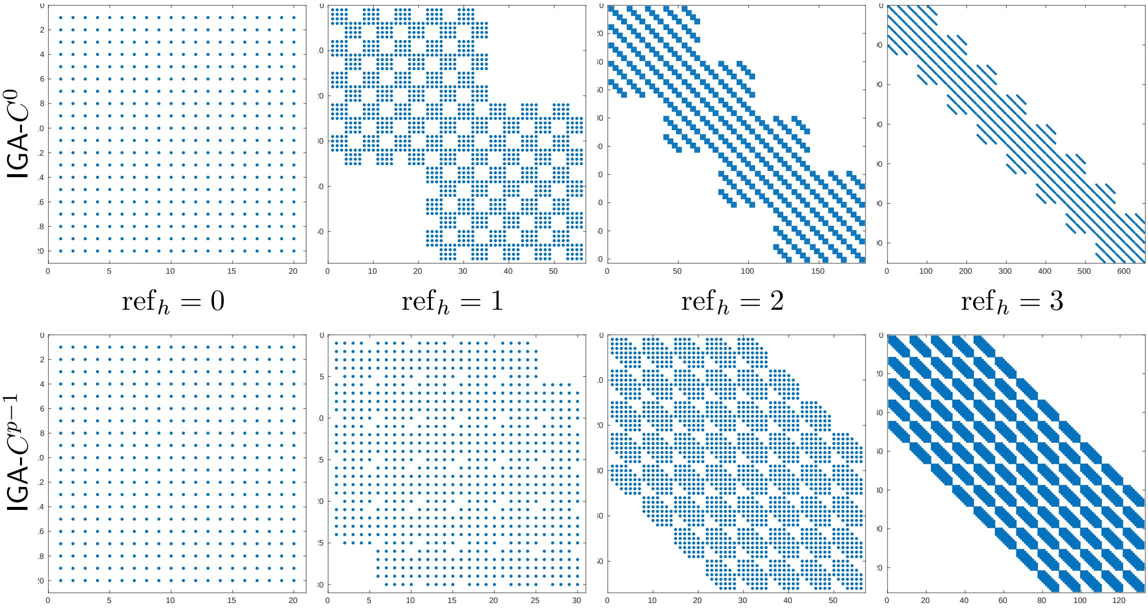
Sparsity pattern: 2d single patch, $p = 1$



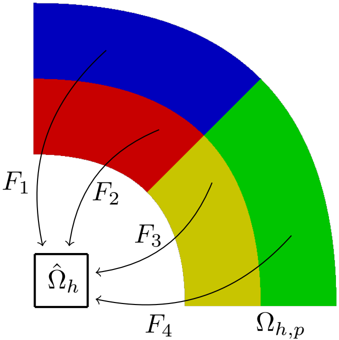
Sparsity pattern: 2d single patch, $p = 2$

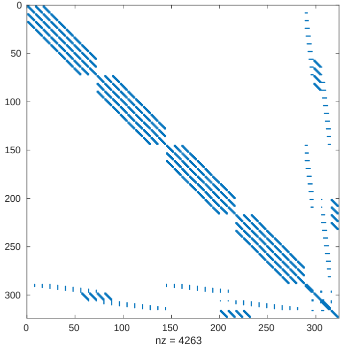


Sparsity pattern: 2d single patch, $p = 3$



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

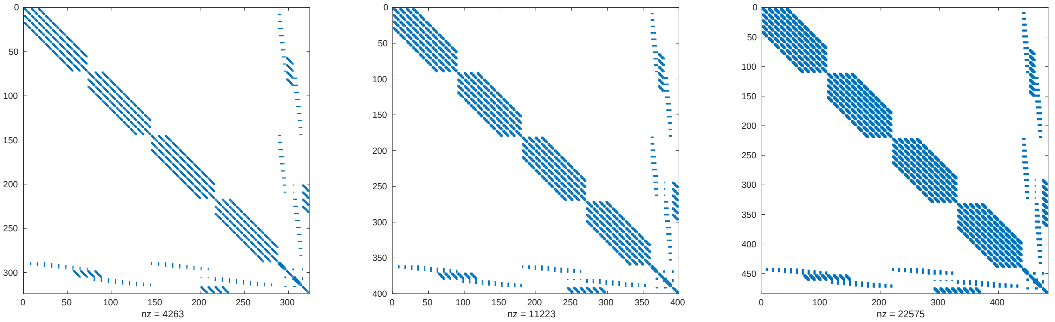




Four-patch geometry with C^0 coupling of conforming degrees of freedom.

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

$p = 1$ $\qquad$ $p = 2$ $\qquad$ $p = 3$

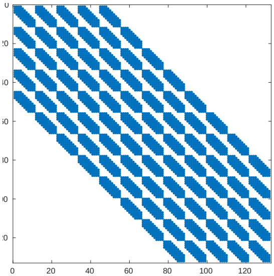


Four-patch geometry with C^0 coupling of conforming degrees of freedom.

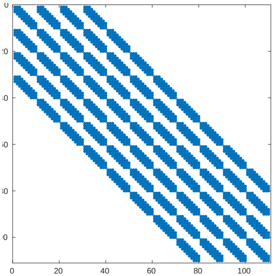
Sketch of our solution strategy

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

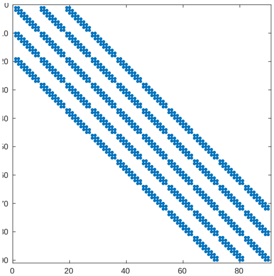
$p = 3$

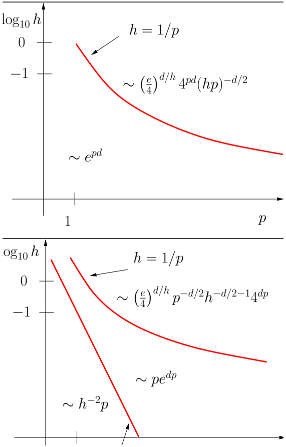


$p = 2$



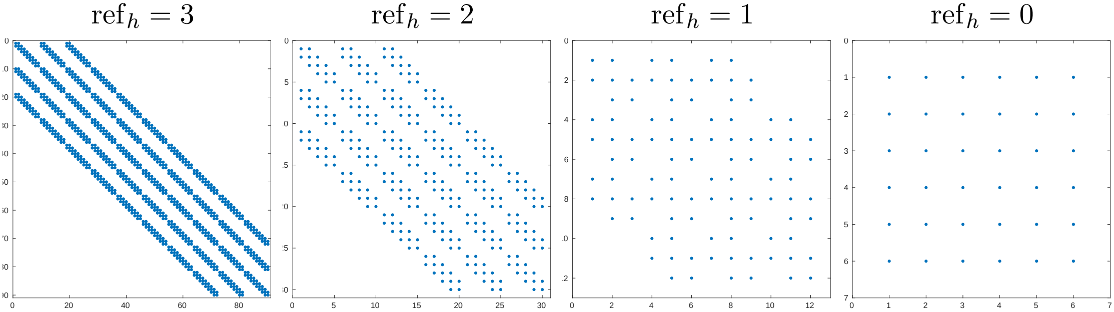
$p = 1$





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 $(\mathcal{V}_{h,p} \not\supset \mathcal{V}_{h,p-1} \not\supset \dots)$
  - **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

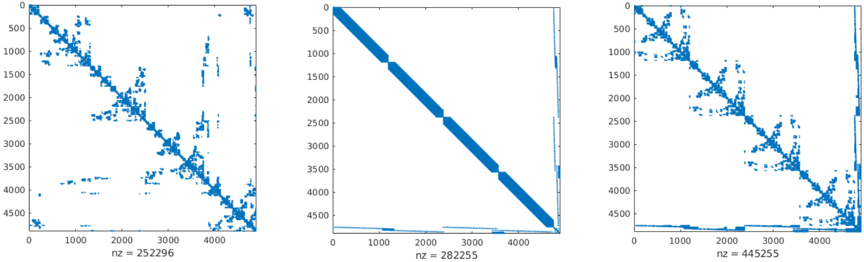


Sketch of our solution strategy

- 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

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



Sketch of our solution strategy

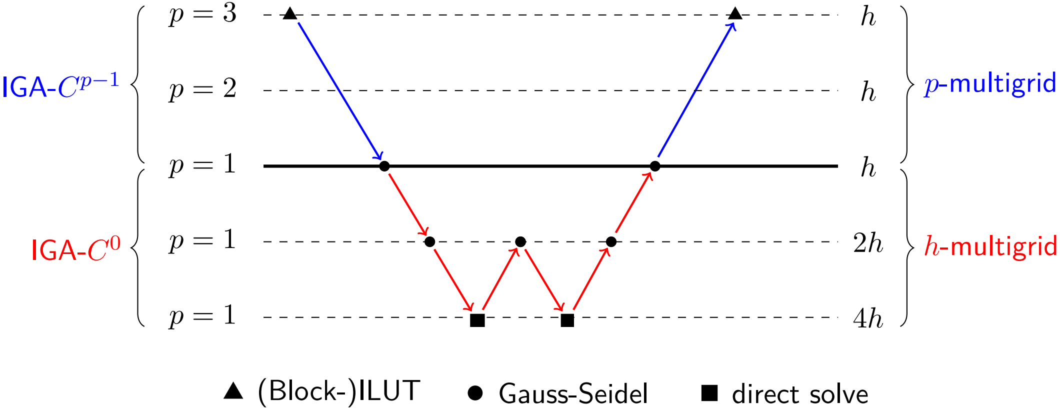
- Coarsening in p reduces the stencil but not so much the number of unknowns
  - p-**multigrid with *direct* projection** $\mathcal{V}_{h,p} \searrow \mathcal{V}_{h,1}$
  - note that spaces are not nested $(\mathcal{V}_{h,p} \not\supset \mathcal{V}_{h,p-1} \not\supset \dots)$
  - **ILUT smoother** at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



The complete multigrid algorithm – the outer p-multigrid part

1. Starting from $\mathrm{u}_{h,p}^{(0,0)}$ apply ν_1 pre-smoothing steps:

$$\mathrm{u}_{h,p}^{(0,m)} := \mathrm{u}_{h,p}^{(0,m-1)} + \mathrm{S}_{h,p}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,m-1)}\right), \quad m = 0, 1, \ldots, \nu_1$$

The complete multigrid algorithm – the outer p-multigrid part

1. Starting from $\mathrm{u}_{h,p}^{(0,0)}$ apply ν_1 pre-smoothing steps:

$$\mathrm{u}_{h,p}^{(0,m)} := \mathrm{u}_{h,p}^{(0,m-1)} + \mathrm{S}_{h,p}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,m-1)}\right), \quad m = 0, 1, \dots, \nu_1$$

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

$$\mathrm{r}_{h,1} = \mathrm{I}_{h,p}^{h,1}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,\nu_1)}\right), \quad \mathrm{I}_{h,p}^{h,1} := \mathrm{M}_{h,1}^{-1}\,\mathrm{M}_{h,p,1}$$

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

The complete multigrid algorithm – the outer p-multigrid part

1. Starting from $\mathrm{u}_{h,p}^{(0,0)}$ apply ν_1 pre-smoothing steps:

$$\mathrm{u}_{h,p}^{(0,m)} := \mathrm{u}_{h,p}^{(0,m-1)} + \mathrm{S}_{h,p}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,m-1)}\right), \quad m = 0, 1, \dots, \nu_1$$

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

$$\mathrm{r}_{h,1} = \mathrm{I}_{h,p}^{h,1}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,\nu_1)}\right), \quad \mathrm{I}_{h,p}^{h,1} := \mathrm{M}_{h,1}^{-1}\,\mathrm{M}_{h,p,1}$$

   with $\mathrm{M}_{h,p,1} = \{(\varphi_i, \psi_j)\}_{i,j}$, 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:

$$\mathrm{A}_{h,1}\,\mathrm{e}_{h,1} = \mathrm{r}_{h,1}$$

The complete multigrid algorithm – the outer p-multigrid part

1. Starting from $\mathrm{u}_{h,p}^{(0,0)}$ apply ν_1 pre-smoothing steps:

$$\mathrm{u}_{h,p}^{(0,m)} := \mathrm{u}_{h,p}^{(0,m-1)} + \mathrm{S}_{h,p}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,m-1)}\right), \quad m = 0, 1, \ldots, \nu_1$$

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

$$\mathrm{r}_{h,1} = \mathrm{I}_{h,p}^{h,1}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,\nu_1)}\right), \quad \mathrm{I}_{h,p}^{h,1} := \mathrm{M}_{h,1}^{-1}\,\mathrm{M}_{h,p,1}$$

   with $\mathrm{M}_{h,p,1} = \{(\varphi_i, \psi_j)\}_{i,j}$, 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:

$$\mathrm{A}_{h,1}\,\mathrm{e}_{h,1} = \mathrm{r}_{h,1}$$

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

$$\mathrm{u}_{h,p}^{(0,\nu_1)} := \mathrm{u}_{h,p}^{(0,\nu_1)} + \mathrm{I}_{h,1}^{h,p}\left(\mathrm{e}_{h,1}\right), \quad \mathrm{I}_{h,1}^{h,p} := \mathrm{M}_{h,p}^{-1}\,\mathrm{M}_{h,1,p}$$

The complete multigrid algorithm – the outer p-multigrid part

1. Starting from $\mathrm{u}_{h,p}^{(0,0)}$ apply ν_1 pre-smoothing steps:

$$\mathrm{u}_{h,p}^{(0,m)} := \mathrm{u}_{h,p}^{(0,m-1)} + \mathrm{S}_{h,p}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,m-1)}\right), \quad m = 0, 1, \ldots, \nu_1$$

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

$$\mathrm{r}_{h,1} = \mathrm{I}_{h,p}^{h,1}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,\nu_1)}\right), \quad \mathrm{I}_{h,p}^{h,1} := \mathrm{M}_{h,1}^{-1}\,\mathrm{M}_{h,p,1}$$

   with $\mathrm{M}_{h,p,1} = \{(\varphi_i, \psi_j)\}_{i,j}$, 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:

$$\mathrm{A}_{h,1}\,\mathrm{e}_{h,1} = \mathrm{r}_{h,1}$$

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

$$\mathrm{u}_{h,p}^{(0,\nu_1)} := \mathrm{u}_{h,p}^{(0,\nu_1)} + \mathrm{I}_{h,1}^{h,p}\left(\mathrm{e}_{h,1}\right), \quad \mathrm{I}_{h,1}^{h,p} := \mathrm{M}_{h,p}^{-1}\,\mathrm{M}_{h,1,p}$$

5. Apply ν_2 post-smoothing steps as in 1. to obtain $\mathrm{u}_{h,p}^{(1,0)} := \mathrm{u}_{h,p}^{(0,\nu_1+\nu_2)}$ and repeat steps 1.–5. until $\|\,\mathrm{r}_{h,p}^{(k)}\| < \mathrm{tol}\|\,\mathrm{r}_{\mathrm{h,p}}^{(0)}\|$ for some tolerance parameter tol.

The complete multigrid algorithm – the outer p-multigrid part

1. Starting from $\mathrm{u}_{h,p}^{(0,0)}$ apply ν_1 pre-smoothing steps:

$$\mathrm{u}_{h,p}^{(0,m)} := \mathrm{u}_{h,p}^{(0,m-1)} + \mathrm{S}_{h,p}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,m-1)}\right), \quad m = 0, 1, \ldots, \nu_1$$

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

$$\mathrm{r}_{h,1} = \mathrm{I}_{h,p}^{h,1}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,\nu_1)}\right), \quad \mathrm{I}_{h,p}^{h,1} := \mathrm{M}_{h,1}^{-1}\,\mathrm{M}_{h,p,1} \quad \text{mass lumping}$$

   with $\mathrm{M}_{h,p,1} = \{(\varphi_i, \psi_j)\}_{i,j}$, 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:

$$\mathrm{A}_{h,1}\,\mathrm{e}_{h,1} = \mathrm{r}_{h,1}$$

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

$$\mathrm{u}_{h,p}^{(0,\nu_1)} := \mathrm{u}_{h,p}^{(0,\nu_1)} + \mathrm{I}_{h,1}^{h,p}\left(\mathrm{e}_{h,1}\right), \quad \mathrm{I}_{h,1}^{h,p} := \mathrm{M}_{h,p}^{-1}\,\mathrm{M}_{h,1,p} \quad \text{mass lumping (B-splines!)}$$

5. Apply ν_2 post-smoothing steps as in 1. to obtain $\mathrm{u}_{h,p}^{(1,0)} := \mathrm{u}_{h,p}^{(0,\nu_1+\nu_2)}$ and repeat steps 1.–5. until $\|\,\mathrm{r}_{h,p}^{(k)}\| < \mathrm{tol}\|\,\mathrm{r}_{\mathrm{h,p}}^{(0)}\|$ for some tolerance parameter tol.

The complete multigrid algorithm – the outer p-multigrid part

1. Starting from $\mathrm{u}_{h,p}^{(0,0)}$ apply ν_1 pre-smoothing steps:

$$\mathrm{u}_{h,p}^{(0,m)} := \mathrm{u}_{h,p}^{(0,m-1)} + \mathrm{S}_{h,p}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,m-1)}\right), \quad m = 0, 1, \ldots, \nu_1$$

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

$$\mathrm{r}_{h,1} = \mathrm{I}_{h,p}^{h,1}\left(\mathrm{f}_{h,p} - \mathrm{A}_{h,p}\,\mathrm{u}_{h,p}^{(0,\nu_1)}\right), \quad \mathrm{I}_{h,p}^{h,1} := \mathrm{M}_{h,1}^{-1}\,\mathrm{M}_{h,p,1} \quad \text{mass lumping}$$

   with $\mathrm{M}_{h,p,1} = \{(\varphi_i, \psi_j)\}_{i,j}$, 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:

$$\mathrm{A}_{h,1}\,\mathrm{e}_{h,1} = \mathrm{r}_{h,1}$$

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

$$\mathrm{u}_{h,p}^{(0,\nu_1)} := \mathrm{u}_{h,p}^{(0,\nu_1)} + \mathrm{I}_{h,1}^{h,p}\left(\mathrm{e}_{h,1}\right), \quad \mathrm{I}_{h,1}^{h,p} := \mathrm{M}_{h,p}^{-1}\,\mathrm{M}_{h,1,p} \quad \text{mass lumping (B-splines!)}$$

5. Apply ν_2 post-smoothing steps as in 1. to obtain $\mathrm{u}_{h,p}^{(1,0)} := \mathrm{u}_{h,p}^{(0,\nu_1+\nu_2)}$ and repeat steps 1.–5. until $\|\,\mathrm{r}_{h,p}^{(k)}\| < \mathrm{tol}\|\,\mathrm{r}_{\mathrm{h,p}}^{(0)}\|$ for some tolerance parameter tol.

The complete multigrid algorithm – the inner h-multigrid part

3.1. Starting from $\mathrm{u}_{h,1}^{(k,0)}$ apply ν_1 pre-smoothing steps:

$$\mathrm{u}_{h,1}^{(k,m)} := \mathrm{u}_{h,1}^{(k,m-1)} + \mathrm{S}_{h,1}\left(\mathrm{f}_{h,1} - \mathrm{A}_{h,1}\,\mathrm{u}_{h,1}^{(k,m-1)}\right), \quad m = 0, 1, \ldots, \nu_1$$

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

$$\mathrm{r}_{2h,1} = \mathrm{I}_{h,1}^{2h,1}\left(\mathrm{f}_{h,1} - \mathrm{A}_{h,1}\,\mathrm{u}_{h,1}^{(k,\nu_1)}\right), \quad \mathrm{I}_{h,1}^{2h,1} \text{ linear interpolation}$$

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

$$\mathrm{A}_{2h,1}\,\mathrm{e}_{2h,1} = \mathrm{r}_{2h,1}$$

3.4. Project the error onto $\mathcal{V}_{h,1}$ and update the solution:

$$\mathrm{u}_{h,1}^{(k,\nu_1)} := \mathrm{u}_{h,1}^{(k,\nu_1)} + \mathrm{I}_{2h,1}^{h,1}\left(\mathrm{e}_{2h,1}\right), \quad \mathrm{I}_{2h,1}^{h,1} := \frac{1}{2}\left(\mathrm{I}_{h,1}^{2h,1}\right)^{\top}$$

3.5. Apply ν_2 post-smoothing steps as in 3.1. to obtain $\mathrm{u}_{h,1}^{(k+1,0)} := \mathrm{u}_{h,1}^{(k,\nu_1+\nu_2)}$ and repeat steps 3.1.–3.5. according to the h-multigrid cycle (V- or W-cycle).

Multigrid components

	h-multigrid	p-multigrid
restriction operator	$\mathrm{I}_{h,1}^{2h,1}$ linear interpolation	$\mathrm{I}_{h,1}^{h,p} := \mathrm{M}_{h,p}^{-1}\,\mathrm{M}_{h,1,p}$
prolongation operator	$\mathrm{I}_{2h,1}^{h,1} := \frac{1}{2}\left(\mathrm{I}_{h,1}^{2h,1}\right)^{\top}$	$\mathrm{I}_{h,p}^{h,1} := \mathrm{M}_{h,1}^{-1}\,\mathrm{M}_{h,p,1}$

Multigrid components

	h-multigrid	p-multigrid
restriction operator	$\mathrm{I}_{h,1}^{2h,1}$ linear interpolation	$\mathrm{I}_{h,1}^{h,p} := \mathrm{M}_{h,p}^{-1}\mathrm{M}_{h,1,p}$
prolongation operator	$\mathrm{I}_{2h,1}^{h,1} := \frac{1}{2}\left(\mathrm{I}_{h,1}^{2h,1}\right)^{\top}$	$\mathrm{I}_{h,p}^{h,1} := \mathrm{M}_{h,1}^{-1}\mathrm{M}_{h,p,1}$
smoothing operator	incomplete LU factorization of $\mathrm{A}_{h,p} \approx \mathrm{L}_{h,p}\mathrm{U}_{h,p}$, whereby all elements smaller than 10^{-13} are dropped and the amount of non-zero entries per row are kept constant	

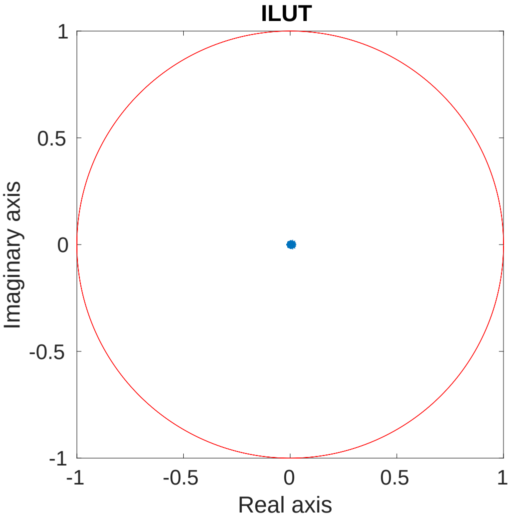
Y. Saad, ILUT: A dual threshold incomplete LU factorization, DOI: 10.1002/nla.1680010405

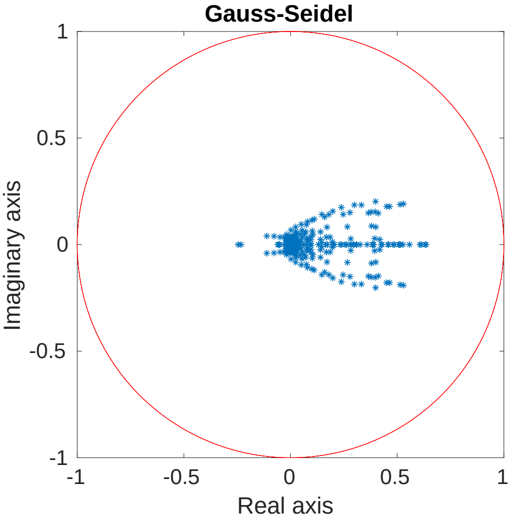
Multigrid components

	h-multigrid	p-multigrid
restriction operator	$\mathrm{I}_{h,1}^{2h,1}$ linear interpolation	$\mathrm{I}_{h,1}^{h,p} := \mathrm{M}_{h,p}^{-1}\,\mathrm{M}_{h,1,p}$
prolongation operator	$\mathrm{I}_{2h,1}^{h,1} := \frac{1}{2}\left(\mathrm{I}_{h,1}^{2h,1}\right)^{\top}$	$\mathrm{I}_{h,p}^{h,1} := \mathrm{M}_{h,1}^{-1}\,\mathrm{M}_{h,p,1}$
smoothing operator	incomplete LU factorization of $\mathrm{A}_{h,p} \approx \mathrm{L}_{h,p}\,\mathrm{U}_{h,p}$, whereby all elements smaller than 10^{-13} are dropped and the amount of non-zero entries per row are kept constant	
$\mathrm{A}_{h,p}$ operator	rediscretization	

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

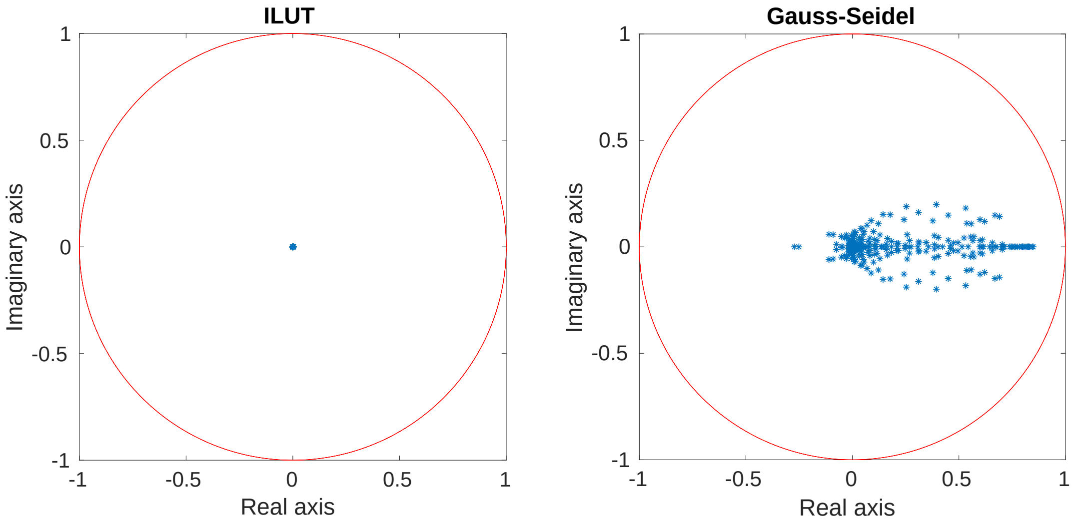
Spectrum of the iteration matrix: Poisson on quarter annulus, $p = 2$





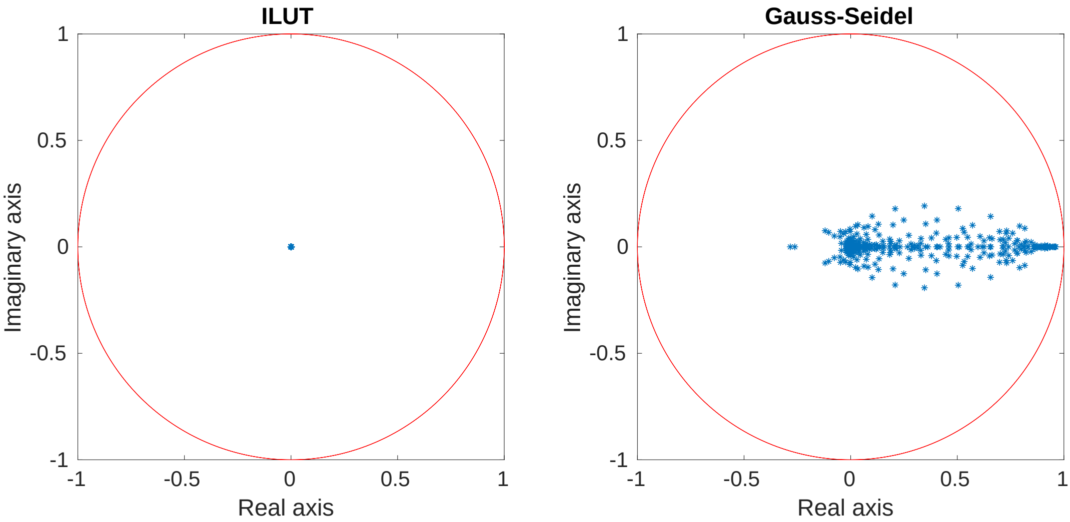
R. Tielen *et al.* 2020, DOI: 10.1016/j.cma.2020.113347

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



R. Tielen *et al.* 2020, DOI: 10.1016/j.cma.2020.113347

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



R. Tielen *et al.* 2020, DOI: 10.1016/j.cma.2020.113347

Numerical examples

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

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	4	30	3	62	3	176	3	491
$h=2^{-7}$	4	29	3	61	3	172	3	499
$h=2^{-8}$	5	30	3	60	3	163	3	473
$h=2^{-9}$	5	32	3	61	3	163	3	452

R. Tielen et al. 2020, DOI: 10.1016/j.cma.2020.113347

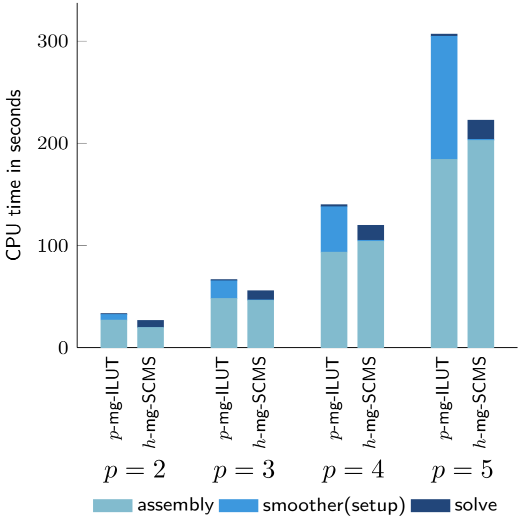
Numerical examples

#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

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h = 2^{-6}$	5	–	3	–	3	–	4	–
$h = 2^{-7}$	5	–	3	–	4	–	4	–
$h = 2^{-8}$	5	–	3	–	3	–	4	–
$h = 2^{-9}$	5	–	4	–	3	–	4	–

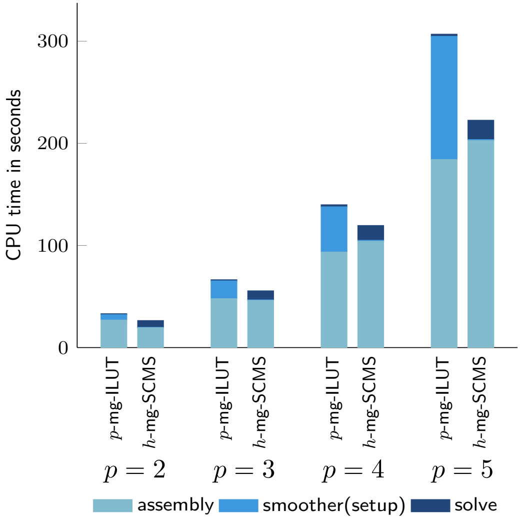
R. Tielen et al. 2020, DOI: 10.1016/j.cma.2020.113347

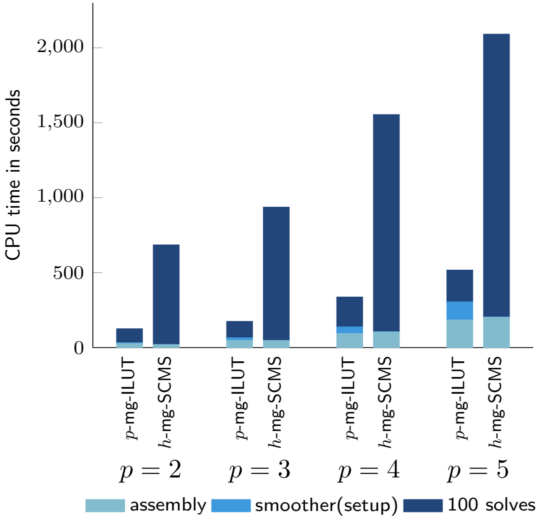
Computational efficiency: p- *vs.* h-*multigrid*



Comparison with h-multigrid method with subspace corrected mass smoother [Takacs, 2017]

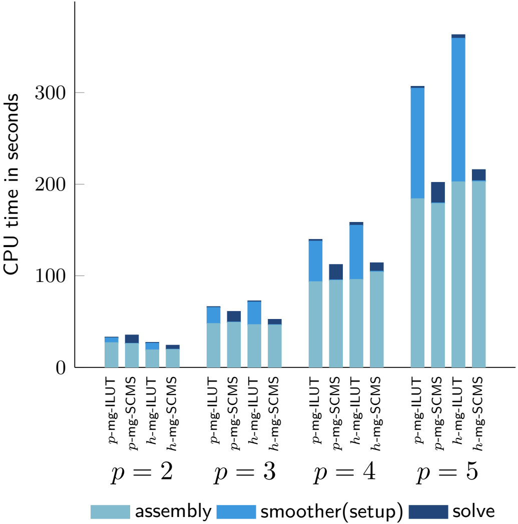
Computational efficiency: p- *vs.* h-*multigrid*

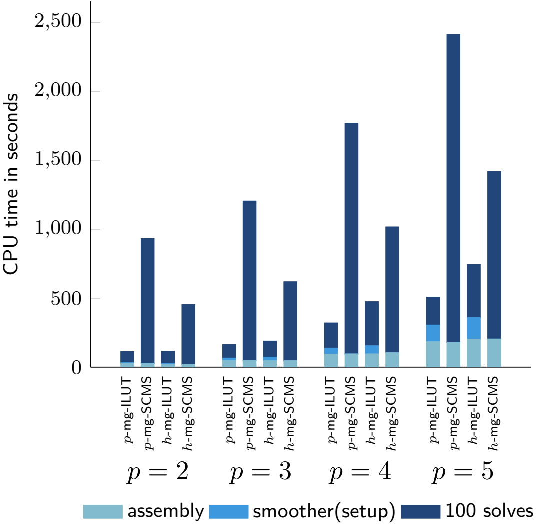




Comparison with h-multigrid method with subspace corrected mass smoother [Takacs, 2017]

Computational efficiency: $\{h,p\}$-*multigrid* + $\{$*ILUT,SCMS*$\}$-*smoother*

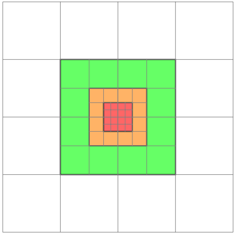


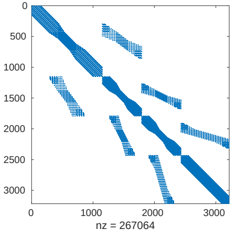


Numerical examples: *THB splines*

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

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-4}$	6	17	8	47	7	177	10	1033
$h=2^{-5}$	6	16	7	44	8	182	7	923
$h=2^{-6}$	6	17	5	43	6	201	12	1009





R. Tielen et al. 2020, DOI: 10.1016/j.cma.2020.113347

Block ILUT

Exact LU decomposition of the block matrix A

$$\begin{bmatrix} \mathrm{A}_{11} & & & \mathrm{A}_{\Gamma 1} \\ & \ddots & & \vdots \\ & & \mathrm{A}_{N_p N_p} & \mathrm{A}_{\Gamma N_p} \\ \mathrm{A}_{1\Gamma} & \cdots & \mathrm{A}_{N_p \Gamma} & \mathrm{A}_{\Gamma\Gamma} \end{bmatrix} = \begin{bmatrix} \mathrm{L}_1 & & & \\ & \ddots & & \\ & & \mathrm{L}_{N_p} & \\ \mathrm{B}_1 & \cdots & \mathrm{B}_{N_p} & \mathrm{I} \end{bmatrix} \begin{bmatrix} \mathrm{U}_1 & & & \mathrm{C}_1 \\ & \ddots & & \vdots \\ & & \mathrm{U}_{N_p} & \mathrm{C}_{N_p} \\ & & & \mathrm{S} \end{bmatrix},$$

with

$$\mathrm{A}_{\ell\ell} = \mathrm{L}_\ell \, \mathrm{U}_\ell, \qquad \mathrm{B}_\ell = \mathrm{A}_{\ell\Gamma} \, \mathrm{U}_\ell^{-1}, \qquad \mathrm{C}_\ell = \mathrm{L}_\ell^{-1} \, \mathrm{A}_{\Gamma\ell}, \qquad \mathrm{S} = \mathrm{A}_{\Gamma\Gamma} - \sum_{\ell=1}^{N_p} \mathrm{B}_\ell \, \mathrm{C}_\ell$$

Block ILUT

Approximate LU decomposition of the block matrix A

$$\begin{bmatrix} \mathrm{A}_{11} & & & \mathrm{A}_{\Gamma 1} \\ & \ddots & & \vdots \\ & & \mathrm{A}_{N_pN_p} & \mathrm{A}_{\Gamma N_p} \\ \mathrm{A}_{1\Gamma} & \cdots & \mathrm{A}_{N_p\Gamma} & \mathrm{A}_{\Gamma\Gamma} \end{bmatrix} \approx \begin{bmatrix} \tilde{\mathrm{L}}_1 & & & \\ & \ddots & & \\ & & \tilde{\mathrm{L}}_{N_p} & \\ \tilde{\mathrm{B}}_1 & \cdots & \tilde{\mathrm{B}}_{N_p} & \mathrm{I} \end{bmatrix} \begin{bmatrix} \tilde{\mathrm{U}}_1 & & & \tilde{\mathrm{C}}_1 \\ & \ddots & & \vdots \\ & & \tilde{\mathrm{U}}_{N_p} & \tilde{\mathrm{C}}_{N_p} \\ & & & \tilde{\mathrm{S}} \end{bmatrix},$$

with

$$\mathrm{A}_{\ell\ell} = \mathrm{L}_\ell \, \mathrm{U}_\ell, \qquad \mathrm{B}_\ell = \mathrm{A}_{\ell\Gamma} \, \mathrm{U}_\ell^{-1}, \qquad \mathrm{C}_\ell = \mathrm{L}_\ell^{-1} \, \mathrm{A}_{\Gamma\ell}, \qquad \mathrm{S} = \mathrm{A}_{\Gamma\Gamma} - \sum_{\ell=1}^{N_p} \mathrm{B}_\ell \, \mathrm{C}_\ell$$

Let us replace L_ℓ and U_ℓ by their (local) ILUT factorizations (compute in parallel!)

$$\mathrm{A}_{\ell\ell} \approx \tilde{\mathrm{L}}_\ell \, \tilde{\mathrm{U}}_\ell, \qquad \tilde{\mathrm{B}}_\ell = \mathrm{A}_{\ell\Gamma} \, \tilde{\mathrm{U}}_\ell^{-1}, \qquad \tilde{\mathrm{C}}_\ell = \tilde{\mathrm{L}}_\ell^{-1} \, \mathrm{A}_{\Gamma\ell}, \qquad \tilde{\mathrm{S}} = \mathrm{A}_{\Gamma\Gamma} - \sum_{\ell=1}^{N_p} \tilde{\mathrm{B}}_\ell \, \tilde{\mathrm{C}}_\ell$$

I.C.L. Nievinski et al. Parallel implementation of a two-level algebraic ILU(k)-based domain decomposition preconditioner, TEMA (São Carlos) 19(1), Jan-Apr 2018

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

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

	$p=2$ # patches			$p=3$ # patches			$p=4$ # patches			$p=5$ # patches		
	4	16	64	4	16	64	4	16	64	4	16	64
$h=2^{-5}$	3(5)	4(7)	4(9)	3(5)	3(7)	4(11)	2(4)	2(6)	4(–)	2(4)	2(6)	–(–)
$h=2^{-6}$	3(5)	3(5)	4(7)	3(5)	3(7)	4(10)	3(6)	2(7)	3(11)	3(5)	3(7)	3(10)
$h=2^{-7}$	3(5)	3(5)	3(5)	3(5)	3(6)	3(8)	3(5)	2(6)	3(10)	–(5)	6(7)	3(11)

Numbers in parentheses correspond to global ILUT

R. Tielen *et al.* A block ILUT smoother for multipatch geometries in Isogeometric Analysis, To appear in: Springer INdAM Series, Springer, 2021

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

#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

	$p=2$			$p=3$			$p=4$			$p=5$		
	# patches			# patches			# patches			# patches		
	4	16	64	4	16	64	4	16	64	4	16	64
$h = 2^{-5}$	4(6)	4(8)	7(11)	3(6)	3(9)	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)
$h = 2^{-7}$	4(6)	4(6)	4(7)	3(6)	3(7)	3(8)	2(7)	3(7)	3(10)	4(6)	3(8)	3(12)

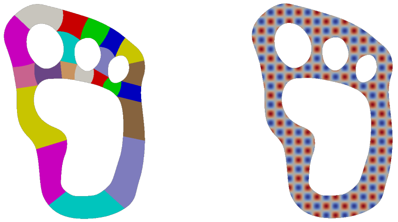
Numbers in parentheses correspond to global ILUT

R. Tielen *et al.* A block ILUT smoother for multipatch geometries in Isogeometric Analysis, To appear in: Springer INdAM Series, Springer, 2021

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

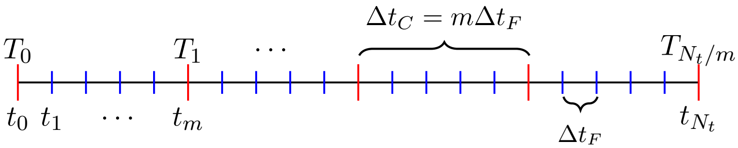
#4: Poisson's equation on the Yeti footprint

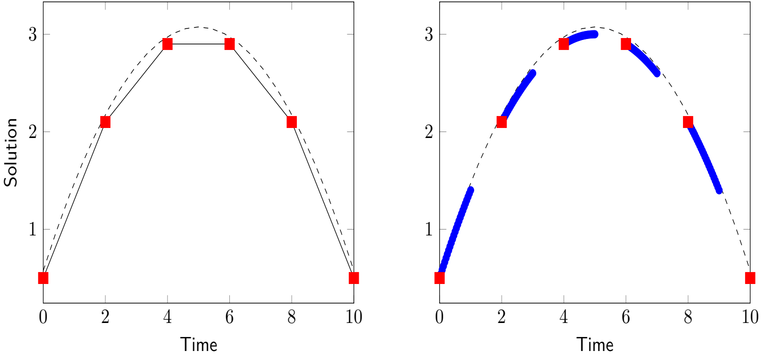
	$p=2$		$p=3$		$p=4$		$p=5$	
	block	global	block	global	block	global	block	global
$h=2^{-3}$	4	5	2	4	2	4	2	4
$h=2^{-4}$	4	8	3	5	3	5	2	4
$h=2^{-5}$	4	8	3	6	3	5	3	5



R. Tielen *et al.* A block ILUT smoother for multipatch geometries in Isogeometric Analysis, To appear in: Springer INdAM Series, Springer, 2021

Part II: Multigrid reduction in time (MGRIT)





S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16^{th} Copper Mountain Conference on Multigrid Methods 2013

Sketch of the MGRIT algorithm

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

$$[\mathrm{M}_{h,p} + \Delta t_F \; \mathrm{K}_{h,p}] \; \mathrm{u}_{h,p}^{n+1} = \mathrm{M}_{h,p} \, \mathrm{u}_{h,p}^{n} + \mathrm{f}_{h,p}$$

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16^{th} Copper Mountain Conference on Multigrid Methods 2013

Sketch of the MGRIT algorithm

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

$$[\,\mathrm{M}_{h,p} + \Delta t_F\ \mathrm{K}_{h,p}]\ \mathrm{u}_{h,p}^{n+1} = \mathrm{M}_{h,p}\,\mathrm{u}_{h,p}^{n} + \mathrm{f}_{h,p}$$

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

$$\mathrm{A}_{h,p}\,\mathrm{U}_{h,p} = \begin{bmatrix} \mathrm{I}_{h,p} & & & \\ -\Psi_{h,p}\,\mathrm{M}_{h,p} & \mathrm{I}_{h,p} & & \\ & \ddots & \ddots & \\ & & -\Psi_{h,p}\,\mathrm{M}_{h,p} & \mathrm{I}_{h,p} \end{bmatrix} \begin{bmatrix} \mathrm{u}_{h,p}^{0} \\ \mathrm{u}_{h,p}^{1} \\ \vdots \\ \mathrm{u}_{h,p}^{N_t} \end{bmatrix} = \Delta t_F \begin{bmatrix} \Psi_{h,p}\,\mathrm{f}_{h,p} \\ \Psi_{h,p}\,\mathrm{f}_{h,p} \\ \vdots \\ \Psi_{h,p}\,\mathrm{f}_{h,p} \end{bmatrix}$$

with

$$\Psi_{h,p} = [\,\mathrm{M}_{h,p} + \Delta t_F\ \mathrm{K}_{h,p}]^{-1}$$

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16^{th} Copper Mountain Conference on Multigrid Methods 2013

Sketch of the MGRIT algorithm, cont'd

Reordering of $\mathrm{A}_{h,p}$ into (F)ine and (C)oarse time levels yields

$$\begin{bmatrix} \mathrm{A}_{FF} & \mathrm{A}_{FC} \\ \mathrm{A}_{CF} & \mathrm{A}_{CC} \end{bmatrix} = \begin{bmatrix} \mathrm{I}_F & 0 \\ \mathrm{A}_{CF}\,\mathrm{A}_{FF}^{-1} & \mathrm{I}_C \end{bmatrix} \begin{bmatrix} \mathrm{A}_{FF} & 0 \\ 0 & \mathrm{S} \end{bmatrix} \begin{bmatrix} \mathrm{I}_F & \mathrm{A}_{FF}^{-1}\,\mathrm{A}_{FC} \\ 0 & \mathrm{I}_C \end{bmatrix}$$

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16^{th} Copper Mountain Conference on Multigrid Methods 2013

Sketch of the MGRIT algorithm, cont'd

Reordering of $\mathrm{A}_{h,p}$ into (F)ine and (C)oarse time levels yields

$$\begin{bmatrix} \mathrm{A}_{FF} & \mathrm{A}_{FC} \\ \mathrm{A}_{CF} & \mathrm{A}_{CC} \end{bmatrix} = \begin{bmatrix} \mathrm{I}_F & 0 \\ \mathrm{A}_{CF}\,\mathrm{A}_{FF}^{-1} & \mathrm{I}_C \end{bmatrix} \begin{bmatrix} \mathrm{A}_{FF} & 0 \\ 0 & \mathrm{S} \end{bmatrix} \begin{bmatrix} \mathrm{I}_F & \mathrm{A}_{FF}^{-1}\,\mathrm{A}_{FC} \\ 0 & \mathrm{I}_C \end{bmatrix}$$

with block-diagonal fine-level system matrix

$$\mathrm{A}_{FF} = \mathrm{I}_{N_t/m, N_t/m} \otimes \underbrace{\begin{pmatrix} \mathrm{I}_{h,p} & & & \\ -\Psi_{h,p}\,\mathrm{M}_{h,p} & \mathrm{I}_{h,p} & & \\ & \ddots & \ddots & \\ & & -\Psi_{h,p}\,\mathrm{M}_{h,p} & \mathrm{I}_{h,p} \end{pmatrix}}_{m \times m \text{ blocks}}$$

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16^{th} Copper Mountain Conference on Multigrid Methods 2013

Sketch of the MGRIT algorithm, cont'd

Reordering of $\mathrm{A}_{h,p}$ into (F)ine and (C)oarse time levels yields

$$\begin{bmatrix} \mathrm{A}_{FF} & \mathrm{A}_{FC} \\ \mathrm{A}_{CF} & \mathrm{A}_{CC} \end{bmatrix} = \begin{bmatrix} \mathrm{I}_{F} & 0 \\ \mathrm{A}_{CF}\,\mathrm{A}_{FF}^{-1} & \mathrm{I}_{C} \end{bmatrix} \begin{bmatrix} \mathrm{A}_{FF} & 0 \\ 0 & \mathrm{S} \end{bmatrix} \begin{bmatrix} \mathrm{I}_{F} & \mathrm{A}_{FF}^{-1}\,\mathrm{A}_{FC} \\ 0 & \mathrm{I}_{C} \end{bmatrix}$$

with block-diagonal fine-level system matrix

$$\mathrm{A}_{FF} = \mathrm{I}_{N_t/m,N_t/m} \otimes \underbrace{\begin{pmatrix} \mathrm{I}_{h,p} & & & \\ -\Psi_{h,p}\,\mathrm{M}_{h,p} & \mathrm{I}_{h,p} & & \\ & \ddots & \ddots & \\ & & -\Psi_{h,p}\,\mathrm{M}_{h,p} & \mathrm{I}_{h,p} \end{pmatrix}}_{m\times m \text{ blocks}}$$

and the Schur complement $\mathrm{S} = \mathrm{A}_{CC} - \mathrm{A}_{CF}\,\mathrm{A}_{FF}^{-1}\,\mathrm{A}_{FC}$

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16^{th} Copper Mountain Conference on Multigrid Methods 2013

Sketch of the MGRIT algorithm, cont'd

Approximate the Schur complement

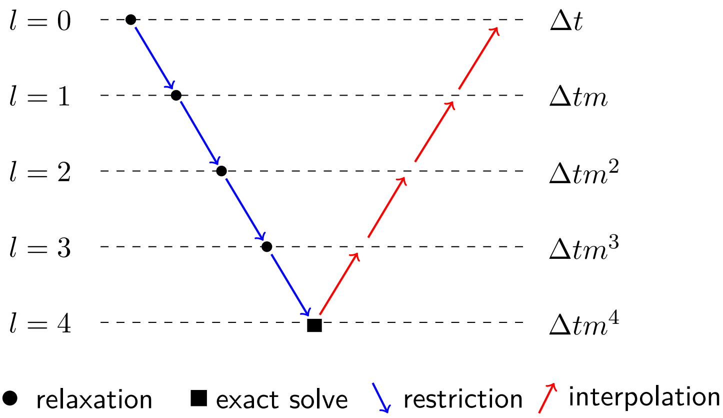
$$\mathrm{S} = \begin{bmatrix} \mathrm{I} & & & \\ -(\Psi_{h,p}\,\mathrm{M}_{h,p})^m & \mathrm{I} & & \\ & \ddots & \ddots & \\ & & -(\Psi_{h,p}\,\mathrm{M}_{h,p})^m & \mathrm{I} \end{bmatrix} \approx \begin{bmatrix} \mathrm{I} & & & \\ -\Phi_{h,p}\,\mathrm{M}_{h,p} & \mathrm{I} & & \\ & \ddots & \ddots & \\ & & -\Phi_{h,p}\,\mathrm{M}_{h,p} & \mathrm{I} \end{bmatrix}$$

with *coarse integrator*

$$\Phi_{h,p} = \left[\,\mathrm{M}_{h,p} + \Delta t_C\ \mathrm{K}_{h,p}\right]^{-1}$$

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16^{th} Copper Mountain Conference on Multigrid Methods 2013

The MGRIT-IGA V-cycle



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

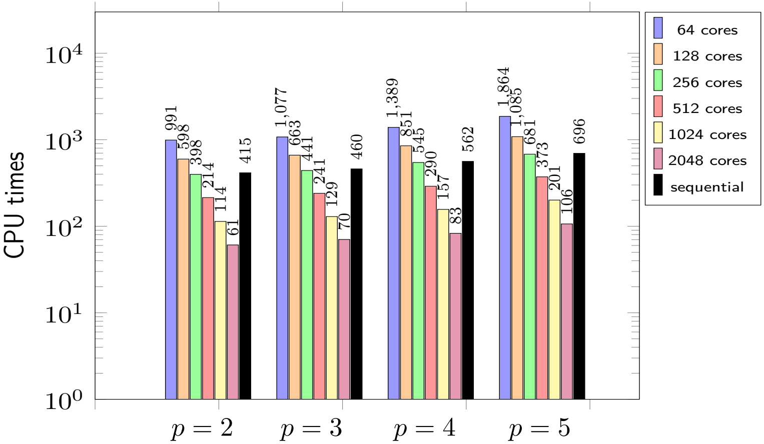
- open-source implementation of the optimal-scaling multigrid solver in MPI/C with C++ interface)
- extendable by overloading callback functions

Try it yourself

`https://github.com/gismo/gismo/tree/xbraid/extensions/gsXBraid`

Numerical examples: *Strong scaling of MGRIT-IGA*

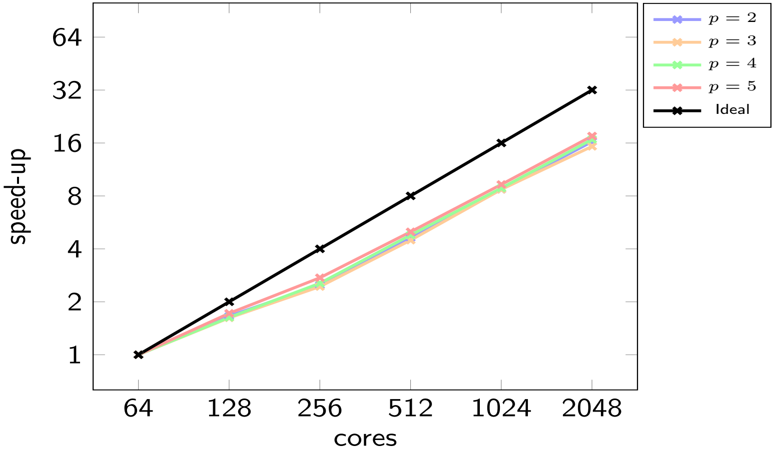
#5: Heat-Eq with $h = 2^{-6}$ spatial resolution solved for $N_t = 10.000$ time steps with backward Euler method on 128 Xeon Gold 6130 CPUs (2.10GHz, 96GB, 16 cores)



R. Tielen *et al.* 2021, arXiv:2107.05337

Numerical examples: *Speed-up of MGRIT-IGA*

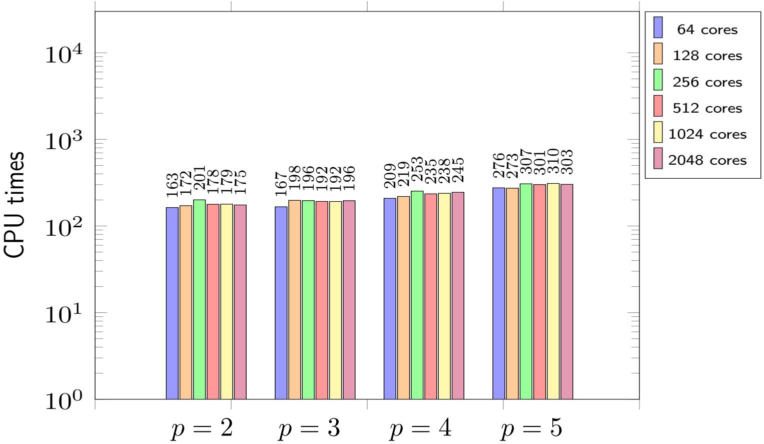
#5: Heat-Eq with $h = 2^{-6}$ spatial resolution solved for $N_t = 10.000$ time steps with backward Euler method on 128 Xeon Gold 6130 CPUs (2.10GHz, 96GB, 16 cores)



R. Tielen *et al.* 2021, arXiv:2107.05337

Numerical examples: *Weak scaling of MGRIT-IGA*

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

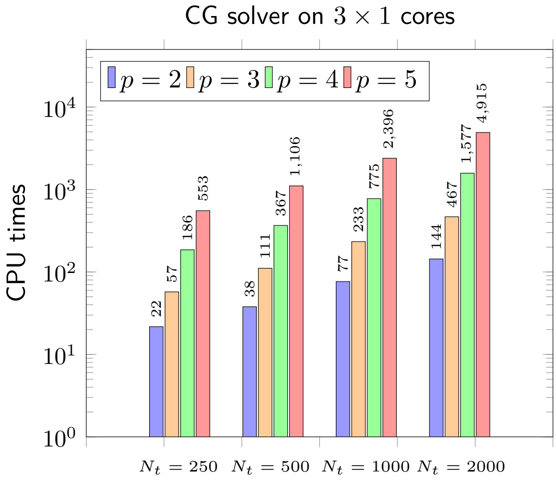


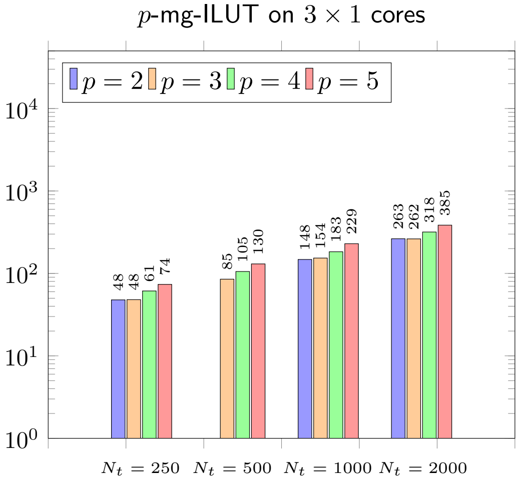
R. Tielen *et al.* 2021, arXiv:2107.05337

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?

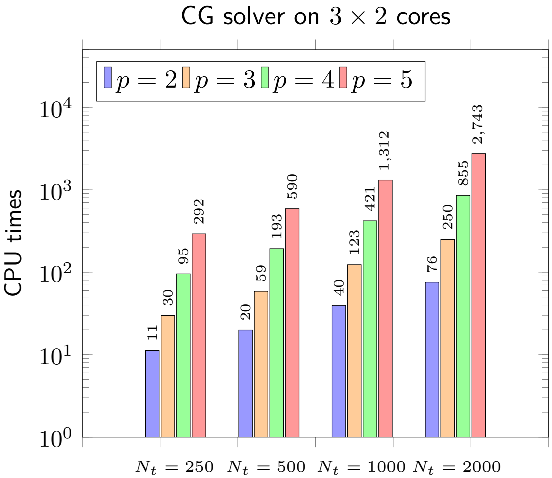
No!

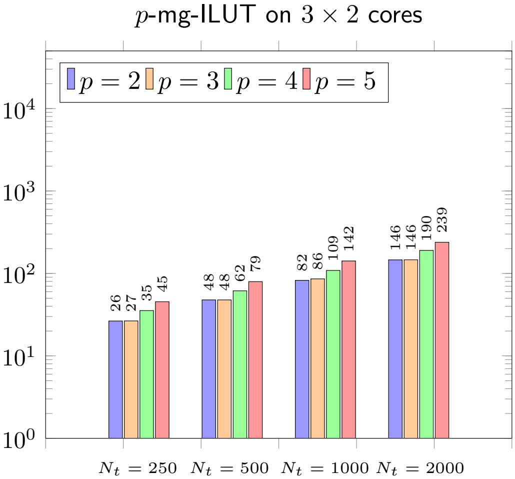




Do we really need p-multigrid or would a standard solver be good enough?

No!





Further reading on IGA solvers

R. Tielen, M. Möller, D. Göddeke and C. Vuik: *p-multigrid methods and their comparison to h-multigrid methods within Isogeometric Analysis*, CMAME 372 (2020)

R. Tielen, M. Möller and C. Vuik: *A block ILUT smoother for multipatch geometries in Isogeometric Analysis*, In: Springer INdAM Series, Springer, 2021

R. Tielen, M. Möller and C. Vuik: *Multigrid Reduced in Time for Isogeometric Analysis*, Submitted to: Proceedings of the Young Investigators Conference 2021.

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

R. Tielen: *p-multigrid methods for isogeometric analysis*, doctoral thesis, TU Delft, to be defended in Oct. 2021

Analysis-suitable parametrizations: PDE-based parametrization techniques

Notation

Jacobian of the push-forward mapping

$$J = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix}$$

Metric tensor of the push-forward mapping

$$G = J^\top J \begin{pmatrix} \mathbf{x}_\xi \cdot \mathbf{x}_\xi \cdot & \mathbf{x}_\xi \cdot \mathbf{x}_\eta \\ \mathbf{x}_\eta \cdot \mathbf{x}_\xi \cdot & \mathbf{x}_\eta \cdot \mathbf{x}_\eta \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix}$$

Problem formulation

Let $\hat{\Omega} = [0,1]^2$ and $\mathbf{x}(\xi,\eta) = (x(\xi,\eta), y(\xi,\eta))^\top$, $\mathbf{x} : \hat{\Omega} \rightarrow \Omega \subset \mathbb{R}^2$. Furthermore, let a homeomorphic boundary correspondence $\mathbf{x}|_{\hat{\Gamma}} = \Gamma$ with $\hat{\Gamma} := \partial\hat{\Omega}$ and $\Gamma := \partial\Omega$ be given.

The aim is to extend the mapping into the interior such that it is bijective and (optionally) satisfies additional ‘grid’ quality criteria (e.g. local orthogonality of grid lines).

Solution strategy

Let us represent the geometry mapping in terms of a given[1] B-spline basis $\mathcal{B}_{\mathbf{h},\mathbf{p}}$

$$\mathbf{x}_{\mathbf{h},\mathbf{p}}(\xi,\eta) = \sum_{j\in\mathcal{J}_B} \mathbf{x}_j \hat{\varphi}_j(\xi,\eta) + \sum_{j\in\mathcal{J}_I} \mathbf{x}_j \hat{\varphi}_j(\xi,\eta), \qquad \mathbf{x}_j \in \mathbb{R}^2, \quad \hat{\varphi}_j \in \mathcal{B}_{\mathbf{h},\mathbf{p}}$$

Here, $\mathcal{J}_B$ and $\mathcal{J}_I$ are index sets that identify the basis functions that vanish and do not vanish at the boundary, respectively. Formally, $\mathcal{J}_B \cap \mathcal{J}_I = \emptyset$ and $\mathcal{J}_B \cap \mathcal{J}_I = \{1, 2, \dots, N_b\}$.

[1] We will see that the 'right' basis is constructed step by step via adaptive local refinement

Solution strategy

Let us represent the geometry mapping in terms of a given[1] B-spline basis $\mathcal{B}_{\mathbf{h},\mathbf{p}}$

$$\mathbf{x}_{\mathbf{h},\mathbf{p}}(\xi,\eta) = \sum_{j \in \mathcal{J}_B} \mathbf{x}_j \hat{\varphi}_j(\xi,\eta) + \sum_{j \in \mathcal{J}_I} \mathbf{x}_j \hat{\varphi}_j(\xi,\eta), \qquad \mathbf{x}_j \in \mathbb{R}^2, \quad \hat{\varphi}_j \in \mathcal{B}_{\mathbf{h},\mathbf{p}}$$

Here, $\mathcal{J}_B$ and $\mathcal{J}_I$ are index sets that identify the basis functions that vanish and do not vanish at the boundary, respectively. Formally, $\mathcal{J}_B \cap \mathcal{J}_I = \emptyset$ and $\mathcal{J}_B \cap \mathcal{J}_I = \{1, 2, \dots, N_b\}$.

In a first step, we will determine the position of the **boundary control points** $\mathbf{x}_j$, $j \in \mathcal{I}_B$ such that $\mathbf{x}_{\mathbf{h},\mathbf{p}}(\xi,\eta)|_{\hat{\Gamma}} = \Gamma$ (homeomorphic boundary correspondence).

[1] We will see that the 'right' basis is constructed step by step via adaptive local refinement

Solution strategy

Let us represent the geometry mapping in terms of a given[1] B-spline basis $\mathcal{B}_{\mathbf{h},\mathbf{p}}$

$$\mathbf{x}_{\mathbf{h},\mathbf{p}}(\xi,\eta) = \sum_{j \in \mathcal{J}_B} \mathbf{x}_j \hat{\varphi}_j(\xi,\eta) + \sum_{j \in \mathcal{J}_I} \mathbf{x}_j \hat{\varphi}_j(\xi,\eta), \qquad \mathbf{x}_j \in \mathbb{R}^2, \quad \hat{\varphi}_j \in \mathcal{B}_{\mathbf{h},\mathbf{p}}$$

Here, $\mathcal{J}_B$ and $\mathcal{J}_I$ are index sets that identify the basis functions that vanish and do not vanish at the boundary, respectively. Formally, $\mathcal{J}_B \cap \mathcal{J}_I = \emptyset$ and $\mathcal{J}_B \cap \mathcal{J}_I = \{1, 2, \ldots, N_b\}$.

In a first step, we will determine the position of the **boundary control points** $\mathbf{x}_j$, $j \in \mathcal{I}_B$ such that $\mathbf{x}_{\mathbf{h},\mathbf{p}}(\xi,\eta)|_{\hat{\Gamma}} = \Gamma$ (homeomorphic boundary correspondence).

In a second step, we will extend the geometry mapping into the interior, that is, we will determine the position of the **inner control points** $\mathbf{x}_j$, $j \in \mathcal{I}_I$ such that the resulting mapping is bijective and (optionally) satisfies additional 'grid' quality criteria.

[1] We will see that the 'right' basis is constructed step by step via adaptive local refinement

Step 1: constructing B-spline boundary curves from analytic contours

Let the boundaries of $\hat{\Omega}$ and Ω be subdivided into four segments $\mathcal{S} = \{\mathsf{n}, \mathsf{s}, \mathsf{w}, \mathsf{e}\}$

$$\bigcup_{\alpha \in \mathcal{S}} \overline{\gamma}^{\alpha} = \hat{\Gamma} \quad \text{and} \quad \bigcup_{\alpha \in \mathcal{S}} \overline{\Gamma}^{\alpha} = \Gamma$$

with corresponding homeomorphic **boundary transformations**

$$\mathbf{f}^{\alpha} : \overline{\gamma}^{\alpha} \to \overline{\Gamma}^{\alpha}, \quad \alpha \in \mathcal{S}$$

Furthermore, let

$$\Xi_{\xi}^{\alpha} = \left[\xi_1^{\alpha}, \xi_2^{\alpha}, \ldots, \xi_{N_{\xi}+p_{\xi}+1}^{\alpha}\right] \quad \text{and} \quad \Xi_{\eta}^{\alpha} = \left[\eta_1^{\alpha}, \eta_2^{\alpha}, \ldots, \eta_{N_{\xi}+p_{\eta}+1}^{\alpha}\right]$$

be uniform open knot vectors for the north/south and east/west boundary curves, respectively, and $\mathcal{V}_{h,p}^{\alpha}$ the corresponding one-dimensional B-spline spaces.

Step 1: constructing B-spline boundary curves from analytic contours

For each $\alpha \in \mathcal{S}$ we individually do the following:

- $L_2(\hat{\Gamma}^\alpha)$-project $\mathbf{f}^\alpha$ onto $\mathcal{V}_{h,p}^\alpha$ with the two corner points constrained to the values of $\mathbf{f}^\alpha(0)$ and $\mathbf{f}^\alpha(1)$ to obtain the initial B-spline curve $\mathbf{f}_{h,p}^\alpha$

Step 1: constructing B-spline boundary curves from analytic contours

For each $\alpha \in \mathcal{S}$ we individually do the following:

- $L_2(\hat{\Gamma}^\alpha)$-project $\mathbf{f}^\alpha$ onto $\mathcal{V}_{h,p}^\alpha$ with the two corner points constrained to the values of $\mathbf{f}^\alpha(0)$ and $\mathbf{f}^\alpha(1)$ to obtain the initial B-spline curve $\mathbf{f}_{h,p}^\alpha$
- define the element-wise average residual function as **error indicator**

$$E(\mathbf{f}_{h,p}^\alpha) = \sum_{e_k \in \overline{\gamma}^\alpha} \frac{1}{|e_k|} \int_{e_k} \|\mathbf{f}_{h,p}^\alpha(t) - \mathbf{f}^\alpha(t)\|^2 \, \mathrm{d}t$$

where e_k denotes a one-dimensional element on $\hat{\Gamma}$ (defined by the unique values of the knot vector) and $|e_k|$ its length in the parametric domain.

Step 1: constructing B-spline boundary curves from analytic contours

For each $\alpha \in \mathcal{S}$ we individually do the following:

- $L_2(\hat{\Gamma}^\alpha)$-project $\mathbf{f}^\alpha$ onto $\mathcal{V}_{h,p}^\alpha$ with the two corner points constrained to the values of $\mathbf{f}^\alpha(0)$ and $\mathbf{f}^\alpha(1)$ to obtain the initial B-spline curve $\mathbf{f}_{h,p}^\alpha$
- define the element-wise average residual function as **error indicator**

$$E(\mathbf{f}_{h,p}^\alpha) = \sum_{e_k \in \overline{\gamma}^\alpha} \frac{1}{|e_k|} \int_{e_k} \|\mathbf{f}_{h,p}^\alpha(t) - \mathbf{f}^\alpha(t)\|^2 \, \mathrm{d}t$$

  where e_k denotes a one-dimensional element on $\hat{\Gamma}$ (defined by the unique values of the knot vector) and $|e_k|$ its length in the parametric domain.

- Elements for which the above error indicator is too large are refined by adding an additional knot in the center. The projection and refinement steps are then repeated until all elements are sufficiently accurate or a maximum refinement level is reached.

Step 1: constructing B-spline boundary curves from point clouds

In many engineering applications, boundaries are given as ordered sets of points, i.e.

$$\mathbf{P}^{\alpha} = \{\mathbf{p}_1^{\alpha}, \mathbf{p}_2^{\alpha}, \dots \mathbf{p}_{M^{\alpha}}^{\alpha}\}, \qquad \alpha \in \mathcal{S}$$

For each of the four boundary segments, let us recursively define

$$l_i^{\alpha} := l_{i-1}^{\alpha} + \|\mathbf{p}_i - \mathbf{p}_{i-1}\|, \qquad i = 2, 3, \dots, M^{\alpha}$$

starting at $l_1^{\alpha} = 0$. From that we compute the arc-length parametrized sequence

$$t_i^{\alpha} := \frac{l_i^{\alpha}}{l_{M^a}^{\alpha}}, \qquad i = 1, 2, \dots, M^{\alpha}$$

Furthermore, let

$$\mathbf{X}_B := \{\mathbf{x}_j \ : \ j \in \mathcal{J}_B\}, \qquad \mathbf{m}^{\alpha}(t) := \begin{cases} (t, 1) & \text{if } \alpha = \mathrm{n} \\ (t, 0) & \text{if } \alpha = \mathrm{s} \\ (0, t) & \text{if } \alpha = \mathrm{w} \\ (1, t) & \text{if } \alpha = \mathrm{e} \end{cases}$$

Step 1: constructing B-spline boundary curves from point clouds

The aim is to select $\mathbf{x}_j \in \mathbf{X}_B$ such that $\mathbf{x}_{\mathbf{h},\mathbf{p}}(\mathbf{m}^\alpha(t_i^\alpha)) \simeq \mathbf{p}_i^\alpha$ *at* the parametric values.

We perform a **least-squares regression** (possibly with stabilization) to minimize

$$R(\Gamma, \mathbf{X}_B) = \frac{1}{2} \sum_{\alpha \in \mathcal{S}} \left(\sum_{i=1}^{M^\alpha} \left\| \mathbf{x}_{\mathbf{h},\mathbf{p}}(\mathbf{m}^\alpha(t_i^\alpha)) - \mathbf{p}_i^\alpha \right\|^2 + \gamma \int_{\gamma^\alpha} \left\| \partial_{\mathbf{t}} \mathbf{x}_{\mathbf{h},\mathbf{p}} \right\|^2 \, \mathrm{d}\gamma \right)$$

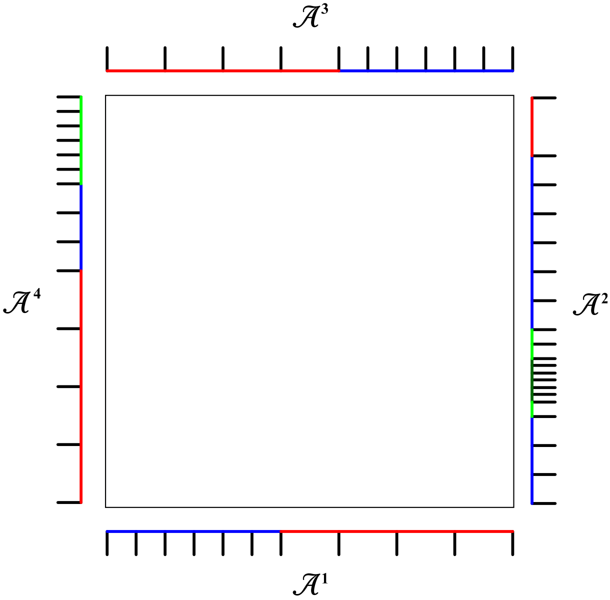
where $\gamma > 0$ is a parameter and $\partial_{\mathbf{t}}$ denotes the directional derivative in tangential direction.

The mismatch

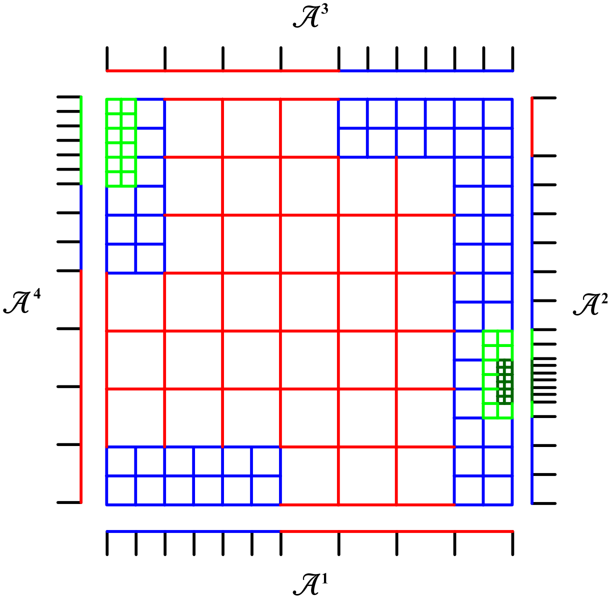
$$r_i^\alpha = \left\| \mathbf{x}_{\mathbf{h},\mathbf{p}}(\mathbf{m}^\alpha(t_i^\alpha)) - \mathbf{p}_i^\alpha \right\|$$

is used as error indicator. If $r_i^\alpha > \text{tol}$ we insert an additional knot $\xi_{l+\frac{1}{2}}^\alpha$ at the center of the knot span $[\xi_l^\alpha, \xi_{l+1}^\alpha] \subset \Xi_\xi^\alpha$ that contains the parameter value t_i^α.

Step 1: result



Step 1: result



Overview of methods to compute inner control points (step 2)

- Algebraic methods
- Optimization based methods
  - convex/non-convex cost function
  - unconstrained/constrained optimization
- PDE based methods
  - parabolic, hyperbolic, and elliptic schemes

Overview of methods to compute inner control points (step 2)

- Algebraic methods
- Optimization based methods
  - convex/non-convex cost function
  - unconstrained/constrained optimization
- PDE based methods
  - parabolic, hyperbolic, and elliptic schemes

Algebraic methods

Idea: generate a mapping by evaluating a closed-form expression

Coon's patch approach

$$\begin{aligned}\mathbf{x}(\xi,\eta) &= (1-\xi)\mathbf{x}(0,\eta)+\xi\mathbf{x}(1,\eta)\\ &+ (1-\eta)\mathbf{x}(\xi,0)+\eta\mathbf{x}(\xi,1)\\ &- \begin{bmatrix}1-\xi & \xi\end{bmatrix}\begin{pmatrix}\mathbf{x}(0,0) & \mathbf{x}(0,1)\\ \mathbf{x}(1,0) & \mathbf{x}(1,1)\end{pmatrix}\begin{pmatrix}1-\eta\\ \eta\end{pmatrix}\end{aligned}$$

There is no guarantee that the resulting mapping is bijective, that is, free of foldings.

Optimization based methods

Idea: generate a mapping by solving the minimization problem

$$\int_{\hat{\Omega}} \alpha_1 Q_1(\mathbf{x}) + \alpha_2 Q_2(\mathbf{x}) + \ldots \, \mathrm{d}\boldsymbol{\xi} \rightarrow \min_{\mathbf{x}\in\hat{\Omega}} \quad \text{s.t.} \quad \mathbf{x}|_{\hat{\Gamma}} = \Gamma$$

where $\alpha_k \geq 0$ and the cost functions $Q_k(\mathbf{x})$ are as follows:

$$\begin{aligned}
\text{length:}\quad & Q(\mathbf{x}) = \|\mathbf{x}_\xi\|^2 + \|\mathbf{x}_\eta\|^2 \\
\text{uniformity:}\quad & Q(\mathbf{x}) = \|\mathbf{x}_{\xi\xi}\|^2 + 2\|\mathbf{x}_{\xi\eta}\|^2 + \|\mathbf{x}_{\eta\eta}\|^2 \\
\text{Liao:}\quad & Q(\mathbf{x}) = g_{11}^2 + 2g_{12}^2 + g_{22}^2 \\
\text{area:}\quad & Q(\mathbf{x}) = \det J^2 \\
\text{(area) orthogonality:}\quad & Q(\mathbf{x}) = g_{11}g_{22} \quad \text{or} \quad Q(\mathbf{x}) = g_{12}^2 \\
\text{eccentricity:}\quad & Q(\mathbf{x}) = \left(\frac{\mathbf{x}_\xi \cdot \mathbf{x}_{\xi\xi}}{g_{11}}\right)^2 + \left(\frac{\mathbf{x}_\eta \cdot \mathbf{x}_{\eta\eta}}{g_{22}}\right)^2
\end{aligned}$$

Again, there is no guarantee that the resulting mapping is bijective.

Optimization methods, cont'd

Penalization: a possible remedy to mitigate grid folding is to impose an infinite barrier close to the boundary of the feasibility region, e.g.

$$\text{Winslow: } Q(\mathbf{x}) = \frac{g_{11} + g_{22}}{\det J}$$

Optimization methods, cont'd

Penalization: a possible remedy to mitigate grid folding is to impose an infinite barrier close to the boundary of the feasibility region, e.g.

$$\text{Winslow: } Q(\mathbf{x}) = \frac{g_{11} + g_{22}}{\det J}$$

Constrained optimization: another approach is to augment the optimization problem with constraints that ensure that the resulting mapping is bijective (non-trivial!).

Optimization methods, cont'd

Penalization: a possible remedy to mitigate grid folding is to impose an infinite barrier close to the boundary of the feasibility region, e.g.

$$\text{Winslow:}\ Q(\mathbf{x}) = \frac{g_{11} + g_{22}}{\det J}$$

Constrained optimization: another approach is to augment the optimization problem with constraints that ensure that the resulting mapping is bijective (non-trivial!).

[Hinz et al. 2020]: in the context of IGA $\det J$ is a piecewise-polynomial function of higher polynomial degree that can be projected onto a spline basis that contains it.
A *sufficient condition* for $\det J > 0$ is that all coefficients of the basis expansion are positive (as B-spline basis functions are strictly positive on their support).

However, we need an initial guess that already satisfies the constraint.

Elliptic grid generation (EGG)

Instead of starting from the push-forward mapping $\mathbf{x} : \hat{\Omega} \to \Omega \subset \mathbb{R}^2$ let us consider the inverse mapping $\mathbf{x}^{-1} = \boldsymbol{\xi} : \Omega \to \hat{\Omega}$ and impose the **Laplace problem**

$$\Delta \boldsymbol{\xi} = 0 \quad \text{in } \Omega \quad \text{s.t.} \quad \boldsymbol{\xi}|_{\Gamma} = \hat{\Gamma}$$

Elliptic grid generation (EGG)

Instead of starting from the push-forward mapping $\mathbf{x} : \hat{\Omega} \to \Omega \subset \mathbb{R}^2$ let us consider the inverse mapping $\mathbf{x}^{-1} = \boldsymbol{\xi} : \Omega \to \hat{\Omega}$ and impose the **Laplace problem**

$$\Delta \boldsymbol{\xi} = 0 \quad \text{in } \Omega \quad \text{s.t.} \quad \boldsymbol{\xi}|_{\Gamma} = \hat{\Gamma}$$

Assuming that $\Omega \subset \mathbb{R}^2$ is an open, simply connected domain that is topologically equivalent to $\hat{\Omega} = [0, 1]^2$ and a homeomorphic boundary correspondence $\boldsymbol{\xi}|_{\Gamma} = \hat{\Gamma}$ is given one can show (Chap. 9, Castillo: Mathematical Aspects of Numerical Grid Generation, SIAM 1991) that the exact solution is bijective. This only holds for planar parametrizations and only if the target domain is convex (that's why we start with the pull-back mapping).

Elliptic grid generation (EGG)

Instead of starting from the push-forward mapping $\mathbf{x} : \hat{\Omega} \to \Omega \subset \mathbb{R}^2$ let us consider the inverse mapping $\mathbf{x}^{-1} = \boldsymbol{\xi} : \Omega \to \hat{\Omega}$ and impose the **Laplace problem**

$$\Delta \boldsymbol{\xi} = 0 \quad \text{in } \Omega \quad \text{s.t.} \quad \boldsymbol{\xi}|_{\Gamma} = \hat{\Gamma}$$

Assuming that $\Omega \subset \mathbb{R}^2$ is an open, simply connected domain that is topologically equivalent to $\hat{\Omega} = [0,1]^2$ and a homeomorphic boundary correspondence $\boldsymbol{\xi}|_{\Gamma} = \hat{\Gamma}$ is given one can show (Chap. 9, Castillo: Mathematical Aspects of Numerical Grid Generation, SIAM 1991) that the exact solution is bijective. This only holds for planar parametrizations and only if the target domain is convex (that's why we start with the pull-back mapping).

Let is invert the above problem and scale it to obtain a **nonlinear elliptic problem**

$$\mathcal{L}(\mathbf{x}) := \frac{g_{22}\mathbf{x}_{\xi\xi} - 2g_{12}\mathbf{x}_{\xi\eta} + g_{11}\mathbf{x}_{\eta\eta}}{g_{11} + g_{22} + \epsilon} = 0 \quad \text{in } \hat{\Omega} \quad \text{s.t.} \quad \mathbf{x}|_{\hat{\Gamma}} = \Gamma$$

Since $g_{11} \geq 0$ and $g_{22} \geq 0$ the denominator is strictly positive for some parameter $\epsilon > 0$.

Elliptic grid generation (EGG), cont'd

Variational problem find $\mathbf{x} \in \mathcal{V}_{\Gamma}^2 := \{\mathbf{w} \in \mathcal{H}^2(\hat{\Omega}) \times \mathcal{H}^2(\hat{\Omega}) \;:\; \mathbf{w}|_{\hat{\Gamma}} = \Gamma\}$ such that

$$\int_{\hat{\Omega}} \boldsymbol{\sigma} \cdot \mathcal{L}(\mathbf{x}) \, \mathrm{d}\boldsymbol{\xi} = 0 \qquad \forall \boldsymbol{\sigma} \in \mathcal{V}_{\mathbf{0}}^2$$

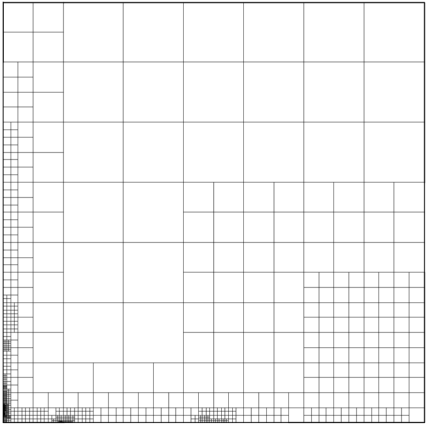
The discretized version is obtained by letting $\mathcal{V}_{\mathbf{h},\mathbf{p},\{\Gamma,\mathbf{0}\}}^2 := [\text{span}\,\mathcal{B}_{\mathbf{h},\mathbf{p}}]^2 + \text{b.c.} \approx \mathcal{V}_{\{\Gamma,\mathbf{0}\}}^2$ for a B-spline basis $\mathcal{B}_{\mathbf{h},\mathbf{p}}$ that is at least C^1 and seeking $\mathbf{x}_{\mathbf{h},\mathbf{p}} \in \mathcal{V}_{\mathbf{h},\mathbf{p},\Gamma}^2$ such that

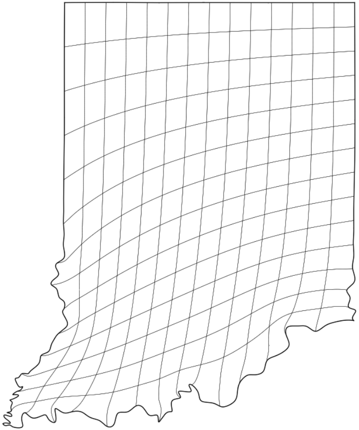
$$\int_{\hat{\Omega}} \boldsymbol{\sigma}_{\mathbf{h},\mathbf{p}} \cdot \mathcal{L}(\mathbf{x}_{\mathbf{h},\mathbf{p}}) \, \mathrm{d}\boldsymbol{\xi} = 0 \qquad \forall \boldsymbol{\sigma}_{\mathbf{h},\mathbf{p}} \in \mathcal{V}_{\mathbf{h},\mathbf{p},\mathbf{0}}^2$$

This is a **nonlinear** problem for the *inner* control points $\mathbf{x}_j$, $j \in \mathcal{J}_I$ as the boundary control points $\mathbf{x}_j$, $j \in \mathcal{J}_B$ are fixed through the Dirichlet boundary condition $\mathbf{x}_{\mathbf{h},\mathbf{p}}|_{\hat{\Gamma}} = \Gamma_{\mathbf{h},\mathbf{p}}$.

We solve this root-finding problem with a Newton-type algorithm combined with a multigrid solver to speed up convergence. Details can be found in the doctoral thesis by Jochen Hinz.

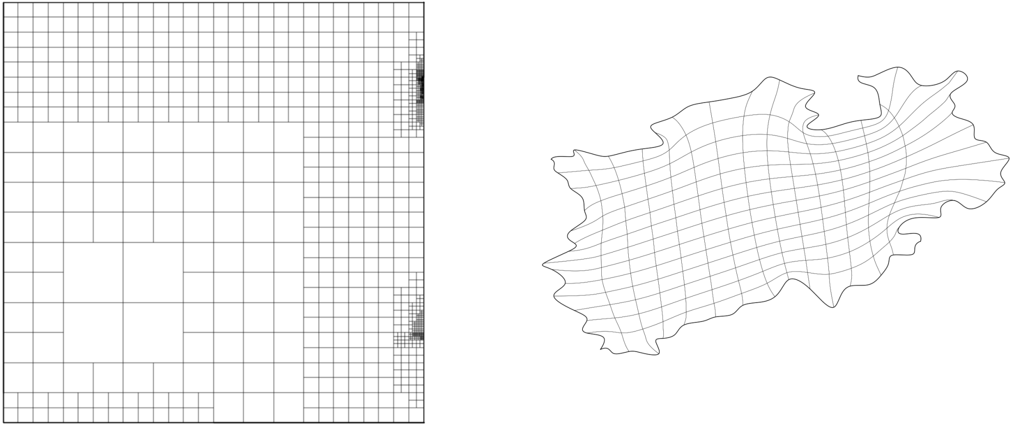
Examples





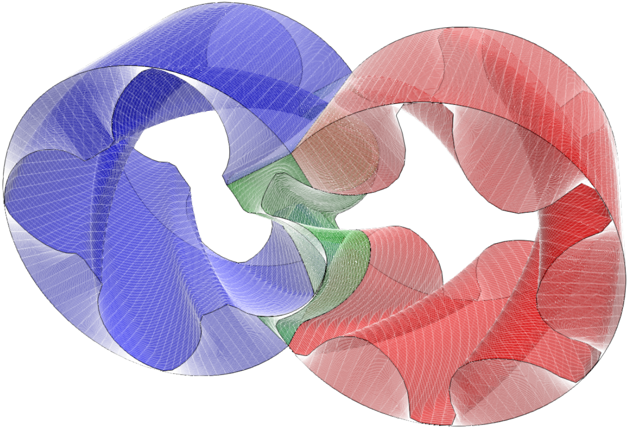
Parametrization of the U.S. state of Indiana with 2338 bicubic THB-spline basis functions.

Examples



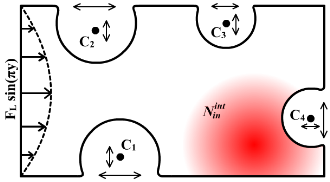
Parametrization of North-Rhine Westphalia with 2676 bicubic THB-spline basis functions.

Examples



Three-patch parametrization of the fluid passage of the twin-screw rotary compressor with tensor-product bicubic+linear B-splines with special treatment for C^0 multi-patch coupling.

Design optimization of a cooling element



Design parameters

$$\boldsymbol{\lambda} = (x_k, y_k, r_k), \qquad k = 1, 2, 3, 4$$

Governing equation

$$-\kappa \Delta u^{\boldsymbol{\lambda}}(\mathbf{x}) + 10^{-3} u^{\boldsymbol{\lambda}}(\mathbf{x}) = A \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0\|}{2\sigma^2}\right) \qquad \text{in } \Omega^{\boldsymbol{\lambda}}$$

$$\kappa\, \partial_n u^{\boldsymbol{\lambda}}(\mathbf{x}) = \begin{cases} -h_{\text{cooling}} + F_L \sin(\pi y) & \text{on } \Gamma_L^{\boldsymbol{\lambda}} \\ -h_{\text{cooling}} & \text{on } \Gamma^{\boldsymbol{\lambda}} \setminus \Gamma_L^{\boldsymbol{\lambda}} \end{cases}$$

with

$$-h_{\text{cooling}} = \sum_{k=1}^{4} \frac{r_k^3}{20\|\mathbf{x} - \mathbf{x}_k\|^2} (u^{\boldsymbol{\lambda}}(\mathbf{x}) - T_\infty)$$

All details can be found in: J. Hinz *et al.* The role of PDE-based parameterization techniques in gradient-based IGA shape optimization applications. CMAME 378, 113685, 2021.

Design optimization of a cooling element, cont'd

The aim is to minimize the 'idealized manufacturing costs' of the cooling element such that the heat source temperature $T^{\boldsymbol{\lambda}}(u^{\boldsymbol{\lambda}}, \Omega^{\boldsymbol{\lambda}})$ does not exceed the upper bound $T_{\max} = 80$.

Optimization problem

$$J(u^{\boldsymbol{\lambda}}, \Omega^{\boldsymbol{\lambda}}, \boldsymbol{\lambda}) := \int_{\Omega^{\boldsymbol{\lambda}}} 1 \,\mathrm{d}S + \sum_{k=1}^{4} \frac{100 r_k^2}{\pi} \to \min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \quad \text{s.t.} \quad T_{\max} - T^{\boldsymbol{\lambda}} \geq 0$$

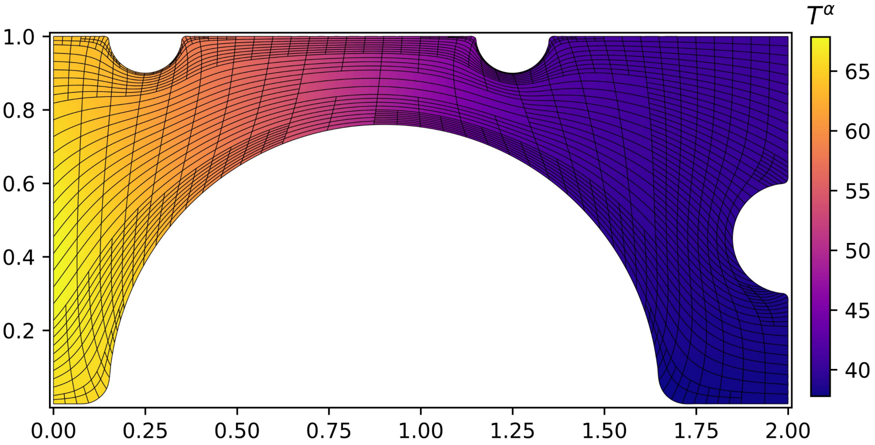
where $\boldsymbol{\Lambda}$ is the space of all 'admissible' designs (30 additional inequalities), i.e. the active coolers do not overlap and the genus of $\Omega^{\boldsymbol{\lambda}}$ does not change (no topology change).

Solution strategy

$$\boldsymbol{\lambda}^{\ell} \xrightarrow{\text{EGG}} \Omega^{\boldsymbol{\lambda}^{\ell}}_{\mathbf{h},\mathbf{p}} \xrightarrow{\text{IGA}} u^{\boldsymbol{\lambda}^{\ell}}_{\tilde{\mathbf{h}},\tilde{\mathbf{p}}} \xrightarrow{\text{evaluate}} J(u^{\boldsymbol{\lambda}^{\ell}}_{\tilde{\mathbf{h}},\tilde{\mathbf{p}}}, \Omega^{\boldsymbol{\lambda}^{\ell}}_{\mathbf{h},\mathbf{p}}, \Omega^{\boldsymbol{\lambda}^{\ell}}_{\mathbf{h},\mathbf{p}}) \xrightarrow{\text{compute}} \frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} \xrightarrow{\text{IPOPT}} \boldsymbol{\lambda}^{\ell+1}$$

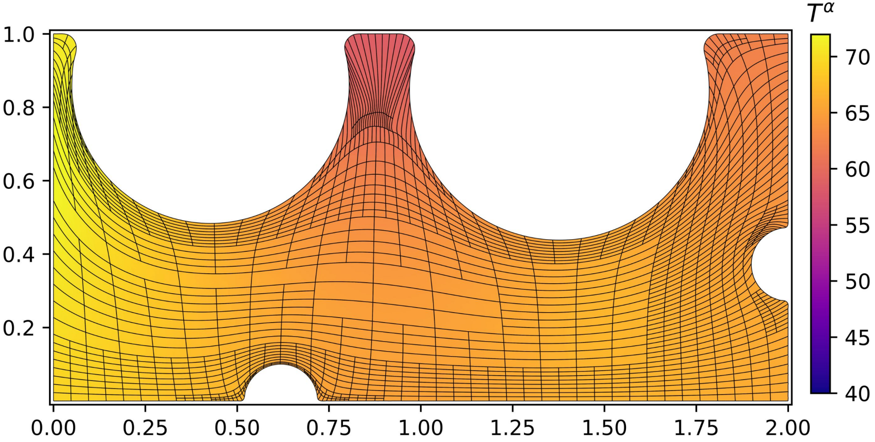
All details can be found in: J. Hinz *et al.* The role of PDE-based parameterization techniques in gradient-based IGA shape optimization applications. CMAME 378, 113685, 2021.

Design optimization of a cooling element, cont'd



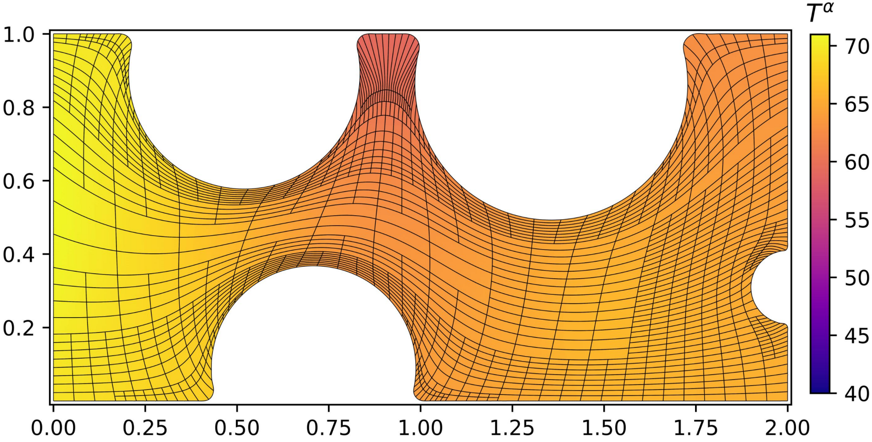
Temperature field $u^{\boldsymbol{\lambda}^0}_{\mathbf{h},\mathbf{p}}$ of the initial guess of the cooling element, $J^{\boldsymbol{\lambda}^0}_{\mathbf{h},\mathbf{p}} = 10.66$

Design optimization of a cooling element, cont'd



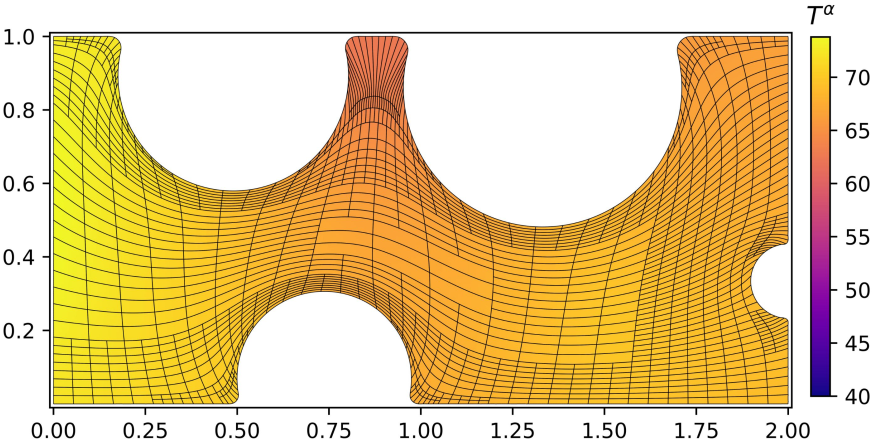
Temperature field of $u_{\mathbf{h},\mathbf{p}}^{\boldsymbol{\lambda}^4}$ the cooling element after $\ell = 4$ iterations

Design optimization of a cooling element, cont'd



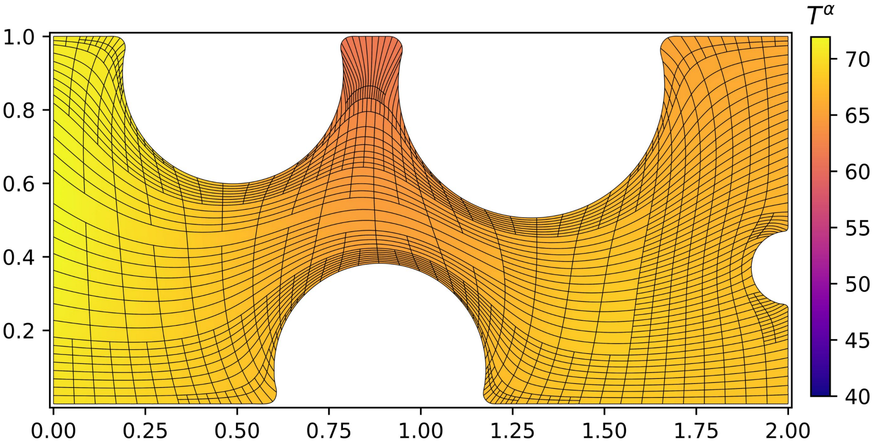
Temperature field $u_{\mathbf{h},\mathbf{p}}^{\boldsymbol{\lambda}^7}$ of the cooling element after $\ell = 7$ iterations

Design optimization of a cooling element, cont'd



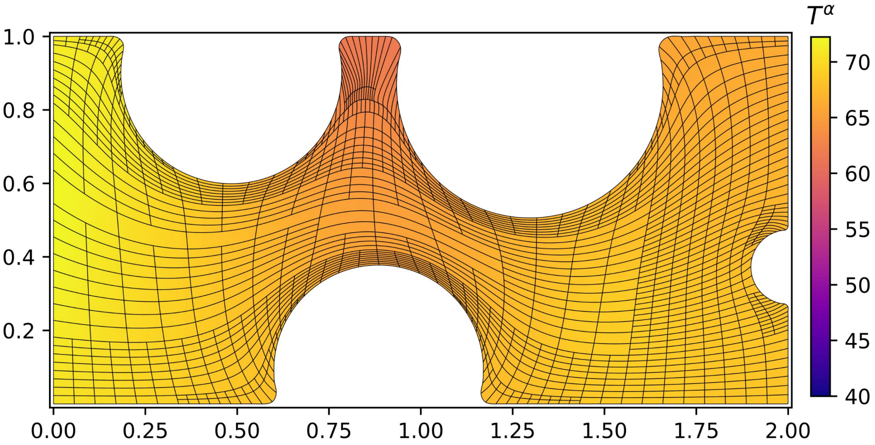
Temperature field $u_{\mathbf{h},\mathbf{p}}^{\boldsymbol{\lambda}^{10}}$ of the cooling element after $\ell = 10$ iterations

Design optimization of a cooling element, cont'd



Temperature field $u_{\mathbf{h},\mathbf{p}}^{\boldsymbol{\lambda}^{13}}$ of the cooling element after $\ell = 13$ iterations

Design optimization of a cooling element, cont'd



Temperature field $u_{\mathbf{h},\mathbf{p}}^{\boldsymbol{\lambda}^{15}}$ of the cooling element after $\ell = 15$ iterations, $J_{\mathbf{h},\mathbf{p}}^{\boldsymbol{\lambda}^{15}} = 6.29$

Further reading on IGA parametrization techniques ahs design optimization

J. Hinz: *PDE-based parameterization techniques for isogeometric analysis applications*, doctoral thesis, TU Delft, 2020

J. Hinz, A. Jaeschke, M. Möller and C. Vuik: *The role of PDE-based parametrization techniques in gradient-based IGA shape optimization*, CMAME 378:113685, 2021

A. Shamanskiy, M. Gfrerer, J. Hinz and B. Simeon: *Isogeometric parametrization inspired by large elastic deformation*, CMAME 363:112920, 2020

J. Hinz, J. Helmig, M. Möller and S. Elgeti: *Boundary-conforming finite element methods for twin-screw extruders using spline-based parametrization techniques*, CMAME 361:112740, 2020

J. Hinz, M. Möller and C. Vuik: *An IGA framework for PDE-based planar parametrization on convex multipatch domains*, In: Proceedings of IGAA 2018

J. Hinz, M. Möller and C. Vuik: *Spline-based parameterization techniques for twin-screw machine geometries*, In: IOP Conf. Series: Material Science and Engineering 425(1):012030, 2018

M. Möller and J. Hinz: *Isogeometric analysis framework for the numerical simulation of rotary screw machines*, In: IOP Conf. Series: Material Science and Engineering 425(1):012032, 2018

J. Hinz, M. Möller and C. Vuik: *Elliptic grid generation techniques in the framework of isogeometric analysis applications*, CAGD 65, 2018