

# Distributed time-parallel solution of transient problems with MGRIT and p-multigrid methods

Matthias Möller, Roel Tielen

Department of Applied Mathematics  
Delft University of Technology, NL

16th U.S. National Congress on Computational Mechanics  
July 25-29, 2021, A Virtual Event

MS 604: Isogeometric Methods

Acknowledgement: K. Vuik (TUD), D. Göddeke (University of Stuttgart)

# State of the art in IGA solvers

# 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

## 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]

# 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]

# 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]

## $p$ -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]



# 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)

$$\begin{aligned} -\nabla \cdot (\mathbb{D}\nabla u(\mathbf{x})) + \mathbf{v} \cdot \nabla u(\mathbf{x}) + ru(\mathbf{x}) &= f & \mathbf{x} \in \Omega \\ u(\mathbf{x}) &= g & \mathbf{x} \in \Gamma \end{aligned}$$

### Part II: Heat equation (Heat-Eq)

$$\begin{aligned} \partial_t u(\mathbf{x}, t) - \kappa \Delta u(\mathbf{x}, t) &= f & \mathbf{x} \in \Omega, t \in [0, T] \\ u(\mathbf{x}, t) &= g & \mathbf{x} \in \Gamma, t \in [0, T] \\ u(\mathbf{x}, 0) &= u^0(\mathbf{x}) & \mathbf{x} \in \Omega \end{aligned}$$

$d$ -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  $u \in \mathcal{H}_g^1(\Omega)$  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

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

## 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

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

$$k(w, u) := \kappa \int_{\Omega} \nabla u \cdot \nabla u \, dx \quad l(w) := \langle w, f \rangle$$

## 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

$$\mathbf{A}_{h,p} u_{h,p} = \mathbf{f}_{h,p}$$

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

$$[\mathbf{M}_{h,p} + \Delta t \mathbf{K}_{h,p}] u_{h,p}^{n+1} = \mathbf{M}_{h,p} u_{h,p}^n + \mathbf{f}_{h,p}$$

The unknown solution vector is given by

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

## Algebraic equations

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

$$\mathbf{A}_{h,p} u_{h,p} = \mathbf{f}_{h,p}$$

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

$$[\mathbf{M}_{h,p} + \Delta t \mathbf{K}_{h,p}] u_{h,p}^{n+1} = \mathbf{M}_{h,p} u_{h,p}^n + \mathbf{f}_{h,p}$$

The unknown solution vector is given by

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

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

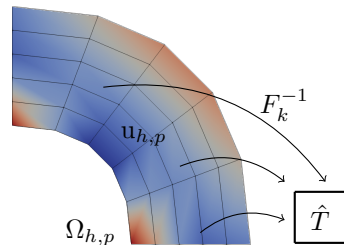
$$\mathbf{A}_{h,p} = \{a(\varphi_i, \varphi_j)\}_{i,j}, \quad \mathbf{K}_{h,p} = \{k(\varphi_i, \varphi_j)\}_{i,j}, \quad \mathbf{M}_{h,p} = \{\langle \varphi_i, \varphi_j \rangle\}_{i,j}, \quad \mathbf{f}_{h,p} = \{l(\varphi_i)\}_i$$

## 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’

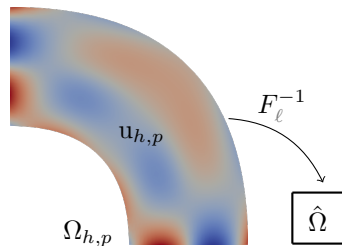
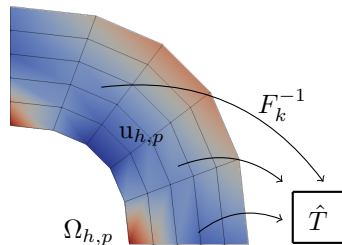
$$\mathcal{V}_{h,p} = \{v \in C^0(\bar{\Omega}) : v|_{T_k} \in \mathbb{Q}_p \circ F_k^{-1}, \forall T_k \in \mathcal{T}_h \\ v|_{\Gamma} = 0\}$$

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

**IGA:** patch-wise ‘pull-back’

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

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

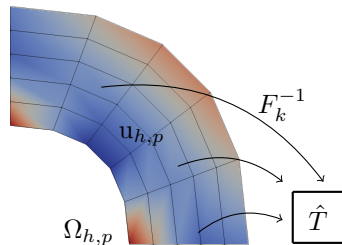


# Ansatz spaces

**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

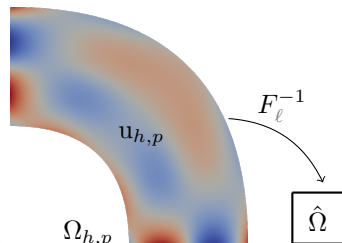
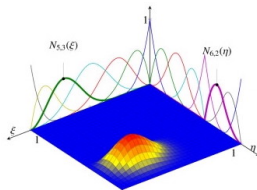


**IGA:** patch-wise ‘pull-back’

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

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

Think of IGA patches as macro elements



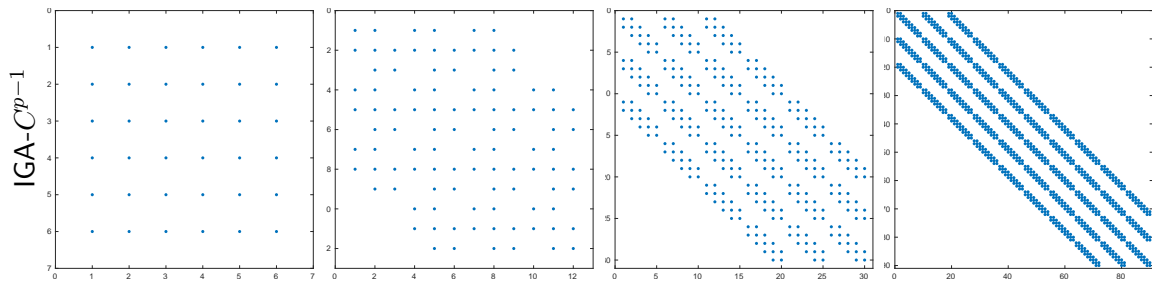
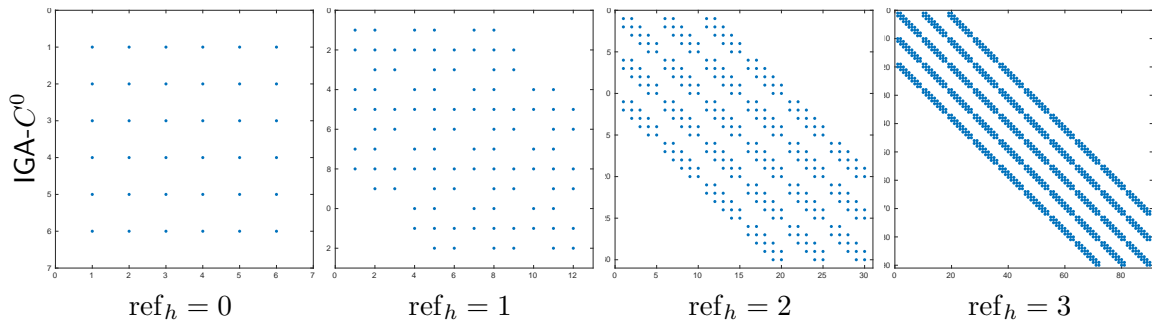
B-spline illustration taken from: H.Nguyen-XuanaLoc *et al.*, DOI: 10.1016/j.tafmec.2014.07.008

# Condition number

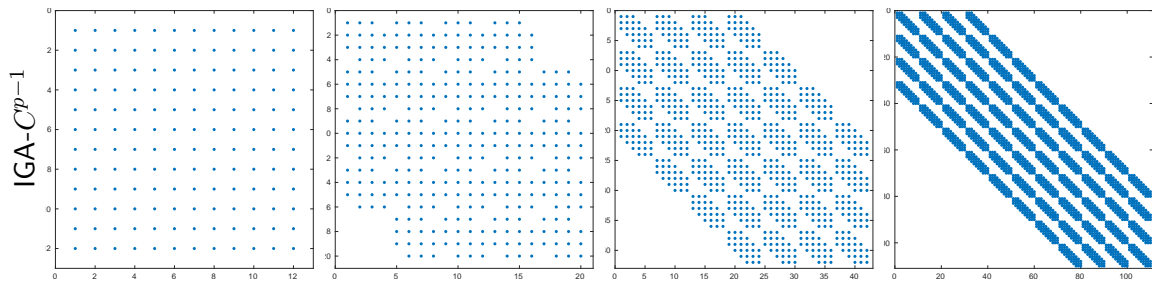
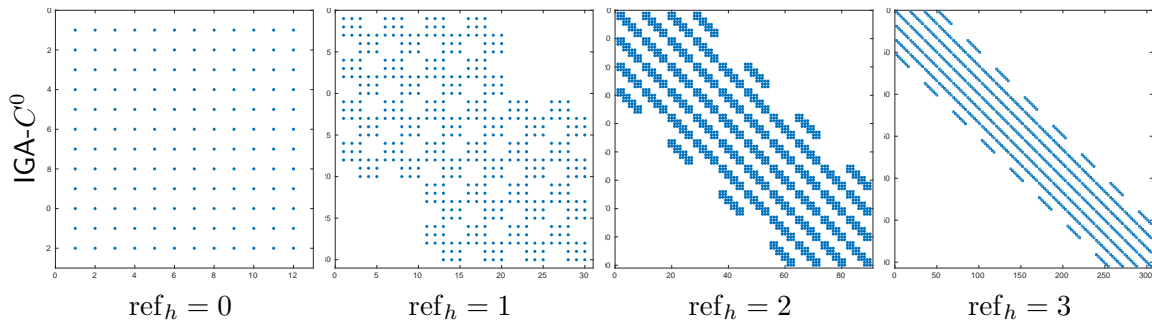
|                  | SEM-NI            | IGA- $C^0$             | IGA- $C^{p-1}$ |
|------------------|-------------------|------------------------|----------------|
| $\mathcal{K}(M)$ | $\sim p^d$        | $\sim p^{-d/2} 4^{pd}$ |                |
| $\mathcal{K}(K)$ | $\sim h^{-2} p^3$ |           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

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

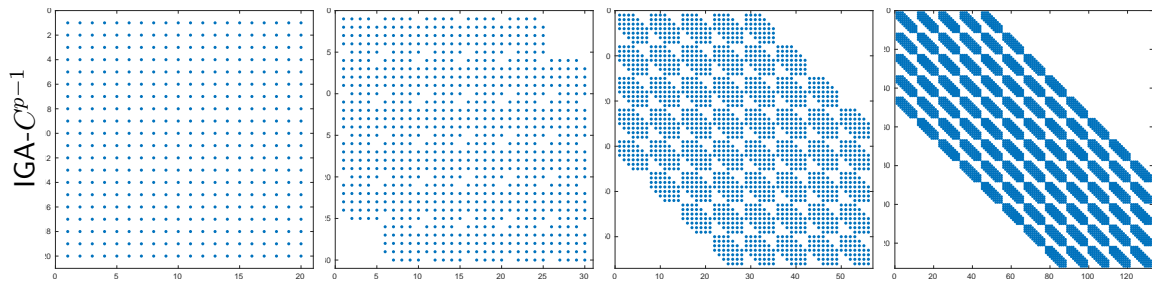
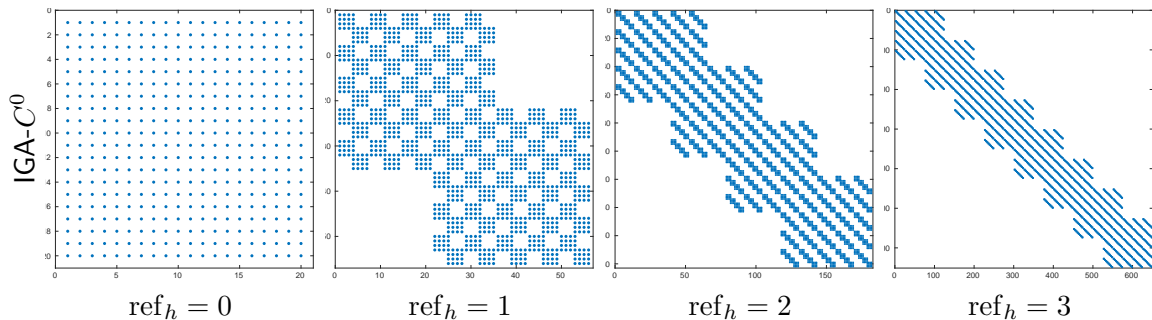
# Sparsity pattern: 2d single patch, $p = 1$



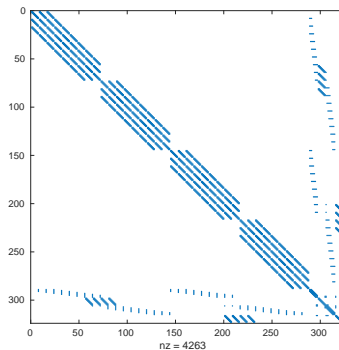
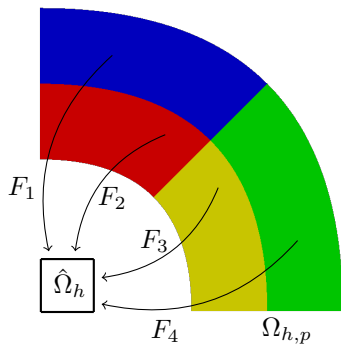
# Sparsity pattern: 2d single patch, $p = 2$



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



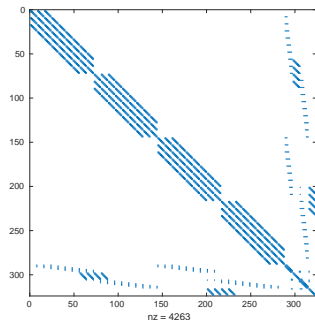
# Sparsity pattern: 2d multi-patch IGA- $C^{p-1}$ , $\text{ref}_h = 3$



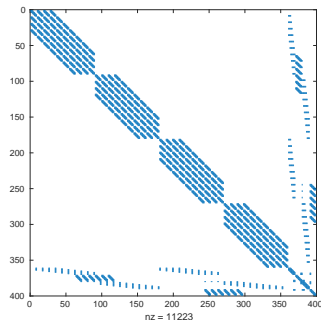
Four-patch geometry with  $C^0$  coupling of conforming degrees of freedom.

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

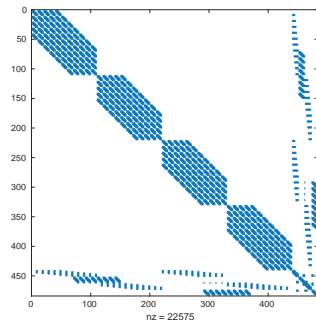
$p = 1$



$p = 2$



$p = 3$

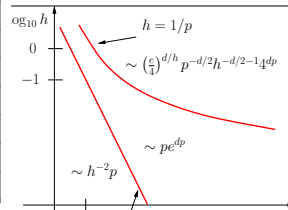
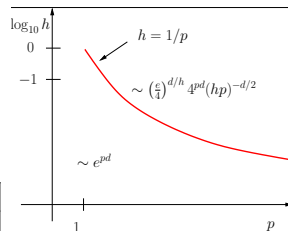
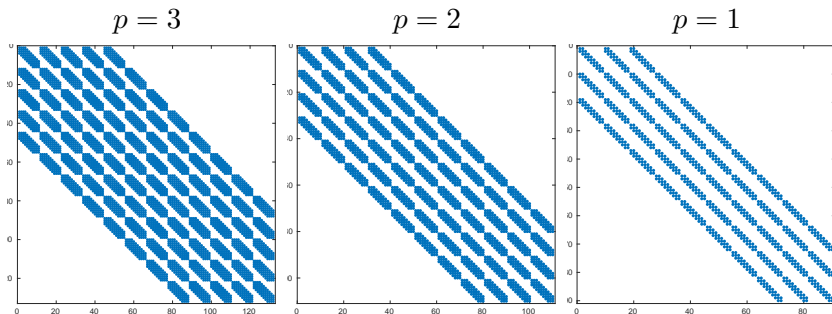


Four-patch geometry with  $C^0$  coupling of conforming degrees of freedom.



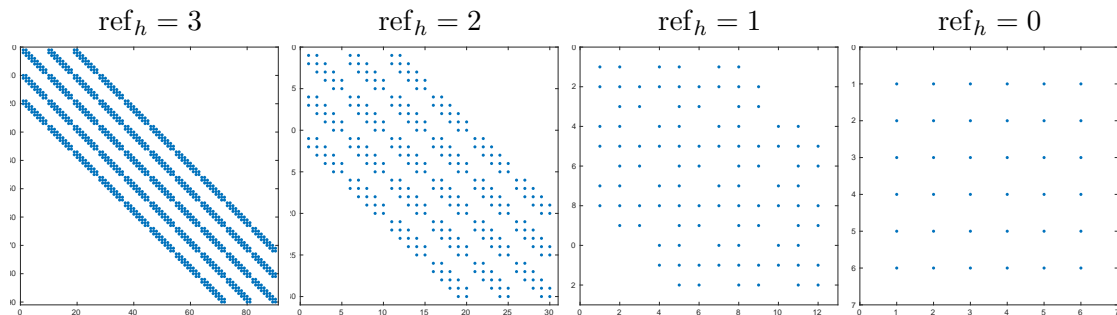
# Sketch of our solution strategy

- Coarsening in  $p$  reduces the stencil but not so much the number of unknowns
  - $p$ -multigrid with **direct projection**  $\mathcal{V}_{h,p} \searrow \mathcal{V}_{h,1}$
  - 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



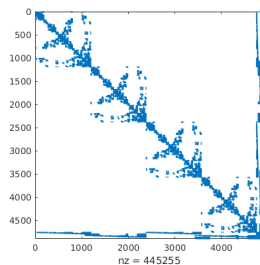
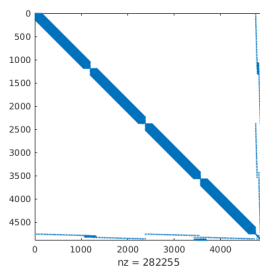
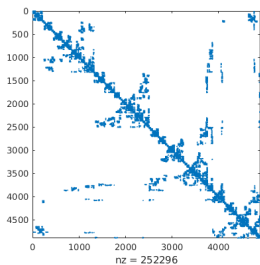
## Sketch of our solution strategy

- 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

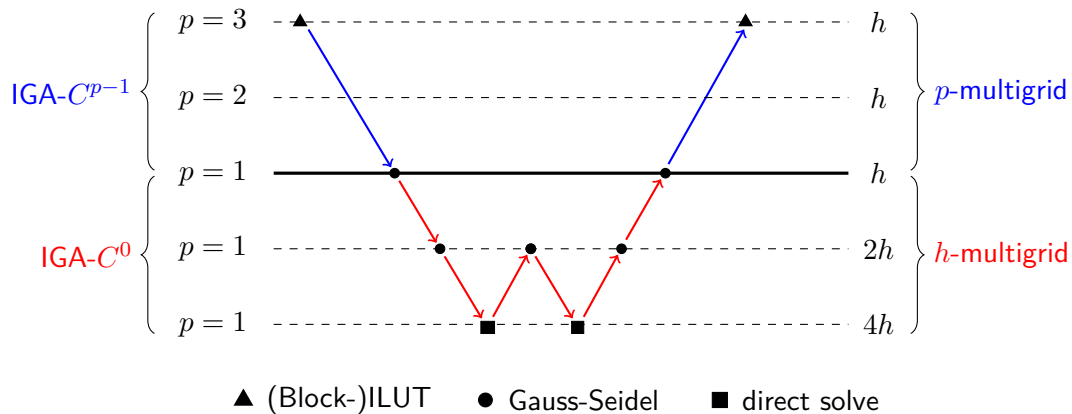
- Coarsening in  $p$  reduces the stencil but not so much the number of unknowns
  - $p$ -**multigrid with direct projection**  $\mathcal{V}_{h,p} \searrow \mathcal{V}_{h,1}$
  - note that spaces are not nested ( $\mathcal{V}_{h,p} \not\supset \mathcal{V}_{h,p-1} \not\supset \dots$ )
  - **ILUT smoother**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



## 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  $u_{h,p}^{(0,0)}$  apply  $\nu_1$  pre-smoothing steps:

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

## The complete multigrid algorithm – the outer $p$ -multigrid part

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

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

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  $\psi_j \in \mathcal{V}_{h,1}$

## The complete multigrid algorithm – the outer $p$ -multigrid part

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

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

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:

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



## The complete multigrid algorithm – the outer $p$ -multigrid part

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

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

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:

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

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:

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

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  $\text{tol}$ .

## The complete multigrid algorithm – the outer $p$ -multigrid part

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

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

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:

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

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  $\text{tol}$ .

## The complete multigrid algorithm – the outer $p$ -multigrid part

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

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

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:

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

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  $\text{tol}$ .

## The complete multigrid algorithm – the inner $h$ -multigrid part

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

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

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

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

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  $\mathcal{V}_{h,1}$  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

|                       | $h$ -multigrid   | $p$ -multigrid                            |
|-----------------------|--|---|
| restriction operator  | $I_{h,1}^{2h,1}$ linear interpolation                              | $I_{h,1}^{h,p} := M_{h,p}^{-1} M_{h,1,p}$ |
| prolongation operator | $I_{2h,1}^{h,1} := \frac{1}{2} \left( I_{h,1}^{2h,1} \right)^\top$ | $I_{h,p}^{h,1} := M_{h,1}^{-1} M_{h,p,1}$ |
|                       |  |   |
|                       |  |   |
|                       |  |   |

## Multigrid components

|                       | $h$ -multigrid   | $p$ -multigrid                            |
|-----------------------|--|---|
| restriction operator  | $I_{h,1}^{2h,1}$ linear interpolation  | $I_{h,1}^{h,p} := M_{h,p}^{-1} M_{h,1,p}$ |
| prolongation operator | $I_{2h,1}^{h,1} := \frac{1}{2} \left( I_{h,1}^{2h,1} \right)^\top$   | $I_{h,p}^{h,1} := M_{h,1}^{-1} M_{h,p,1}$ |
| smoothing operator    | 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 |   |
|                       |  |   |

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

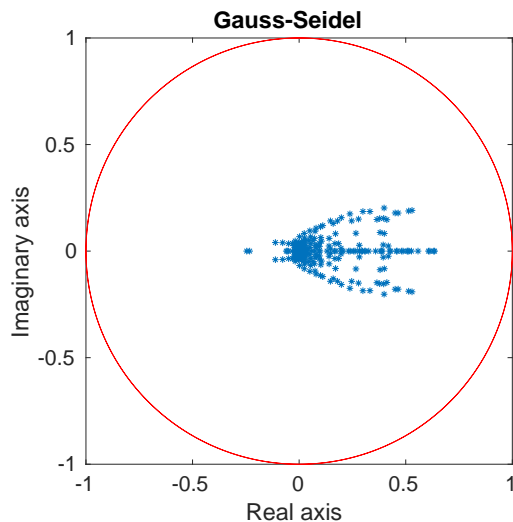
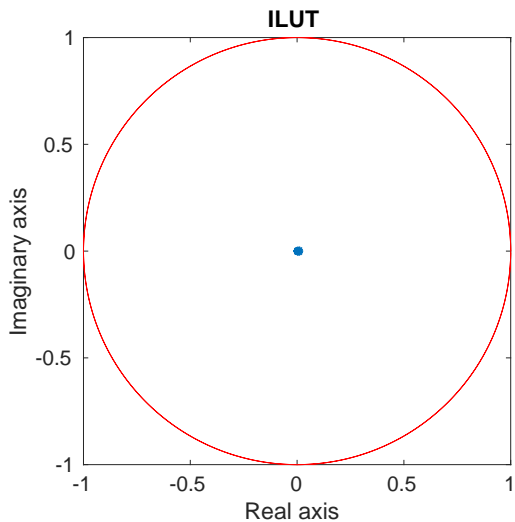
## Multigrid components

|                       | $h$ -multigrid   | $p$ -multigrid                            |
|-----------------------|--|---|
| restriction operator  | $I_{h,1}^{2h,1}$ linear interpolation  | $I_{h,1}^{h,p} := M_{h,p}^{-1} M_{h,1,p}$ |
| prolongation operator | $I_{2h,1}^{h,1} := \frac{1}{2} \left( I_{h,1}^{2h,1} \right)^\top$   | $I_{h,p}^{h,1} := M_{h,1}^{-1} M_{h,p,1}$ |
| smoothing operator    | 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 |   |
| $A_{h,p}$ operator    | rediscretization   |   |

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

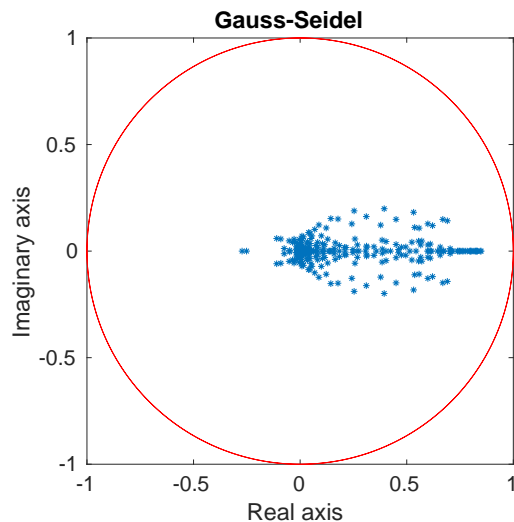
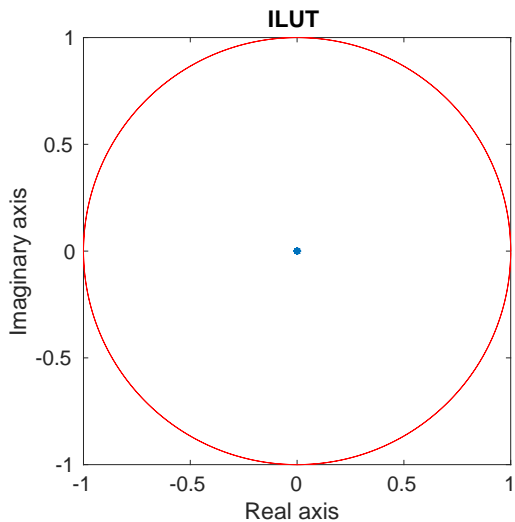


## Spectrum of the iteration matrix: Poisson on quarter annulus, $p = 2$



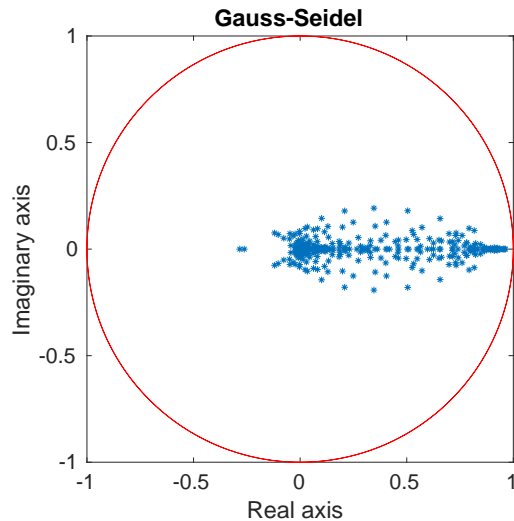
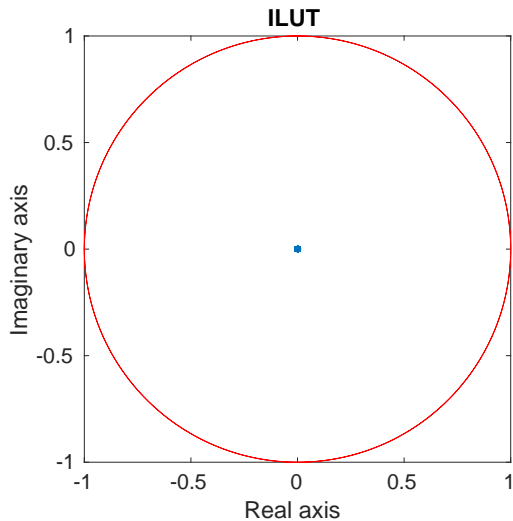
R. Tielen *et al.* 2020, DOI: [10.1016/j.cma.2020.113347](https://doi.org/10.1016/j.cma.2020.113347)

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



R. Tielen *et al.* 2020, DOI: [10.1016/j.cma.2020.113347](https://doi.org/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](https://doi.org/10.1016/j.cma.2020.113347)

## Numerical examples

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

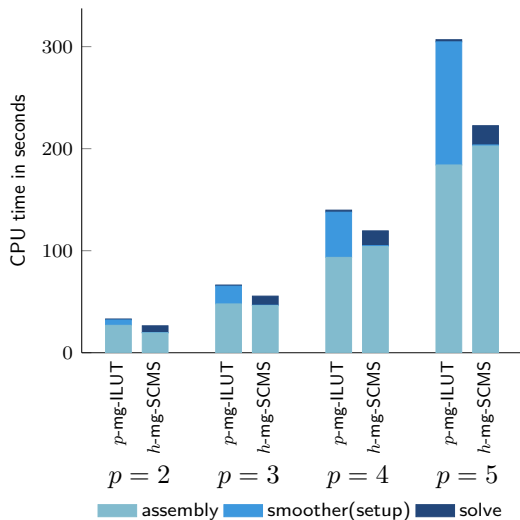
|              | $p = 2$ |    | $p = 3$ |    | $p = 4$ |     | $p = 5$ |     |
|--------------|---------|----|---------|----|---------|-----|---------|-----|
|              | ILUT    | GS | ILUT    | GS | ILUT    | GS  | ILUT    | GS  |
| $h = 2^{-6}$ | 4       | 30 | 3       | 62 | 3       | 176 | 3       | 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 |

## 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

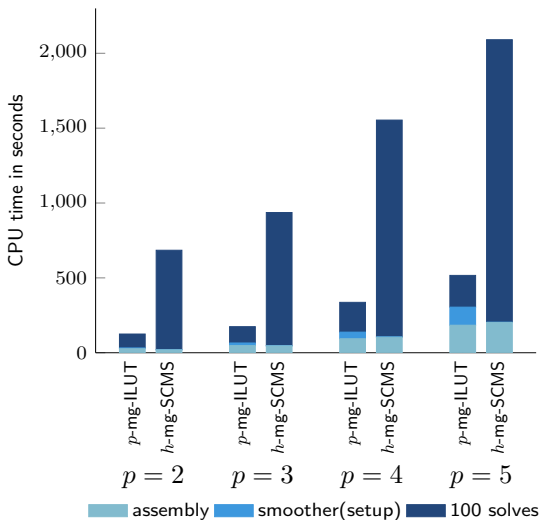
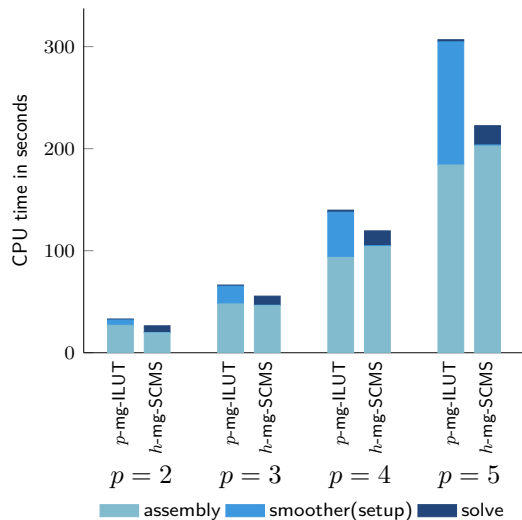
|              | $p = 2$ |    | $p = 3$ |    | $p = 4$ |    | $p = 5$ |    |
|--------------|---------|----|---------|----|---------|----|---------|----|
|              | ILUT    | GS | ILUT    | GS | ILUT    | GS | ILUT    | GS |
| $h = 2^{-6}$ | 5       | —  | 3       | —  | 3       | —  | 4      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

## Computational efficiency: $p$ - vs. $h$ -multigrid



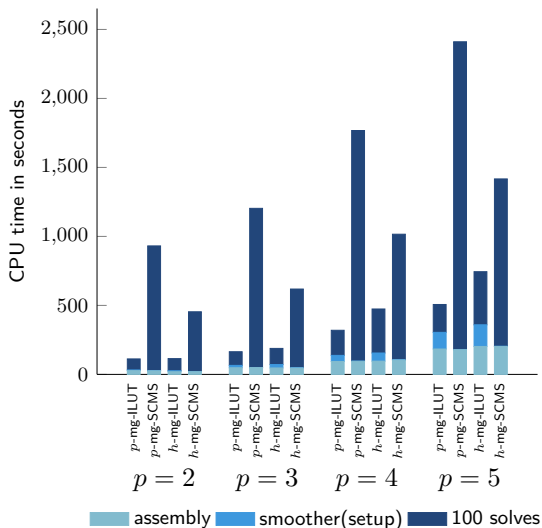
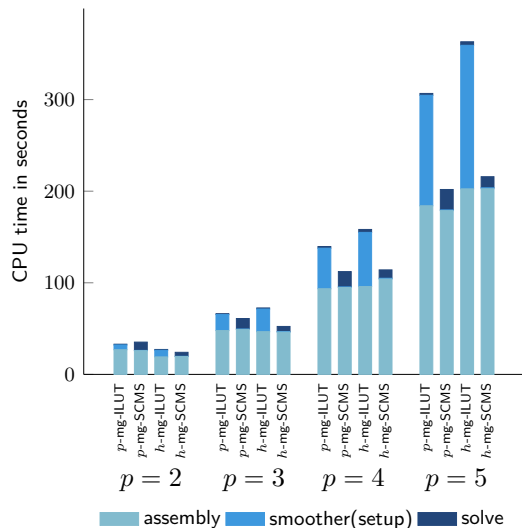
Comparison with  $h$ -multigrid method with subspace corrected mass smoother [Takacs, 2017]

## 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

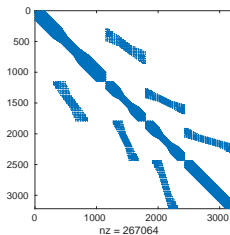
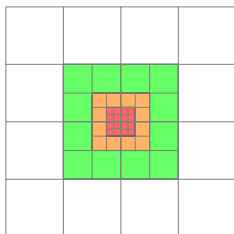




## Numerical examples: *THB splines*

#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](https://doi.org/10.1016/j.cma.2020.113347)

## Block ILUT

Exact LU decomposition of the block matrix  $A$

$$\begin{bmatrix} A_{11} & & & A_{\Gamma 1} \\ & \ddots & & \vdots \\ & & A_{N_p N_p} & A_{\Gamma N_p} \\ A_{1\Gamma} & \cdots & A_{N_p \Gamma} & A_{\Gamma \Gamma} \end{bmatrix} = \begin{bmatrix} L_1 & & & \\ & \ddots & & \\ & & L_{N_p} & \\ B_1 & \cdots & B_{N_p} & I \end{bmatrix} \begin{bmatrix} U_1 & & & C_1 \\ & \ddots & & \vdots \\ & & U_{N_p} & C_{N_p} \\ & & & S \end{bmatrix},$$

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  $A$

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

with

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

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

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

---

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

|              | $p = 2$<br># patches |      |      | $p = 3$<br># patches |      |       | $p = 4$<br># patches |      |       | $p = 5$<br># 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

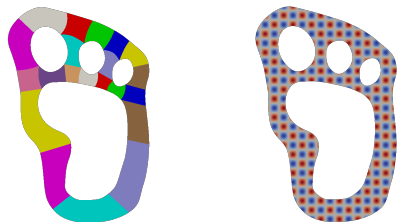
---

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

# 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

- 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)

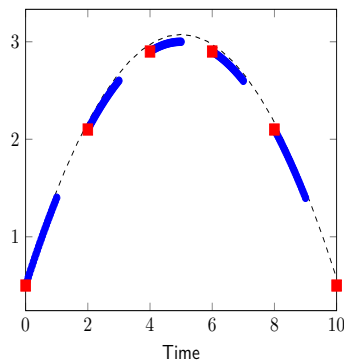
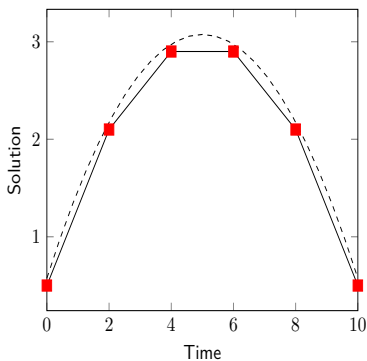
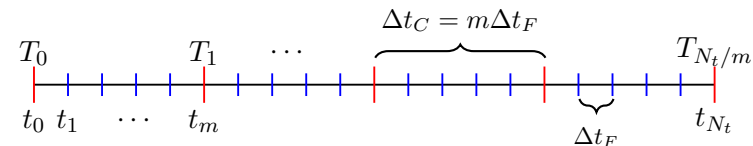
## ③ Part II: Multigrid reduction in time (MGRIT)

Introduction to MGRIT

MGRIT-IGA

## ④ Conclusions

## Part II: Multigrid reduction in time (MGRIT)



S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16<sup>th</sup> 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

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

---

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16<sup>th</sup> 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

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

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

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

with

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

---

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16<sup>th</sup> Copper Mountain Conference on Multigrid Methods 2013

## Sketch of the MGRIT algorithm, cont'd

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

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

---

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16<sup>th</sup> Copper Mountain Conference on Multigrid Methods 2013

## 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

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

---

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16<sup>th</sup> Copper Mountain Conference on Multigrid Methods 2013

## 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

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

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

---

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16<sup>th</sup> Copper Mountain Conference on Multigrid Methods 2013

## Sketch of the MGRIT algorithm, cont'd

Approximate the Schur complement

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

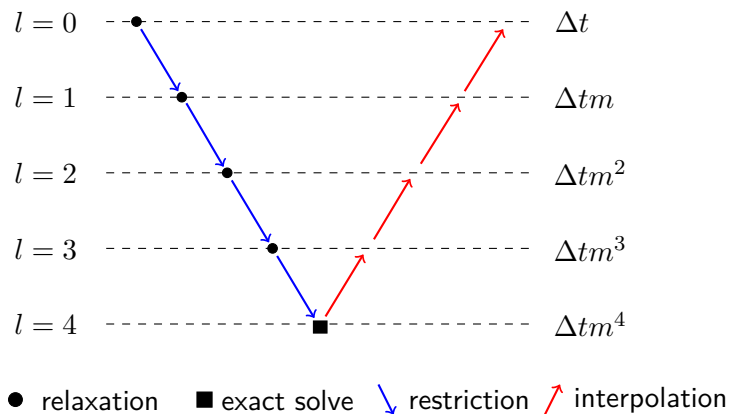
with *coarse integrator*

$$\Phi_{h,p} = [M_{h,p} + \Delta t_C K_{h,p}]^{-1}$$

---

S. Friedhoff, et al. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, 16<sup>th</sup> 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



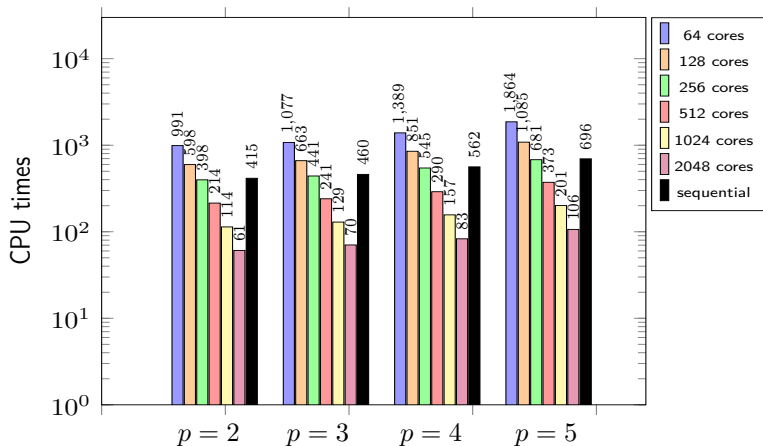
### Try it yourself

<https://github.com/gismo/gismo/tree/xbraid/extensions/gsXBraid>



## Numerical examples: *Strong scaling of MGRIT-IGA*

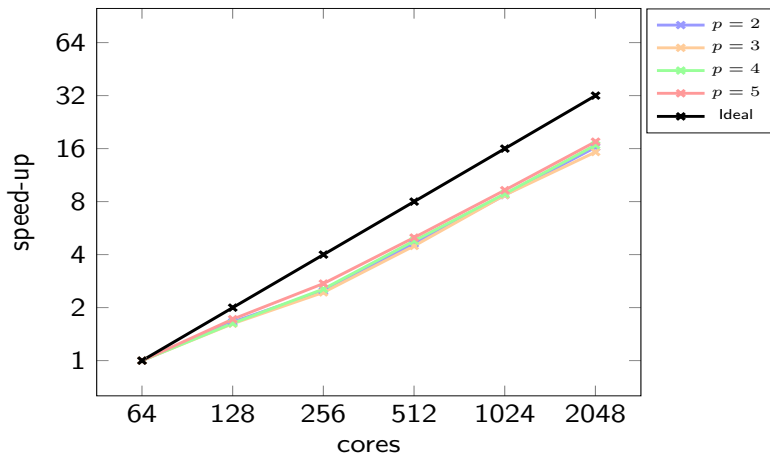
**#5:** Heat-Eq with  $h = 2^{-6}$  spatial resolution solved for  $N_t = 10.000$  time steps with backward Euler method on 128 Xeon Gold 6130 CPUs (2.10GHz, 96GB, 16 cores)



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

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

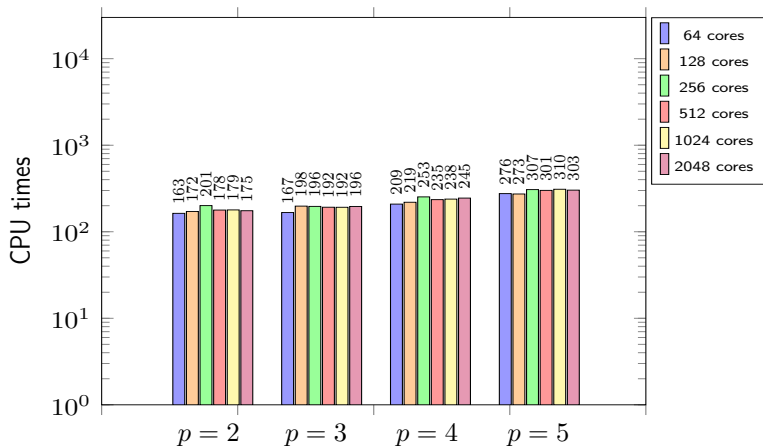
**#5:** Heat-Eq with  $h = 2^{-6}$  spatial resolution solved for  $N_t = 10.000$  time steps with backward Euler method on 128 Xeon Gold 6130 CPUs (2.10GHz, 96GB, 16 cores)



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  $N_t = \text{cores}/64 \cdot 1.000$  time steps with backward Euler method on 128 Xeon Gold 6130 CPUs (2.10GHz, 96GB, 16 cores)



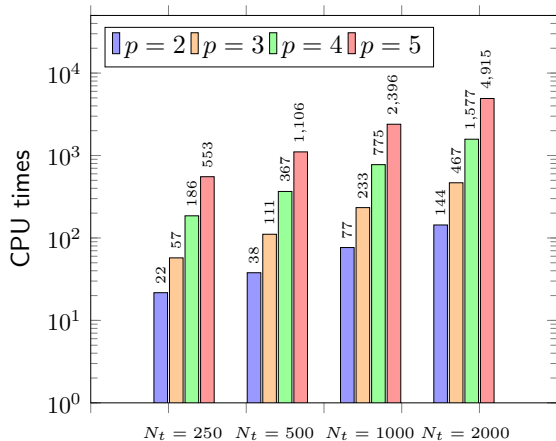
R. Tielen *et al.* 2021, arXiv:2107.05337

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

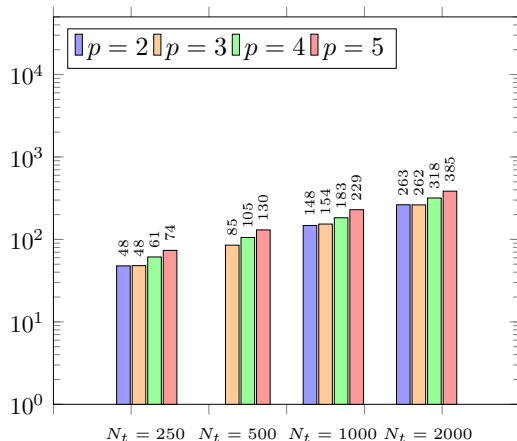
Do we really need  $p$ -multigrid or would a standard solver be good enough?

No!

CG solver on  $3 \times 1$  cores



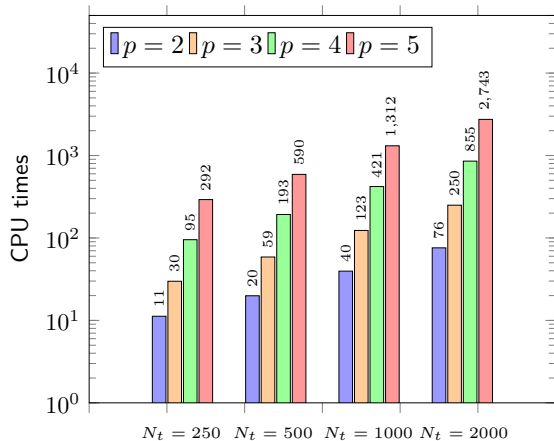
$p$ -mg-ILUT on  $3 \times 1$  cores



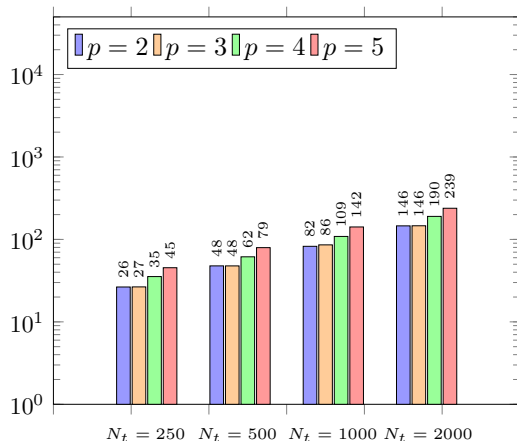
Do we really need  $p$ -multigrid or would a standard solver be good enough?

No!

CG solver on  $3 \times 2$  cores



$p$ -mg-ILUT on  $3 \times 2$  cores



### **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  $N_t$

## **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, D. Göttsche and C. Vuik: *p-multigrid methods and their comparison to h-multigrid methods within Isogeometric Analysis*, Computer Methods in Applied Mechanics and Engineering, Vol 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

Thank you for your attention!