P-multigrid methods for Isogeometric analysis

Tielen, R.P.W.M.

DOI
10.4233/uuid:8445e901-e31e-4602-8630-5b39a3de7ff6

Publication date
2021

Document Version
Final published version

Citation (APA)
Tielen, R. P. W. M. (2021). *P-multigrid methods for Isogeometric analysis*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:8445e901-e31e-4602-8630-5b39a3de7ff6

P-MULTIGRID METHODS FOR ISOGEOMETRIC ANALYSIS

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
donderdag 21 oktober 2021 om 10:00 uur

door

Roel Petrus Wilhelmus Maria TIELEN

Wiskundig ingenieur, Technische Universiteit Delft, Nederland,
geboren te Roosendaal, Nederland.

Dit proefschrift is goedgekeurd door de promotoren.

Samenstelling promotiecommissie bestaat uit:

Rector Magnificus,	voorzitter
Prof.dr.ir. C. Vuik,	Technische Universiteit Delft, promotor
Dr.rer.nat. M. Möller,	Technische Universiteit Delft, promotor
Onafhankelijke leden:	
Prof.dr.ir. C.W. Oosterlee	Technische Universiteit Delft
Prof. G. Sangalli	Universiteit van Pavia, Italië
Prof.dr. B. Wohlmuth	Technische Universiteit München, Duitsland
Dr. S. Takacs	Johannes Kepler Universiteit Linz, Oostenrijk
Prof.dr.ir. A.W. Heemink	Technische Universiteit Delft, reservelid
Overige lid:	
Prof.dr.rer.nat. D. Göddeke	Universiteit van Stuttgart, Duitsland

Keywords: Isogeometric Analysis, p-multigrid, Multigrid Reduced in Time

Printed by: ProefschriftMaken

ISBN 978-94-6366-453-0

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

CONTENTS

List of Symbols

In this section, we summarize the most common symbols adopted in this dissertation grouped by category.

B-spline basis functions

Ξ	Knot vector
ξ_i	Knot
p	Spline degree
h	Knot span size (mesh width)
d	Dimension of the domain of interest
$\phi_{i,p}$	Univariate B-spline basis function of degree p
$\Phi_{i,p}$	Multivariate B-spline basis function of degree p
$\mathbf{F}$	Geometry function (single patch)
$\mathbf{F}_{(k)}$	Geometry function (multipatch)
$\mathbf{DF}$	Jacobian of the geometry function
K	Number of patches
N	Number of univariate B-spline basis functions
N_{dof}	Number of multivariate B-spline basis functions
C	B-spline curve
$\mathbf{B}_i$	Control point
S	B-spline surface
$\mathbf{B}_{i,j}$	Control net
$R_{i,p}$	Rational B-spline basis function

Matrices, vectors and operators

$\mathbf{A}_{h,p}$	System matrix
$\tilde{\mathbf{A}}_{h,p}$	Approximation of $\mathbf{A}_{h,p}$
$\mathbf{D}_{h,p}$	Matrix containing the diagonal entries of $\mathbf{A}_{h,p}$
$\mathbf{u}_{h,p}$	Solution vector
$\mathbf{f}_{h,p}$	Right-hand side vector
$\mathbf{r}_{h,p}$	Residual vector
$\mathbf{e}_{h,p}$	Error vector
$I_{h,p}^{\tilde{h},p}$	Prolongation operator (h-multigrid)
$I_{\tilde{h},p}^{h,p}$	Restriction operator (h-multigrid)
I_p^1	Restriction operator (p-multigrid)
I_1^p	Prolongation operator (p-multigrid)
$S_{h,p}$	Smoothing operator

Matrices, vectors and operators (continued)

$\mathbf{M}_p$	Mass matrix
$\mathbf{M}_p^L$	Lumped mass matrix
$\mathbf{P}_1^p$	Transfer matrix
$\mathbf{L}_{h,p}$	Lower-triangular matrix
$\mathbf{U}_{h,p}$	Upper-triangular matrix

Variational formulation

$\mathbf{D}$	Diffusion tensor
$\mathbf{v}$	Velocity field
R	Reaction term
H^1	order 1 Sobolev space of L_2-integrable functions
H_0^1	Functions in H^1 that vanish on the boundary
$\mathcal{V}$	H_0^1
$\mathcal{V}_{h,p}$	Space of B-spline basis functions of degree p
Ω	Domain of interest
$\partial\Omega$	Boundary of the domain of interest
$\hat{\Omega}$	Parameter domain
$\Omega_{(k)}$	Single subdomain

Variables and parameters

$\mathbb{R}^d$	Euclidean space of dimension d
ξ	Variable in the parameter domain
$\mathbf{x}$	Variable in the physical domain
ϵ	Tolerance for stopping criterion
τ	threshold in ILUT factorization
ρ	fillfactor in ILUT factorization

Multigrid Reduced in Time

t	Time variable
Δt	Time step size
Δt_F	Fine time step size
Δt_C	Coarse time step size
$\mathbf{A}$	System matrix
$\mathbf{u}$	Solution vector
$\mathbf{g}$	Right-hand side vector
$\mathbf{S}$	Schur complement
m	Coarsening factor
L	Number of levels in MGRIT hierarchy

SUMMARY

Isogeometric Analysis is a methodology that bridges the gap between Computer Aided Design (CAD) and the Finite Element Method (FEM) by adopting the building blocks used in CAD, namely Non-Uniform Rational B-Splines and B-splines, as a basis for FEM. The use of these high-order spline functions does not only lead to an accurate representation of the geometry, but has shown to be advantageous in many different fields of research.

In order to obtain accurate numerical solutions, sufficiently fine meshes have to be considered which results in very large linear systems of equations. Furthermore, the condition numbers of the system matrices grow exponentially in the spline degree p, making the use of standard iterative solvers less efficient. Direct methods, on the other hand, might not be a viable alternative for large problem sizes, due to memory constraints and difficulties to parallelize. In recent years, the development of efficient iterative solvers for Isogeometric Analysis has therefore become an active field of research.

For standard FEM, multigrid methods are known to be among the most efficient solvers for elliptic partial differential equations. The direct application of these methods to linear systems arising in Isogeometric Analysis results, however, in multigrid methods with deteriorating performance for higher values of the spline degree p, since the multigrid smoother becomes less and less effective in damping the error. This has led to the development of multigrid methods with non-standard smoothers.

In this dissertation, we propose the use of p-multigrid methods as an alternative solution strategy. Within our p-multigrid method, the coarse grid correction is obtained at level $p = 1$, enabling the use of well-known solution methods for standard Lagrangian FEM (in particular h-multigrid methods). Furthermore, the support of the basis functions significantly reduces at level $p = 1$, thereby reducing the number of non-zero entries in the coarse grid operators. We analyze the performance of our p-multigrid method, adopting different smoothers, for single patch and multipatch geometries. In particular, we perform a spectral analysis to investigate the interplay between the coarse grid correction and smoothing procedure and obtain the asymptotic convergence rate of the p-multigrid method for a representative scenario. Numerical results (i.e., iteration numbers and CPU timings) are obtained for a variety of two- and three-dimensional benchmarks and compared to (state-of-the-art) h-multigrid methods to show the potential of p-multigrid methods in the context of Isogeometric Analysis.

For time-dependent partial differential equations, we apply Multigrid Reduced in Time (MGRIT), which is a parallel-in-time method, on discretizations arising in Isogeometric Analysis. Here, MGRIT is successfully combined with a p-multigrid method to obtain an overall efficient method.

SAMENVATTING

Isogeometrische Analyse is een numerieke methode die een brug slaat tussen computer-ondersteund ontwerpen (COO) en de eindige elementen methode (EEM) door de bouwstenen van COO, namelijk niet uniforme rationele B-splines en B-splines, als een basis voor de EEM te gebruiken. Het gebruik van deze hogere orde spline functies leidt niet alleen tot een accurate representatie van de geometrie, maar heeft aangetoond voordelig te zijn in veel verschillende onderzoeksvelden.
Om nauwkeurige numerieke oplossingen te verkrijgen, moeten voldoende fijne roosters gebruikt worden wat leidt tot erg grote lineaire systemen van vergelijkingen. Bovendien groeit het conditiegetal van de matrices exponentieel met de spline graad p, wat het gebruik van standaard iteratieve methoden minder efficient maakt. Directe methoden zijn, echter, geen alternatief voor grote systemen van vergelijkingen, vanwege geheugenbeperkingen. In recente jaren is de ontwikkeling van efficiente iteratieve oplossingsmethoden voor Isogeometrische Analyse daarom een actief onderzoeksveld geworden.
Voor standaard eindige elementen methodes staan multigrid methoden bekend als de meest efficiente oplossingsmethoden voor elliptische partiële differentiaalvergelijkingen. De directe toepassing van deze methoden op lineaire systemen die voorkomen in Isogeometrische Analyse leidt, echter, tot multigrid methoden die verslechteren voor hogere waarden van p, omdat de smoother minder effectief wordt. Dit heeft geleid tot de ontwikkeling van multigrid methoden met niet-standaard smoothers.
In deze dissertatie stellen we het gebruik van p-multigrid methoden voor als alternatieve oplossingsmethode. Met onze p-multigrid methode wordt de grove correctie verkregen op level $p = 1$, wat het gebruik van welbekende oplossingsmethoden voor standaard Lagrangian eindige elementen (zoals h-multigrid methoden) mogelijk maakt. Bovendien reduceert de support van de basis functies significant op level $p = 1$, waarmee het aantal niet-nul elementen in de grof grid operatoren gereduceerd wordt. We analyseren de prestatie van onze p-multigrid methode, gebruik makend van verschillende smoothers, voor enkele patch en meervoudige patch geometrien. In het bijzonder voeren we een spectraalanalyse uit om de wisselwerking tussen de grove correctie en de smoother te onderzoeken en de asymptotische convergentiesnelheid van de onderliggende p-multigrid methode te verkrijgen. Numerieke resultaten (dat wil zeggen, het aantal iteraties en CPU tijden) worden verkregen voor een verscheidenheid van twee- en driedimensionale problemen en vergeleken met (state-of-the-art) h-multigrid methoden om de potentie van p-multigrid methoden in de context van Isogeometrische Analyse te tonen.
Voor tijdsafhankelijke partiële differentiaalvergelijkingen, passen we Multigrid Gereduceerd in Tijd (MGGT), wat een parallel in tijd methode is, toe op discretizaties in Isogeometrische Analyse. Hierbij wordt MGGT succesvol gecombineerd met p-multigrid methoden om een algeheel efficiente methode te verkrijgen.

PREFACE

This dissertation presents the outcome of my scientific research that I performed from April 2017 till May 2021 at the Delft Institute of Applied Mathematics (DIAM) of Delft University of Technology. The aim of this research is the development of efficient solution strategies to solve linear systems of equations arising in Isogeometric Analysis. In particular, we consider p-multigrid methods as the type of solver to investigate, in which the coarse grid correction is obtained at a low-order level to update the solution at the high-order level.

Chapter 2 and 3 can be considered as a basic introduction to Isogeometric Analysis and multigrid methods. The reader familiar to these subjects could start reading from Chapter 4 onwards. Chapter 4, 5 and 6 are all based on papers published (or submitted) during the PhD research, indicated at the beginning of these chapters.

Chapter 7 has a special role in this thesis, as it focusses on the reproducibility of the results presented in the other chapters. In particular it gives a short introduction to the library adopted in this work, G+Smo (Geometry plus Simulation modules), and shows how results from this dissertation can be obtained by the reader using this library. I strongly believe that this chapter adds a lot of value to this research, as the reproducibility of numerical experiments is key within the academic setting.

Roel Petrus Wilhelmus Maria Tielen
Delft, October 2021

1

INTRODUCTION

The field of computational science and engineering (CSE) deals with the simulation of complex problems that arise in engineering. Many of these problems can be described by partial differential equations (PDEs), but, since for many applications analytic solutions do not exist, numerical methods are required to obtain an accurate approximation of the solution. The Finite Element Method (FEM) is widely used to numerically solve partial differential equations in CSE.

Within FEM, the domain of computation is divided into smaller parts, so-called elements, on which the solution of the underlying PDE is approximated. This division is obtained by dividing the domain of interest into triangles (or quadrilaterals in 2D and tetrahedra or hexahedra in 3D) to obtain a so-called mesh. As a consequence, curved geometries can not be approximated accurately with standard FEM. The PDE is then approximated by a linear combination of basis functions, which are defined locally on each element.

Solving a PDE numerically with FEM requires the solution of a linear system of equations. The solution of this linear system then leads to a numerical approximation of the quantity of interest (e.g. displacement, temperature or pressure). Within the field of Computer Aided Design (CAD), computers are used to create, modify and optimize a design. Typically, CAD is combined with FEM in order to develop designs that fulfil the required prescribed goals. Since the mesh used within the FEM is an approximation of the CAD geometry, remeshing is often necessary, in order to retain an accurate representation of the CAD geometry when performing the Finite Element Analysis (FEA).

Isogeometric Analysis (IgA) tries to unify these two fields, by adopting the same building blocks of CAD, Non-Uniform Rational B-splines (NURBS) and B-splines, for the Finite Element Analysis, thereby circumventing the approximation error of the geometry. Since its introduction in [1], IgA has become a viable alternative to the standard FEM and has been applied to a range of engineering fields, like structural mechanics [2–8], solid and fluid dynamics [9–15] and shape optimization [16–23].

Solving the resulting linear system of equations in IgA, remains, however, a challenging task for higher values of the spline degree of the B-spline basis functions as the condi-

tion numbers of the mass and stiffness matrices increase exponentially with the spline degree, rendering the use of (standard) iterative solvers inefficient. The use of (sparse) direct solvers, on the other hand, is not straightforward due to the increasing stencil of the basis functions and increasing bandwidth of matrices for higher values of the spline degree. Furthermore, direct solvers may not be practical for large problem sizes due to memory constraints, which is a common problem in high-order methods in general. Therefore, research focusses more and more on the development of efficient solution strategies for IgA [24–38]. An efficient class of solvers are multigrid methods [39–48], which combine the use of a basic iterative method, called smoother, with a coarse grid correction scheme. The computational costs of the resulting method have the potential to grow only linearly in the number of degrees of freedom, making multigrid methods one of the most efficient class of solvers. Within the context of IgA, most research focusses on h-multigrid (or geometric multigrid) methods, where a coarse grid correction is obtained by reducing the number of elements on coarser meshes. It was noticed that the use of standard iterative methods as a smoother (e.g. Gauss-Seidel or (damped) Jacobi) leads to multigrid methods with deteriorating performance for higher values of the spline degree which has led to the development of non-standard smoothers to overcome this drawback.
For high-order discretizations, the use of p-multigrid methods has been investigated [49–61] as well. Within p-multigrid methods, a low-order correction is obtained based on linear B-spline basis functions. Since linear B-spline basis functions coincide with (standard) P_1 Lagrange basis functions, well-established solution methods from standard FEM can be adopted to obtain the coarse-grid correction. Coarsening in p has successfully been combined with h-coarsening to further decrease the degrees of freedom at the coarsest level, leading to hp-methods [62–66].
In this thesis, we examine the use of p-multigrid methods as an alternative solution strategy within Isogeometric Analysis. More precisely, we:

- examine the use of p-multigrid methods, adopting different smoothers (e.g. Gauss-Seidel, ILUT), to efficiently solve linear systems of equations arising in Isogeometric Analysis (Chapter 4).
- investigate the use of a block ILUT smoother (instead of a global ILUT smoother) in case of multipatch geometries (Chapter 5).
- combine Multigrid Reduced in Time (MGRIT) methods with a p-multigrid method to solve time-dependent problems in Isogeometric Analysis (Chapter 6).

Within our p-multigrid method, the coarse grid correction will be obtained at level $p = 1$, which enables the use of well-known solution methods for standard Lagrangian FEM. Furthermore, the support of the basis functions significantly reduces at level $p = 1$, thereby reducing the number of non-zero entries in the coarse grid operators. We will analyze the performance of our p-multigrid method, and adopt different smoothers, for single patch and multipatch geometries. In particular, we perform a spectral analysis to investigate the interplay between the coarse grid correction and smoothing procedure and obtain the asymptotic convergence rate of the underlying p-multigrid method. Numerical results, including iteration numbers and CPU timings, are obtained for a variety of

two- and three-dimensional benchmarks and compared to those obtained with (state-of-the-art) h-multigrid methods to show the potential of p-multigrid methods in the context of Isogeometric Analysis. For time-dependent partial differential equations, we apply the Multigrid Reduced in Time (MGRIT) method on discretizations arising in Isogeometric Analysis. MGRIT is a parallel-in-time method that enables parallelism in the temporal direction. In particular, MGRIT is successfully combined with a p-multigrid method to obtain an overall efficient method on modern architectures.

Besides the introduction and conclusion, this thesis consists of six chapters:

2. Isogeometric Analysis
In this chapter, we give a short introduction to different concepts of Isogeometric Analysis. Starting from the variational formulation, the spatial discretization in Isogeometric Analysis is presented. In particular, we focus on the definition of B-spline basis functions, NURBS and possible refinement strategies (h-, k- and p-refinement).

3. Multigrid Methods
In this chapter, we provide a basic introduction to multigrid methods. Furthermore, we discuss both h- and p- multigrid methods and how they can be applied as a stand-alone solver or as a preconditioner within an outer Krylov method. Finally, we present a literature overview on the use of multigrid methods in the context of Isogeometric Analysis.

4. p-multigrid methods for Isogeometric Analysis
In this chapter, we present our p-multigrid method and analyze its performance in detail for single patch geometries. In particular, we consider the convection-diffusion-reaction (CDR) equation on the unit square, Poisson's equation on a quarter annulus and Poisson's equation on the unit cube. A spectral analysis is performed for the p-multigrid method adopting different smoothers (e.g. Gauss-Seidel and ILUT) and we compare our p-multigrid method to h-multigrid methods adopting a variety of smoothers.

5. p-multigrid methods for multipatch geometries
In this chapter, we focus on the use of p-multigrid methods for multipatch geometries. First, we investigate the performance of our p-multigrid method (as introduced in Chapter 4) in this setting. Then, the use of a block ILUT smoother is investigated to obtain a parallelizable p-multigrid method. Results are presented and compared to a global ILUT smoother for a variety of benchmarks.

1

6. Multigrid Reduced in Time (MGRIT)
In this chapter, we adopt the parallel-in-time method MGRIT for discretizations arising in Isogeometric Analysis. First, we introduce the MGRIT method and how it can be used to solve time-dependent PDEs discretized in space by Isogeometric Analysis. The spatial solves necessary within the MGRIT method are performed by adopting our p-multigrid method. Numerical results will be presented in this chapter, in terms of iteration numbers, CPU timings and strong and weak parallel scaling.

7. p-multigrid methods in G+Smo
In this chapter, we provide a short introduction to the open-source library Geometry plus Simulation modules (G+Smo), which is adopted throughout this thesis. Furthermore, we describe how the numerical results obtained in this thesis can be reproduced and extended by the reader using this library.

Each chapter is associated to a color, where the color indicates whether the chapter should be considered as an introduction to the subject (yellow), is based on publications (blue) or is software related (red).

2

ISOGEOMETRIC ANALYSIS

This chapter provides a basic introduction to Isogeometric Analysis. Starting from the variational formulation, showing many similarities with the standard Finite Element Method, B-spline basis functions and Non-Uniform Rational B-splines are introduced. Then, possible refinement strategies and assembly techniques commonly adopted within Isogeometric Analysis are described.

2.1. INTRODUCTION

Isogeometric Analysis aims to bridge the gap between Computer Aided Design (CAD) and the Finite Element Method (FEM) by adopting the same building blocks, B-splines or Non-Uniform Rational B-splines (NURBS), within the variational formulation and geometry description. As such, Isogeometric Analysis can be considered as the natural extension of the FEM to high-order B-spline basis functions. The use of these high-order B-spline basis functions results in a highly accurate representation of (curved) geometries. Furthermore, the (potential) smoothness of the basis functions leads to a higher accuracy per degree of freedom compared to FEM [67] and has shown to be beneficial in a variety of applications.

In this chapter, we provide a short introduction to Isogeometric Analysis. Section 2.2 describes the variational formulation and spatial discretization in Isogeometric Analysis. In Section 2.3, we introduce B-spline basis functions and Non-Uniform Rational B-splines (NURBS). Different refinement strategies and assembly techniques typically adopted within Isogeometric Analysis are discussed in Section 2.5 and 2.6, respectively. The chapter ends with some concluding remarks.

2.2. VARIATIONAL FORMULATION

To illustrate the variational formulation and spatial discretization in Isogeometric Analysis, we consider the convection-diffusion-reaction (CDR) equation:

$$-\nabla\cdot(\mathbf{D}\nabla u)+\mathbf{v}\cdot\nabla u+Ru=f,\quad \text{on } \Omega. \tag{2.1}$$

Here $\mathbf{D}$ denotes the diffusion tensor, $\mathbf{v}$ a divergence-free velocity field and R a source term. Furthermore, $\Omega\subset\mathbb{R}^d$ is a connected, Lipschitz domain in d dimensions, $f\in L^2(\Omega)$ and $u=0$ on the boundary $\partial\Omega$. Let $\mathcal{V}=H_0^1(\Omega)$ denote the space of functions within the Sobolev space $H^1(\Omega)$ that vanish on $\partial\Omega$:

$$\mathcal{V}=\left\{\, u\in H^1(\Omega)\mid u|_{\partial\Omega}=0\,\right\}. \tag{2.2}$$

The variational form of (2.1) is then obtained by multiplication with an arbitrary test function $v\in\mathcal{V}$ and application of integration by parts:

Find $u\in\mathcal{V}$ such that

$$a(u,v)=(f,v)\qquad \forall v\in\mathcal{V}, \tag{2.3}$$

where

$$a(u,v)=\int_\Omega(\mathbf{D}\nabla u)\cdot\nabla v+(\mathbf{v}\cdot\nabla u)v+Ruv\,\mathrm{d}\Omega \tag{2.4}$$

and

$$(f,v)=\int_\Omega fv\,\mathrm{d}\Omega. \tag{2.5}$$

The physical domain Ω is then parameterized by a geometry function

$$\mathbf{F}:\hat{\Omega}\to\Omega,\qquad \mathbf{F}(\boldsymbol{\xi})=\mathbf{x}. \tag{2.6}$$

The geometry function $\mathbf{F}$ describes a bijective mapping connecting the parameter domain $\hat{\Omega} = (0,1)^d$ with the physical domain Ω. In case Ω cannot be described by a single geometry function or such a function would lead to a singularity, the physical domain is divided into a collection of K non-overlapping subdomains $\Omega_{(k)}$ such that

$$\overline{\Omega} = \bigcup_{k=1}^{K} \overline{\Omega}_{(k)}. \tag{2.7}$$

A family of geometry functions $\mathbf{F}_{(k)}$ is then defined to parameterize each subdomain $\Omega_{(k)}$ separately:

$$\mathbf{F}_{(k)} : \hat{\Omega} \to \Omega_{(k)}, \qquad \mathbf{F}_{(k)}(\boldsymbol{\xi}) = \mathbf{x}. \tag{2.8}$$

In this case, we refer to Ω as a multipatch geometry consisting of K patches. Figure 2.1 shows, for illustrative purposes, a quarter annulus which is described by both a single mapping and a collection of mappings.

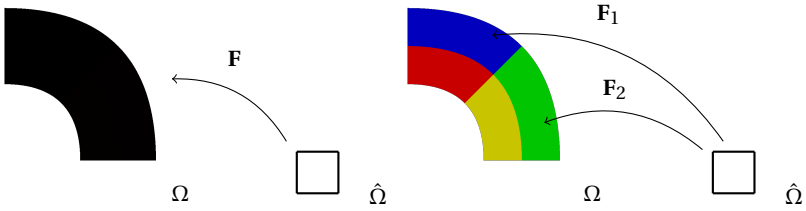


Figure 2.1: A quarter annulus described by a single mapping $\mathbf{F}$ and a collection of mappings $\mathbf{F}_{(k)}$ $(k = 1,\dots,4)$.

2.3. B-SPLINE BASIS FUNCTIONS

Throughout this thesis, B-spline basis functions are considered for the spatial discretization of Equation (2.3). Univariate B-spline basis functions are defined on the parameter domain $\hat{\Omega} = (0,1)$ and are uniquely determined by a knot vector Ξ consisting of knots $\xi_i \in \mathbb{R}$:

$$\Xi = \{\xi_1, \xi_2, \dots, \xi_{N+p}, \xi_{N+p+1}\}. \tag{2.9}$$

Here, N denotes the number of basis functions of degree p described by this knot vector. The difference between two consecutive knots ξ_i and ξ_{i+1} (assuming $\xi_i \neq \xi_{i+1}$) is referred to as the knot span size h_i. A knot vector is said to be *uniform* if all knot spans have an equal knot span size. Throughout this thesis, all knot vectors considered are uniform, where h denotes the knot span size of the knot vector (or 'mesh width'). A knot vector is called *open* if the first and last knots are repeated p times.
Based on the underlying knot vector $\Xi = \{\xi_1, \xi_2, \dots, \xi_{N+p}, \xi_{N+p+1}\}$, B-spline basis functions are then defined by the Cox de Boor formula [68] for $i = 1,\dots,N+p$, starting with the piecewise constants:

$$\phi_{i,0}(\xi) = \begin{cases} 1, & \text{if } \xi_i \le \xi < \xi_{i+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

High-order B-spline basis functions of degree p are then defined for $p > 0$ (with $\frac{0}{0} := 0$):

$$\phi_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i}\phi_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}}\phi_{i+1,p-1}(\xi), \quad i = 1, \ldots, N \tag{2.11}$$

2

B-spline basis functions possess the following appealing properties. They:

- have a compact support given by $[\xi_i, \xi_{i+p+1}]$,
- are non-negative on the entire domain, i.e.,

$$\phi_{i,p}(\xi) \geq 0 \quad \forall \xi \in [\xi_1, \xi_{N+p+1}], \tag{2.12}$$

- are C^{p-m_i} continuous at knot ξ_i, where m_i denotes the multiplicity of knot ξ_i,
- form a partition of unity, i.e.,

$$\sum_{i=1}^{N} \phi_{i,p}(\xi) = 1, \quad \forall \xi \in [\xi_i, \xi_{i+p+1}]. \tag{2.13}$$

The compact support of the B-spline basis functions results in sparse system matrices, although the bandwidth grows with the spline degree p. The non-negativity of the B-spline basis functions implies that all entries of the mass matrix are greater or equal to zero. As a consequence, row-sum lumping can easily be applied without 'losing' part of the total mass. In fact, the partition of unity allows us to apply row-sum lumping already within the variational formulation. In this thesis, we consider B-spline basis functions based on an open and uniform knot vector. Figure 2.2 denotes linear and quadratic B-spline basis functions based on such a knot vector.

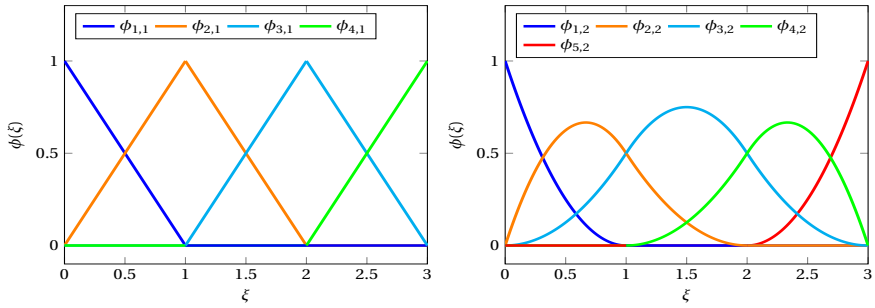


Figure 2.2: Univariate linear (left) and quadratic (right) B-spline basis functions based on the knot vectors $\Xi_1 = \{0,0,1,2,3,3\}$ (left) and $\Xi_2 = \{0,0,0,1,2,3,3,3\}$ (right), respectively.

2.4. SPATIAL DISCRETIZATION

For the spatial discretization of Equation (2.3) the tensor product of univariate B-spline basis functions is adopted. In this section, we consider the two-dimensional case, where

univariate B-spline basis functions $\phi_{i_x,p}(\xi)$ and $\phi_{i_y,q}(\eta)$ of degree p and q, respectively, with maximum continuity are adopted:

$$\Phi_{\vec{i},\vec{p}}(\boldsymbol{\xi}) := \phi_{i_x,p}(\xi)\phi_{i_y,q}(\eta), \qquad \vec{i} = (i_x, i_y),\ \vec{p} = (p,q). \tag{2.14}$$

Here, $\boldsymbol{\xi} = (\xi,\eta)$ and $\vec{i}$ and $\vec{p}$ are multi-indices, with $i_x = 1,\dots,n_x$ and $i_y = 1,\dots,n_y$ denoting the univariate basis functions in the x-dimension and y-dimension, respectively. Furthermore, $i = i_x n_x + (i_y - 1)n_y$ assigns a unique index to each pair of univariate basis functions, where $i = 1,\dots,N_{\text{dof}}$. Here, N_{dof} denotes the number of degrees of freedom, or equivalently, the number of tensor-product basis functions and depends on both h and p. Figure 2.3 shows a quadratic B-spline basis in two dimensions based on univariate quadratic B-spline basis functions. In this thesis, all univariate B-spline basis functions are assumed to be of the same degree (i.e., $p = q$).

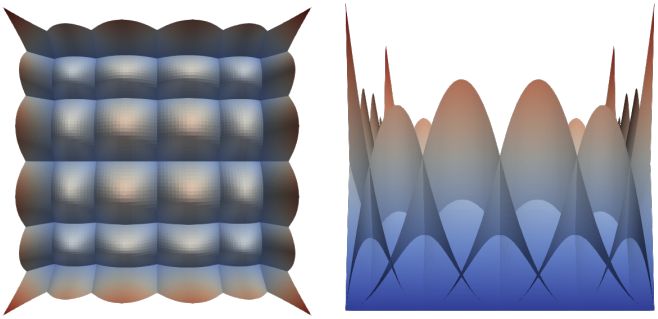


Figure 2.3: A quadratic B-spline basis in two dimensions from the top (left) and side (right) based on univariate quadratic B-spline basis functions.

The spline space $\mathcal{V}_{h,p}$ can then be written, using the inverse of the geometry mapping $\mathbf{F}^{-1}$ as pull-back operator, as follows:

$$\mathcal{V}_{h,p} := \text{span}\left\{\Phi_{\vec{i},\vec{p}} \circ \mathbf{F}^{-1} \,|\, \Phi_{\vec{i},\vec{p}} = 0 \text{ on } \partial\hat{\Omega}\right\}_{i=1,\dots,N_{\text{dof}}}. \tag{2.15}$$

The Galerkin formulation of (2.3) becomes: Find $u_{h,p} \in \mathcal{V}_{h,p}$ such that

$$a(u_{h,p}, v_{h,p}) = (f, v_{h,p}) \qquad \forall\, v_{h,p} \in \mathcal{V}_{h,p}. \tag{2.16}$$

The discretized problem can be written as a linear system

$$\mathbf{A}_{h,p}\mathbf{u}_{h,p} = \mathbf{f}_{h,p}, \tag{2.17}$$

where $\mathbf{A}_{h,p}$ denotes the system matrix resulting from this discretization with B-spline basis functions of degree p and mesh width h. For a more detailed description of the variational formulation in Isogeometric Analysis, we refer to [1].

B-SPLINE CURVES AND SURFACES

A B-spline curve can be constructed by taking a linear combination of B-spline basis functions:

$$C(\xi) = \sum_{i=1}^{N} \mathbf{B}_i \phi_{i,p}(\xi), \tag{2.18}$$

where $\mathbf{B}_i$ denote the N control points of the underlying B-spline curve. The *control polygon* of the B-spline curve can be obtained by piecewise linear interpolation of the control points. Figure 2.4 denotes a B-spline curve with its control polygon in $\mathbb{R}^2$. Note that, in general, the B-spline curve is not interpolatory at the control points unless the multiplicity of a certain knot is equal to the polynomial degree of the B-spline functions. This is only the case for the first and last control point, as this example is based on B-spline functions defined by an open knot vector.

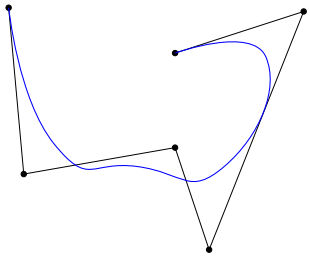


Figure 2.4: A B-spline curve and its control points in $\mathbb{R}^2$, taken from [1].

When the knots and control points are not repeated, the resulting B-spline curve has continuous derivatives up to degree $p-1$. Increasing the multiplicity of the knots or control points by k, results in a decrease of the number of continuous derivatives by k. Next, we introduce the concept of B-spline surfaces. A B-spline surface $S(\xi,\eta)$ is defined as follows:

$$S(\xi,\eta) = \sum_{i=1}^{N} \sum_{j=1}^{M} \phi_{i,p}(\xi)\phi_{j,q}(\eta)\mathbf{B}_{i,j}, \tag{2.19}$$

where $\mathbf{B}_{i,j}$ is the *control net* and $\phi_{i,p}$ and $\phi_{j,q}$ the B-spline basis functions of degree p and q, respectively. Here, N and M denote the number of B-spline basis functions of degree p and q, respectively. Figure 2.5 shows a biquadratic B-spline surface with its control net. B-spline solids can be defined in an analogous way, see [1] for more details. The construction of Non-Uniform Rational B-splines (NURBS) is based on these curves, surfaces and solids. For example, a NURBS curve is defined as follows:

$$\mathbf{C}(\xi) = \sum_{i=1}^{N} \mathbf{B}_i R_{i,p}(\xi), \tag{2.20}$$

where the rational basis functions $R_{i,p}$ are defined as follows:

$$R_{i,p}(\xi) = \frac{w_i \phi_{i,p}(\xi)}{\sum_{j=1}^{N} w_j \phi_{j,p}(\xi)}. \tag{2.21}$$

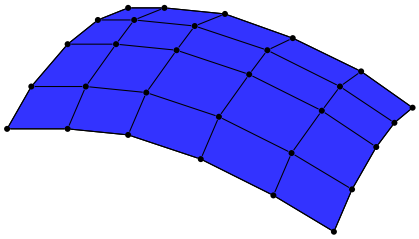


Figure 2.5: A B-spline surface and its control net in $\mathbb{R}^3$, taken from [1].

Here, w_i denotes the i^{th} weight. Again, Non-Uniform Rational B-spline surfaces and solids are defined in an analogous way. As B-spline basis functions, NURBS basis functions form a partition of unity. Furthermore, their continuity and support is the same as for B-splines. Figure 2.6 shows a control net for a toroidal surface and the resulting NURBS surface.

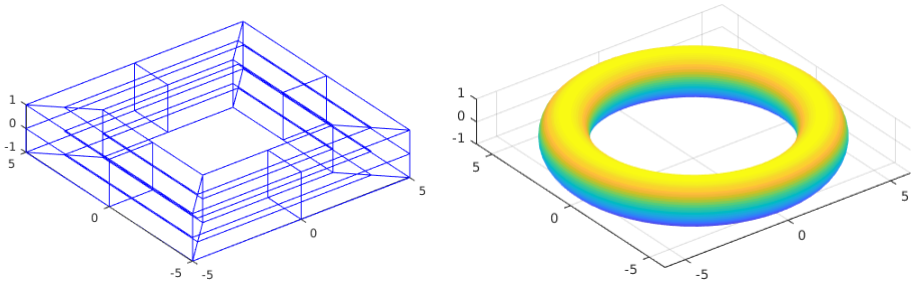


Figure 2.6: Control net (left) for a toroidal surface and the resulting NURBS surface (right), taken from [1].

2.5. REFINEMENT STRATEGIES

Within Isogeometric Analysis, three types of refinement can be distinguished, which are all based on the insertion of knots within the knot vector. The first one, h-refinement, increases the number of basis functions while keeping the degree of the basis functions the same. In particular, we consider uniform h-refinement, in which a knot is inserted halfway each knot span of non-zero size. Starting with the knot vector Ξ from the previous section, uniform h-refinement results in the following knot vector:

$$\Xi = \{0,0,0,1,2,3,3,3\} \Rightarrow \Xi = \{0,0,0,\tfrac{1}{2},1,\tfrac{3}{2},2,\tfrac{5}{2},3,3,3\}. \tag{2.22}$$

Figure 2.7 shows both the original basis functions and those after applying uniform h-refinement. Starting from the knot vector $\Xi = \{0,1\}$, applying h-refinement l times, results in a knot span size of $h = 2^{-l}$ and $\frac{1}{h} + p$ basis functions.
The second type of refinement is referred to as p-refinement. With p-refinement, both the degree of the basis functions and multiplicity of each knot is increased:

$$\Xi = \{0,0,1,2,3,3\} \Rightarrow \Xi = \{0,0,0,1,1,2,2,3,3,3\}. \tag{2.23}$$

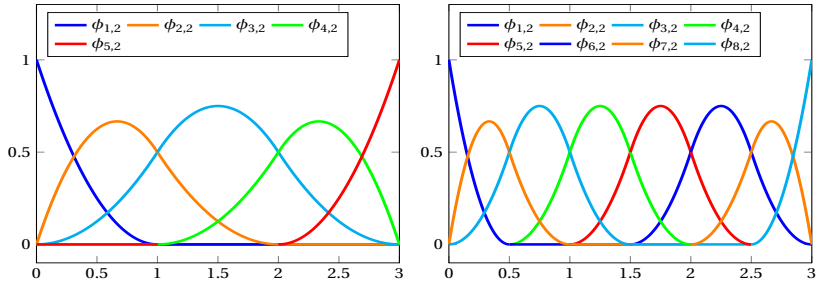


Figure 2.7: A quadratic B-spline basis before (left) and after (right) h-refinement. Here, the initial knot vector is given by $\Xi = \{0,0,0,1,2,3,3,3\}$.

As a consequence, the number of basis functions is increased, but they remain as smooth as the original basis functions. Figure 2.8 shows the original basis functions and those after applying p-refinement.

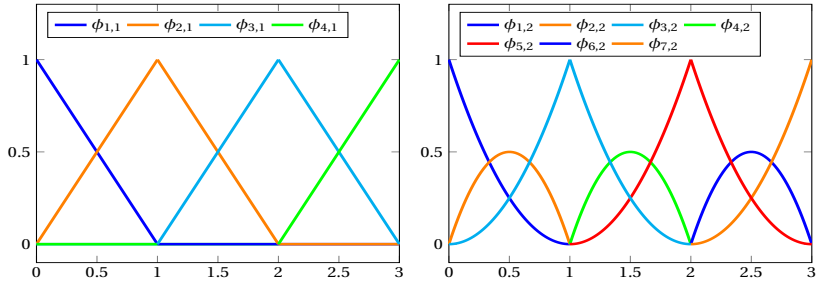


Figure 2.8: p-refinement applied on a linear B-spline basis defined by the knot vector $\Xi = \{0,0,1,2,3,3\}$.

Finally, there is the possibility to apply k-refinement to the knot vector. With k-refinement, knots are inserted at the end points:

$$\Xi = \{0,0,1,2,3,3\} \;\Rightarrow\; \Xi = \{0,0,0,1,2,3,3,3\}. \tag{2.24}$$

The spline degree is then elevated and the continuity is raised accordingly too. Hence, the resulting basis functions are C^{p-1} continuous. Furthermore, the number of basis functions after applying k-refinement is significantly lower compared to p-refinement. Figure 2.9 shows the original basis functions and those after applying k-refinement. It should be noted that k-refinement does not yield nested spaces. Note that there is no equivalent in standard Finite Element Methods for k-refinement.
In this thesis, both uniform h-refinement and k-refinement will be used extensively. Hence, B-spline basis functions with maximum continuity based on an open and uniform knot vector are considered within the p-multigrid methods.

Remark: The reader should not be confused by the fact that even though k-refinement is applied throughout this thesis, we refer to p-multigrid methods. The terminology 'p-multigrid methods' has been adopted from other fields (Finite Element Methods, Discontinuous Galerkin methods), where they have been applied successfully.

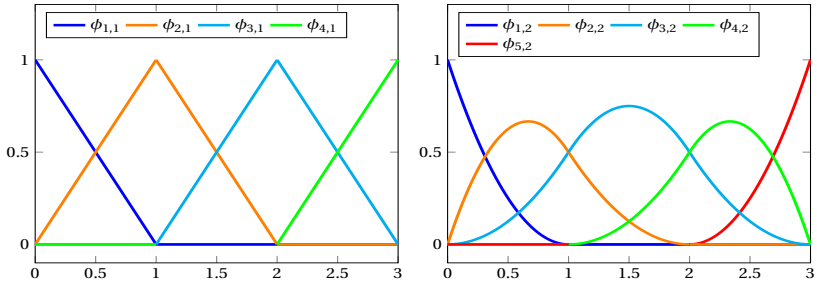


Figure 2.9: k-refinement applied on a linear B-spline basis defined by the knot vector $\Xi = \{0,0,1,2,3,3\}$.

2.6. MATRIX ASSEMBLY

The linear system given by Equation (2.17) is formed by evaluating the integrals arising in the variational formulation. Typically, an element-based assembly is considered, where one loops over all elements and determines all non-zero contributions to the linear system of equations by means of numerical quadrature. Therefore, it is necessary to determine which integrals are non-zero, which depends on the overlap between the basis functions within these integrals. Figure 2.10 shows the support of two quadratic B-spline basis functions in the parameter domain $\hat{\Omega} = (0,1)^2$ and their common support.

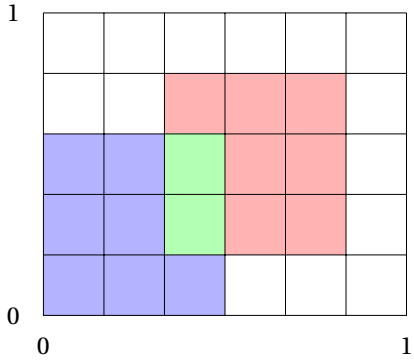


Figure 2.10: Support of two quadratic B-spline basis functions in the parameter domain $\hat{\Omega} = (0,1)^2$ with their common support denoted in green.

It should be noted that integrals are typically evaluated on the parameter domain rather than the physical domain, using the well known integration rule:

$$\int_{\Omega} g(\mathbf{x})\, \mathrm{d}\Omega = \int_{\hat{\Omega}} g\left(\mathbf{F}(\boldsymbol{\xi})\right) |\mathrm{det}\mathbf{DF}(\boldsymbol{\xi})|\, \mathrm{d}\hat{\Omega}, \tag{2.25}$$

where $\mathbf{DF}(\boldsymbol{\xi})$ is the Jacobian matrix of the geometry mapping. In standard FEM a similar transformation occurs to evaluate the integrals in a reference domain, although this mapping is then defined per element rather than globally.

Assuming an element-based assembly loop with standard Gaussian quadrature, assembling the stiffness matrix based on B-spline basis functions of degree p costs $\mathcal{O}(N_{\text{dof}} p^{3d})$ floating point operations (flops). Here, the assembly costs grow significantly for higher values of the spline degree p, due to the increasing support of the B-spline basis functions which results in an increased bandwidth of the stiffness matrix. As the assembly costs are responsible for a significant part of the total computational costs, recent research has focused on more efficient assembly strategies. Examples of such assembly techniques are weighted quadrature [69], sum factorizations [70], low tensor rank approximations [71] and the surrogate matrix methodology [72, 73]. In this dissertation, the element-based assembly loop with standard Gauss-quadrature is adopted. Therefore, assembly costs will be listed separately when presenting the computational times later in this dissertation, as they might dominate the overall computational costs.
The number of non-zero entries per row only grows with the spline degree of the B-spline basis functions and, therefore, the resulting system matrix is sparse. Figure 2.11 shows the stiffness matrix based on quadratic B-spline basis functions for a single patch (left) and multipatch (right) geometry. In case a multipatch geometry is considered, the system matrix has a block structure, where each block is associated to a single patch. The off-diagonal blocks denote the coupling between degrees of freedom at the interface, resulting in an arrowhead matrix. In Chapter 5, this structure will be used to develop a block ILUT smoother for multipatch geometries.

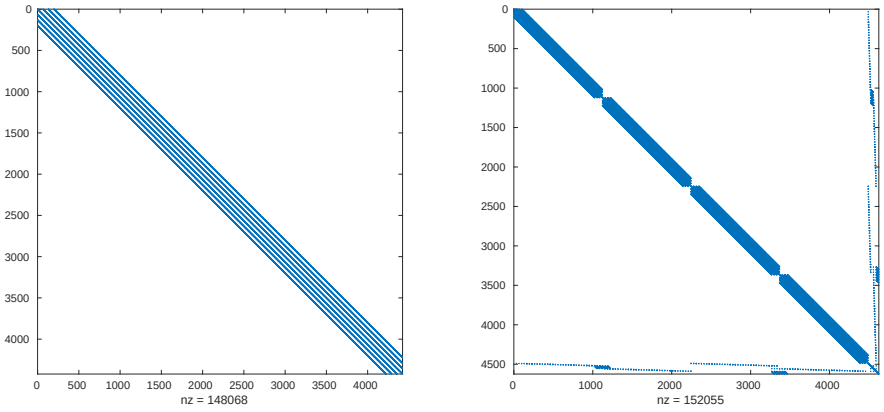


Figure 2.11: Sparsity pattern of the stiffness matrix based on quadratic B-spline basis functions for a single patch (left) and multipatch (right) geometry.

2.7. CONCLUDING REMARKS

In this chapter, we introduced Isogeometric Analysis as a discretization method to solve the convection-diffusion-reaction equation. In particular, we discussed the variational formulation, spatial discretization and B-spline basis functions and their properties. Furthermore, potential refinement strategies and assembly techniques have briefly been introduced.

For other partial differential equations arising in computational science and engineering, an analogous approach as presented here is adopted. In case of time-dependent problems, however, the spatial discretization is typically combined with a time integration method as will be discussed in more detail in Chapter 6.

To solve the resulting linear systems of equations given by Equation (2.17), different solution strategies can be adopted. As we consider the use of a special type of multigrid methods (i.e., p-multigrid methods) in this thesis, the next chapter provides a basic introduction to multigrid methods.

3

Multigrid Methods

This chapter provides a basic introduction to multigrid methods. After discussing the concepts of coarse grid correction and smoothing, the solution procedure of a general multigrid method is presented. In particular, we discuss how multigrid methods can be used as a stand-alone solver or as a preconditioner within a Krylov method. Finally, the state-of-the-art of using multigrid methods to solve linear systems arising in Isogeometric Analysis within the literature is presented.

3.1. INTRODUCTION

Multigrid methods [39–48] are known to be among the most efficient solvers for elliptic problems, such as Equation (2.1), as they can be designed such that the computational costs to solve the resulting linear systems of equations only grows linearly with the number of unknowns. Different types of multigrid methods (e.g. h-multigrid and p-multigrid) exist, which will be discussed in this chapter.
This chapter provides a brief introduction to multigrid methods and is organized as follows: The basics of multigrid are briefly explained in Section 3.2. Section 3.3 then describes the solution procedure, starting from the two-grid cycle. An overview of existing multigrid methods adopted in the context of Isogeometric Analysis is provided in Section 3.4. The chapter ends with some concluding remarks.

3.2. BASIC MULTIGRID

Any multigrid method consist of a smoothing procedure and a coarse grid correction. The combination of both results in a highly efficient method. To describe these basis components, we consider Equation (2.17):

$$\mathbf{A}_{h,p}\mathbf{u}_{h,p} = \mathbf{f}_{h,p}. \tag{3.1}$$

Recall that h denotes the (constant) knot span size and p the spline degree of the B-spline basis functions. Given an approximate solution $\hat{\mathbf{u}}_{h,p}$ of Equation (3.1), we define the *residual* as follows:

$$\mathbf{r}_{h,p} = \mathbf{f}_{h,p} - \mathbf{A}_{h,p}\hat{\mathbf{u}}_{h,p}. \tag{3.2}$$

The *error* is defined as:

$$\mathbf{e}_{h,p} = \mathbf{u}_{h,p} - \hat{\mathbf{u}}_{h,p}. \tag{3.3}$$

Combining Equation (3.2) and (3.3) leads to the *residual equation*:

$$\mathbf{A}_{h,p}\mathbf{e}_{h,p} = \mathbf{r}_{h,p}. \tag{3.4}$$

Solving Equation (3.4) exactly would directly lead to the solution of Equation (3.1). However, since solving the residual equation is as hard as solving the original linear system of equations, it is not useful to do so. Instead, the operator $\mathbf{A}_{h,p}$ is approximated by a simpler operator $\hat{\mathbf{A}}_{h,p}$ and a correction $\hat{\mathbf{e}}_{h,p}$ is determined. Given an initial guess $\mathbf{u}^{(0)}_{h,p}$ and the approximated operator $\hat{\mathbf{A}}_{h,p}$, a natural iterative solution procedure can be defined:

1. Determine the residual:

$$\mathbf{r}_{h,p} = \mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}^{(0)}_{h,p}. \tag{3.5}$$

2. Obtain a correction by approximately solving the residual equation:

$$\hat{\mathbf{A}}_{h,p}\hat{\mathbf{e}}_{h,p} = \mathbf{r}_{h,p}. \tag{3.6}$$

3. Update the initial guess based on this correction:

$$\mathbf{u}_{h,p}^{(1)} = \mathbf{u}_{h,p}^{(0)} + \hat{\mathbf{e}}_{h,p}. \tag{3.7}$$

By combining the different steps of the iterative procedure, we obtain:

$$\begin{aligned}
\mathbf{u}_{h,p}^{(n+1)} &= \mathbf{u}_{h,p}^{(n)} + \hat{\mathbf{e}}_{h,p}, \\
\mathbf{u}_{h,p}^{(n+1)} &= \mathbf{u}_{h,p}^{(n)} + (\hat{\mathbf{A}}_{h,p})^{-1}\mathbf{r}_{h,p}, \\
\mathbf{u}_{h,p}^{(n+1)} &= \mathbf{u}_{h,p}^{(n)} + (\hat{\mathbf{A}}_{h,p})^{-1}\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(n)}\right),
\end{aligned}$$

In this scheme, the operator $\hat{\mathbf{A}}_{h,p}$ still has to be defined. The iterative procedure described above to solve the residual equation leads to a smoothing procedure and coarse grid correction, depending on the choice of $\hat{\mathbf{A}}_{h,p}$, as we will discuss in the remainder of this section.

SMOOTHING

We consider the iterative procedure from the previous section:

$$\mathbf{u}_{h,p}^{(n+1)} = \mathbf{u}_{h,p}^{(n)} + (\hat{\mathbf{A}}_{h,p})^{-1}\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(n)}\right). \tag{3.8}$$

For an efficient procedure, the operator $\hat{\mathbf{A}}_{h,p}$ in Equation (3.8) should represent the original operator $\mathbf{A}_{h,p}$ well. On the other hand, the effect of applying the inverse of $\hat{\mathbf{A}}_{h,p}$ on the residual should be easy to determine. A common choice of $\hat{\mathbf{A}}_{h,p}$ is the following:

$$\hat{\mathbf{A}}_{h,p} = \frac{1}{\omega}\mathbf{D}_{h,p}, \tag{3.9}$$

where $\mathbf{D}_{h,p}$ denotes the diagonal of the original operator $\mathbf{A}_{h,p}$. Note that, in this case, $(\hat{\mathbf{A}}_{h,p})^{-1}$ can easily be determined. This choice of $\hat{\mathbf{A}}_{h,p}$ leads to damped-Jacobi iteration. Another popular choice of $\hat{\mathbf{A}}_{h,p}$ is the lower triangular part of the original operator (including the diagonal), which leads to Gauss-Seidel iteration. In general, all basic iterative methods can be defined by a certain choice of $\hat{\mathbf{A}}_{h,p}$ in Equation (3.8). Within multigrid methods, the commonly adopted basic iterative methods (e.g. Gauss-Seidel) are referred to as *smoother*, as they reduce the high-frequency components of the error. It should be noted, however, that not all basic iterative methods have this property.

COARSE GRID CORRECTION

Instead of choosing $\hat{\mathbf{A}}_{h,p}$ as a part of the original operator, one could also replace $\mathbf{A}_{h,p}$ by a lower-dimensional operator. Considering $\mathbf{A}_{h,p}$ is the result of a discretization with B-spline basis functions of degree p with mesh width h, we could choose $\hat{\mathbf{A}}_{h,p}$ to be the results of a discretization with different values of h and/or p. Following our notation, this would lead to:

$$\hat{\mathbf{A}}_{h,p} = \mathbf{A}_{\tilde{h},\tilde{p}}, \tag{3.10}$$

where $\tilde{h}$ and $\tilde{p}$ have to be chosen. In fact, we are looking for a low-dimensional correction from the space $\mathcal{V}_{\tilde{h},\tilde{p}}$ instead of $\mathcal{V}_{h,p}$. Depending on the choice of $\tilde{h}$ and $\tilde{p}$, a different space $\mathcal{V}_{\tilde{h},\tilde{p}}$ is obtained. Since the dimensions of the coarse grid operator are different from the original operator, operators have to be defined to transfer the residual and error from one space to another. To transfer the error from $\mathcal{V}_{\tilde{h},\tilde{p}}$ to $\mathcal{V}_{h,p}$, a *prolongation* operator is defined, while a *restriction* operator is defined to transfer the residual from $\mathcal{V}_{h,p}$ to $\mathcal{V}_{\tilde{h},\tilde{p}}$:

$$I_{\tilde{h},\tilde{p}}^{h,p} : \mathcal{V}_{\tilde{h},\tilde{p}} \mapsto \mathcal{V}_{h,p}, \qquad I_{h,p}^{\tilde{h},\tilde{p}} : \mathcal{V}_{h,p} \mapsto \mathcal{V}_{\tilde{h},\tilde{p}}. \tag{3.11}$$

Here, both operators act on the vector of coefficients representing a function from the considered spline space as a linear combination of B-spline basis functions. Based on this choice of $\hat{\mathbf{A}}_{h,p}$, the resulting iterative procedure is as follows:

1. Compute the residual at the fine level:

$$\mathbf{r}_{h,p} = \mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(0)}. \tag{3.12}$$

2. Restrict the residual to the coarse level:

$$I_{h,p}^{\tilde{h},\tilde{p}}(\mathbf{r}_{h,p}) = \mathbf{r}_{\tilde{h},\tilde{p}}. \tag{3.13}$$

3. Solve the residual equation to obtain a coarse correction:

$$\mathbf{A}_{\tilde{h},\tilde{p}}\mathbf{e}_{\tilde{h},\tilde{p}} = \mathbf{r}_{\tilde{h},\tilde{p}}. \tag{3.14}$$

4. Prolongate the error to the fine level:

$$I_{\tilde{h},\tilde{p}}^{h,p}(\mathbf{e}_{\tilde{h},\tilde{p}}) = \hat{\mathbf{e}}_{h,p}. \tag{3.15}$$

5. Update the solution according to the obtained correction:

$$\mathbf{u}_{h,p}^{(1)} = \mathbf{u}_{h,p}^{(0)} + \hat{\mathbf{e}}_{h,p}. \tag{3.16}$$

The coarse grid correction can be described, due to the linearity of all operators, in a single formula as follows:

$$\mathbf{u}_{h,p}^{(n+1)} = \left(\mathbf{I}_{h,p} - I_{\tilde{h},\tilde{p}}^{h,p}(\hat{\mathbf{A}}_{h,p})^{-1} I_{h,p}^{\tilde{h},\tilde{p}}\mathbf{A}_{h,p}\right)\mathbf{u}_{h,p}^{(n)}. \tag{3.17}$$

3.3. SOLUTION PROCEDURE

It is widely known that only applying the smoothing procedure or the coarse grid correction, does not result in an efficient (or even converging) method. Therefore, the two-grid solution procedure consists of the combination of smoothing and a coarse grid correction. In particular, we apply *presmoothing*, a *coarse grid correction* and *postsmoothing*:

1. Starting from an initial guess $\mathbf{u}_{h,p}^{(0,0)}$, apply a fixed number ν_1 of presmoothing steps:
$$\mathbf{u}_{h,p}^{(0,m)} = \mathbf{u}_{h,p}^{(0,m-1)} + S_{h,p}\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(0,m-1)}\right), \quad m = 1,\dots,\nu_1, \tag{3.18}$$
where $S_{h,p}$ is a smoothing operator applied to the high-dimensional problem based on the specific choice of $\hat{\mathbf{A}}_{h,p}$ in Equation (3.8).

2. Determine the residual and project it onto the space $\mathcal{V}_{\tilde{h},\tilde{p}}$ using the restriction operator $I_{h,p}^{\tilde{h},\tilde{p}}$:
$$\mathbf{r}_{\tilde{h},\tilde{p}} = I_{h,p}^{\tilde{h},\tilde{p}}\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(0,\nu_1)}\right). \tag{3.19}$$

3. Solve the residual equation to determine the coarse grid error:
$$\mathbf{A}_{\tilde{h},\tilde{p}}\mathbf{e}_{\tilde{h},\tilde{p}} = \mathbf{r}_{\tilde{h},\tilde{p}}. \tag{3.20}$$

4. Project the error $\mathbf{e}_{\tilde{h},\tilde{p}}$ using the prolongation operator $I_{\tilde{h},\tilde{p}}^{h,p}$ and update $\mathbf{u}_{h,p}^{(0,\nu_1)}$:
$$\mathbf{u}_{h,p}^{(0,\nu_1)} := \mathbf{u}_{h,p}^{(0,\nu_1)} + I_{\tilde{h},\tilde{p}}^{h,p}\left(\mathbf{e}_{\tilde{h},\tilde{p}}\right). \tag{3.21}$$

5. Apply ν_2 postsmoothing steps described by (3.18) to obtain $\mathbf{u}_{h,p}^{(0,\nu_1+\nu_2)} =: \mathbf{u}_{h,p}^{(1,0)}$.

Note that in the solution procedure, presmoothing and postsmoothing are identical. However, in case Gauss-Seidel is combined with a Conjugate Gradient method, a forward and backward sweep will be applied in this thesis, respectively, to keep the multigrid method symmetric. In literature, steps (2) – (4) are referred to as coarse grid correction. The two-grid cycle can be extended to a multigrid cycle by recursively applying the procedure above on Equation (3.20). At the coarsest level, the residual equation is then typically solved by means of direct solver. In case the aforementioned procedure is applied exactly once, the resulting cycle is referred to as a V-cycle. Other types are possible, however, as shown in Figure 3.1.

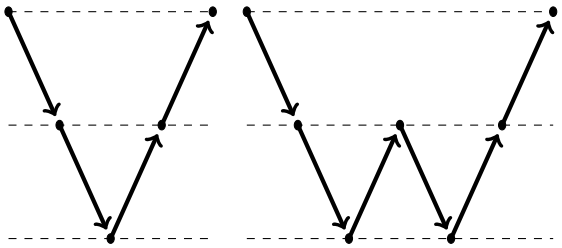


Figure 3.1: Description of a V-cycle (left) and W-cycle (right).

This iterative procedure is repeated until a certain stopping criterion is satisfied. Throughout this thesis, this criterion is based on a reduction of the relative residual:
$$\frac{||\mathbf{r}_{h,p}^{(i)}||_2}{||\mathbf{r}_{h,p}^{(0)}||_2} < \epsilon. \tag{3.22}$$

Here, $\mathbf{r}_{h,p}^{(i)}$ denotes the residual after iteration i and $||\cdot||_2$ the Euclidean norm. Throughout this thesis, a random initial guess is chosen, where each entry is sampled from a uniform distribution on the interval $[-1,1]$ using the same seed. Note that, in case a zero initial guess is adopted, $\mathbf{r}_{h,p}^{(0)}$ can be replaced by $\mathbf{f}_{h,p}$ due to linearity.
Although the general description for each multigrid method is identical, each type of multigrid method is defined by a specific choice of the coarse grid operator, the prolongation and restriction operator and smoother. Throughout this thesis, we will distinguish two main types of multigrid methods: h-multigrid and p-multigrid methods.

3

h-multigrid methods

Within h-multigrid methods, the coarse grid operator is obtained adopting larger mesh widths, but keeping the same spline degree. Typically, this is done by setting $\tilde{h} = 2h$:

$$\hat{\mathbf{A}}_{h,p} = \mathbf{A}_{2h,p}. \tag{3.23}$$

This choice of $\hat{\mathbf{A}}_{h,p}$ has a clear geometric interpretation, as it refers to the operator obtained by a discretization adopting B-spline basis functions with knot span size $2h$ and spline degree p. Since the space $\mathcal{V}_{2h,p}$ is contained in $\mathcal{V}_{h,p}$, a natural choice for the prolongation operator is *injection* or the *canonical embedding*. The restriction operator is then defined as the transposed of the prolongation operator:

$$I_{h,p}^{2h,p} = I_{2h,p}^{h,p}. \tag{3.24}$$

Note that, with h-multigrid methods, the size of the coarse grid operator is significantly smaller compared to the size of the original operator. Therefore, applying the smoother at the coarser levels and solving the residual equation is relatively cheap. As the degree of the B-spline basis functions remains the same with h-multigrid methods, the sparsity pattern is similar at all level of the multigrid hierarchy. Throughout this dissertation, we will use h-multigrid methods to assess the performance of our p-multigrid method. Furthermore, the residual equation in our p-multigrid method at the low-order level will be solved by means of an h-multigrid method.

p-multigrid methods

With p-multigrid methods the coarse grid operator is chosen by adopting a low-order discretization while keeping the same mesh width, for example $\tilde{p} = p - 1$:

$$\hat{\mathbf{A}}_{h,p} = \mathbf{A}_{h,p-1}. \tag{3.25}$$

Here, $\mathbf{A}_{h,p-1}$ corresponds to a discretization with B-spline basis functions of degree $p-1$ and knot span size h. In contrast to h-multigrid methods, the number of degrees of freedom is similar at the coarser level with p-multigrid methods. However, the sparsity pattern differs significantly, as B-spline basis functions of a different spline degree are considered. To transfer errors and residuals between different levels of the multigrid hierarchy, prolongation and restriction operators have to be defined. In Chapter 4 we discuss the proposed prolongation and restriction operator for the p-multigrid method in more detail.

MULTIGRID METHODS AS A PRECONDITIONER

Multigrid methods can be used as a stand-alone solver as described before, or as a preconditioner within an outer Krylov solver. That is, instead of applying the preconditioner directly, the equivalent linear system of equations is solved with an intentionally reduced tolerance or a fixed number of multigrid cycles. Using an intentionally reduced tolerance should be done with care, however, as it transforms multigrid from a linear preconditioner to a non-linear preconditioner. Combining a multigrid method with a Krylov solver can not only result in a speed-up of the computation, but might also lead to a more robust overall method. It should be mentioned that the resulting preconditioned Krylov solver might still converge in $\mathcal{O}(N_{\text{dof}})$ iterations.

The application of a symmetric multigrid method implies that the preconditioner is symmetric. Provided the matrix is symmetric and positive definite (SPD), the Conjugate-Gradient method [74] can be adopted as an outer Krylov solver. Alternatively, Bi-CGSTAB [75], GMRES [76] or IDR(s) [77] can be adopted for non-symmetric preconditioners. The preconditioned Conjugate-Gradient method applied to the system $\mathbf{Ax} = \mathbf{b}$ is given by:

Preconditioned Conjugate-Gradient method

- $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ determine residual
- $\mathbf{z}_0 := \mathbf{M}^{-1}\mathbf{r}_0$ apply preconditioner
- $\mathbf{p}_0 := \mathbf{z}_0$ initialize search direction vector

while $\left(\frac{||\mathbf{r}_k||_2}{||\mathbf{r}_0||_2} > \epsilon\right)$

- $\alpha_k := \frac{\mathbf{r}_k^\top \mathbf{z}_k}{\mathbf{p}_k^\top \mathbf{A}\mathbf{p}_k}$ determine step length
- $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$ update solution vector
- $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$ update residual vector
- **if** $\left(\frac{||\mathbf{r}_{k+1}||_2}{||\mathbf{r}_0||_2}\right)$ exit loop, **else**:
- $\mathbf{z}_{k+1} := \mathbf{M}^{-1}\mathbf{r}_{k+1}$ apply preconditioner
- $\beta_k := \frac{\mathbf{r}_{k+1}^\top \mathbf{z}_{k+1}}{\mathbf{r}_k^\top \mathbf{z}_k}$ determine step length
- $\mathbf{p}_{k+1} := \mathbf{z}_{k+1} + \beta_k \mathbf{p}_k$ update search direction vector

end

Here, the preconditioner $\mathbf{M}^{-1}$ is applied by applying a single multigrid cycle on the related linear system of equations, for example $\mathbf{A}\mathbf{z}_k = \mathbf{r}_k$. As a stopping criterion for the outer Krylov method, we adopt a reduction of the relative residual: $\frac{||\mathbf{r}_k||_2}{||\mathbf{r}_0||_2}$. Throughout this thesis, the Bi-CGSTAB method is adopted as an outer Krylov method in case of a non-

symmetric preconditioner. Note that, two iterations of the Conjugate-Gradient method is roughly as expensive as a single Bi-CGSTAB iteration. The preconditioned Bi-CGSTAB method is given by:

Preconditioned Bi-CGSTAB method

- $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ determine residual
- $\rho_0 = \alpha_0 = \omega_0 := 1$ initialize step length
- $\mathbf{v}_0 = \mathbf{p}_0 := \mathbf{0}$ initialize search direction vector

while $\left(\frac{||\mathbf{r}_k||_2}{||\mathbf{r}_0||_2} > \epsilon\right)$

- $\rho_k = (\mathbf{r}_0^\top, \mathbf{r}_{k-1})$
- $\beta = (\rho_k/\rho_{k-1})(\alpha/\omega_{k-1})$ determine step length
- $\mathbf{p}_k = \mathbf{r}_{k-1} + \beta(\mathbf{p}_{k-1} - \omega_{k-1}\mathbf{v}_{k-1})$ update search direction vector
- $\mathbf{y} = \mathbf{M}^{-1}\mathbf{p}_k$ apply preconditioner
- $\mathbf{v}_k = \mathbf{A}\mathbf{y}$
- $\alpha_k = \rho_k/(\mathbf{r}_0^\top, \mathbf{v}_k)$ determine step length
- $\mathbf{h} = \mathbf{x}_{k-1} + \alpha_k\mathbf{y}$ update solution vector
- **if** $(||\mathbf{h}||_2 < \epsilon)$ $\mathbf{x}_k = \mathbf{h}$, exit loop **else**:
- $\mathbf{s} = \mathbf{r}_{k-1} - \alpha_k\mathbf{v}_k$ update residual vector
- $\mathbf{z} = \mathbf{M}^{-1}\mathbf{s}$ apply preconditioner
- $\mathbf{t} = \mathbf{A}\mathbf{z}$
- $\omega_k = (\mathbf{M}^{-1}\mathbf{t}, \mathbf{M}^{-1}\mathbf{s})/(\mathbf{M}^{-1}\mathbf{t}, \mathbf{M}^{-1}\mathbf{t})$ determine step length
- $\mathbf{x}_k = \mathbf{h} + \omega_k\mathbf{z}$ update solution vector
- **if** $(||\mathbf{r}_k||/||\mathbf{r}_0|| < \epsilon)$ exit loop
- $\mathbf{r}_k = \mathbf{s} - \omega_k\mathbf{t}$ update residual vector

end

3.4. MULTIGRID METHODS FOR ISOGEOMETRIC ANALYSIS

Over the years, various multigrid methods have been applied to solve linear systems of equations arising in Isogeometric Analysis. This section provides a global overview of the existing literature on the use of multigrid methods for Isogeometric Analysis.

A first study on h-multigrid methods in the context of Isogeometric Analysis was presented in [78] for second-order elliptic equations. In this paper, both the approximation and smoothing property of the two-grid solver were analyzed, implying h-independence of this solver. Furthermore, numerical results in both two and three dimensions were presented using (forward) Gauss-Seidel as a smoother. The resulting multigrid method showed, indeed, h-independency, but the number of iterations needed to achieve convergence depended on the spline degree p. The paper showed as well that p-refinement (i.e., C^0-continuous functions) leads to slower convergence compared to k-refinement, when the same spline degree is considered. The use of p-multigrid as an alternative solution strategy was already mentioned in this paper (see Remark 10).

A numerical spectral analysis for h-multigrid methods was presented in [79] and showed that standard smoothers (i.e., Gauss-Seidel and Jacobi) fail to reduce the high-frequency error components for higher values of p. These problems became even more prominent in higher dimensions. Furthermore, the use of h-multigrid methods as a preconditioner within a Conjugate-Gradient method was investigated, showing a reduced number of iterations even though a dependency on p still remained visible. Numerical results with a full multigrid method, in contrast to the often adopted V-cycle, were presented in [80], showing that a very limited amount of cycles were needed to obtain the optimal order of spatial convergence.

These observations led to the development of non-standard smoothers to overcome this problem. For example, a boundary-corrected mass-Richardson smoother has been developed [81] for which it was shown that the convergence rate is indeed independent of both h and p. Numerical results, obtained for one- and two-dimensional benchmark problems, confirm this. In a continuation of this work by the same authors, a smoother based on the stable splitting of spline spaces was presented [82]. As with the boundary-corrected mass-Richardson smoother, a special treatment for the boundary is suggested to capture the spectral difficulties encountered in Isogeometric Analysis. In particular, the spline space is splitted in a (large) interior part and multiple smaller spaces to capture the boundary effects.

An alternative approach is the use of multiplicative Schwarz methods as a smoother [83]. Based on a local Fourier analysis (LFA), the optimal size of the blocks within this smoother has been determined. Numerical results, obtained in one and two dimensions, showed the independence of both h and p for Poisson's equation on single patch geometries. This smoother has been applied as well to the biharmonic equation [84], where a local Fourier analysis is performed to predict the behaviour of the multigrid algorithm. Convergence rates independent of h and p are shown in one and two dimensions, provided that the block size is adjusted based on the considered spline degree p.

As an alternative, a preconditioned Krylov method can be applied as a smoother. In [85] such a multigrid method has been designed, based on the careful analysis of the spectral properties of matrices arising in Isogeometric Analysis. The resulting multigrid method shows, indeed, convergence rates independent of h and p and outperforms a similar

multigrid method adopting Gauss-Seidel as a smoother on the fine level.
Further research then led to the development of a parallel multigrid solver [86] for multipatch geometries. Here, each patch was assigned to a single processor as it would be the case in non-overlapping domain decomposition methods. For the resulting multigrid method, both weak and strong scaling were investigated. In [87], estimates of the approximation error were derived for two-dimensional multipatch geometries and a multigrid method for conforming and non-conforming multipatch geometries was presented in [88].

In general, most of the proposed multigrid methods were initially applied to Poisson type problems. However, research has been done on robust h-multigrid solvers for the biharmonic equation [89], in which Gauss-Seidel smoothing was combined with the subspace corrected mass-smoother. It was shown that the Gauss-Seidel smoother is able to capture the effects of the geometry transformation quite well and leads to a lower number of iterations compared to the mass-smoother (unless the spline degree p is very high). On the other hand, the mass-smoother is robust in the spline degree, but does not perform well on more complex geometries. Although the hybrid smoother is approximately twice as expensive as each smoother separately, the overall costs with the hybrid smoother are lower compared to each of the smoothers applied separately.

3.5. CONCLUDING REMARKS

In this chapter, we briefly discussed the basics of multigrid methods, including the coarse grid correction and smoothing procedure. Furthermore, both h-multigrid and p-multigrid methods were introduced and an overview has been presented of the application of multigrid methods in Isogeometric Analysis. In the next chapter, we present a p-multigrid method for discretizations arising in Isogeometric Analysis which will be applied to the CDR-equation on single patch and multipatch geometries throughout this thesis. Finally, the p-multigrid method will be applied in the context of the Multigrid Reduced in Time (MGRIT) method.

4

P-MULTIGRID METHODS FOR ISOGEOMETRIC ANALYSIS

This chapter presents the p-multigrid method and its components as adopted throughout this dissertation in detail. A spectral analysis is then performed to analyze the interplay between the coarse grid correction and smoother for the p-multigrid method. Numerical results, obtained with the p-multigrid method adopting different smoothers, are compared to h-multigrid methods in terms of iteration numbers and computational efficiency. Finally, the proposed p-multigrid method is applied to THB-spline discretizations.

Parts of this chapter have been published in:
R.Tielen, M. Möller, D. Göddeke and C.Vuik, *p-multigrid methods and their comparison to h-multigrid methods within Isogeometric Analysis*, Computer Methods in Applied Mechanics and Engineering **372** (2020)

Parts of this chapter have been published in:
R.Tielen, M. Möller and C.Vuik: *A direct projection to low-order level for p-multigrid methods in Isogeometric Analysis*, in The Proceedings of the European Numerical Mathematics and Advanced Applications Conference (2019)

4.1. INTRODUCTION

The previous chapters provided a brief introduction to Isogeometric Analysis and multigrid methods. This chapter focusses on our p-multigrid method and its application to linear systems arising in Isogeometric Analysis for single patch geometries. In Section 4.2, the considered benchmarks are presented, while the p-multigrid method is presented in Section 4.3 and a spectral analysis is performed in Section 4.4. Numerical results obtained for the p-multigrid method are presented in Section 4.5 and compared to h-multigrid methods in Section 4.6. In particular, we compare the performance of both multigrid methods when adopting a state-of-the-art smoother in Section 4.7. Finally, we apply our p-multigrid method in the context of THB-splines in Section 4.8. The chapter ends with some concluding remarks.

4.2. BENCHMARKS

To analyze the performance of the proposed p-multigrid method, different benchmarks are considered. In this chapter, we consider a variety of two- and three-dimensional benchmarks on single patch geometries:

Benchmark 1. Let Ω be the quarter annulus with an inner and outer radius of 1 and 2, respectively. The coefficients are chosen as follows:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad R = 0. \tag{4.1}$$

Furthermore, homogeneous Dirichlet boundary conditions are applied and the right-hand side is chosen such that the exact solution u is given by

$$u(x,y) = -(x^2+y^2-1)(x^2+y^2-4)xy^2.$$

Benchmark 2. Here, the unit square is adopted as domain of interest, i.e., $\Omega = [0,1]^2$, and the coefficients are chosen as follows:

$$\mathbf{D} = \begin{bmatrix} 1.2 & -0.7 \\ -0.4 & 0.9 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} 0.4 \\ -0.2 \end{bmatrix}, \qquad R = 0.3. \tag{4.2}$$

Homogeneous Dirichlet boundary conditions are applied and the right-hand side is chosen such that the exact solution u is given by

$$u(x,y) = \sin(\pi x)\sin(\pi y).$$

Benchmark 3. The unit cube is here adopted as domain of interest, i.e., $\Omega = [0,1]^3$, and the coefficients are chosen as follows:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \qquad R = 0. \tag{4.3}$$

Homogeneous Dirichlet boundary conditions are applied and the right-hand side is chosen such that the exact solution u is given by:

$$u(x,y) = \sin(\pi x)\sin(\pi y)\sin(\pi z).$$

These benchmarks investigate the influence of the geometric factor, coefficients in the CDR equation and spatial dimension on the performance of the p-multigrid method. More information regarding the adopted geometry functions for these benchmarks can be found in Chapter 7. Multipatch geometries, which are often encountered in more practical engineering problems, will be considered in Chapter 5.

4.3. p-MULTIGRID METHOD

As discussed in the previous chapter, multigrid methods aim to solve linear systems of equations by defining a hierarchy of discretizations. At each level of the multigrid hierarchy a smoother is applied, whereas on the coarsest level a correction is determined by means of a direct solver. With p-multigrid methods, this correction is based on a low-order discretization. In this section, we present our p-multigrid method in detail.
Starting from $\mathcal{V}_{h,1}$, a sequence of spaces $\mathcal{V}_{h,1},\dots,\mathcal{V}_{h,p}$ is obtained by applying k-refinement. Note that, since basis functions with maximal continuity are considered, the spaces are not nested. A single step of the two-grid correction scheme for the p-multigrid method consists of the following steps:

1. Starting from an initial guess $\mathbf{u}_{h,p}^{(0,0)}$, apply a fixed number ν_1 of presmoothing steps:
$$\mathbf{u}_{h,p}^{(0,m)} = \mathbf{u}_{h,p}^{(0,m-1)} + S_{h,p}\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(0,m-1)}\right), \quad m = 1,\dots,\nu_1, \tag{4.4}$$
where $S_{h,p}$ is a smoothing operator applied to the high-order problem.

2. Determine the residual at level p and project it onto the space $\mathcal{V}_{h,p-1}$ using the restriction operator I_p^{p-1}:
$$\mathbf{r}_{h,p-1} = I_p^{p-1}\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(0,\nu_1)}\right). \tag{4.5}$$

3. Solve the residual equation at level $p-1$ to determine the coarse grid error:
$$\mathbf{A}_{h,p-1}\mathbf{e}_{h,p-1} = \mathbf{r}_{h,p-1}. \tag{4.6}$$

4. Project the error $\mathbf{e}_{h,p-1}$ onto the space $\mathcal{V}_{h,p}$ using the prolongation operator I_{p-1}^{p} and update $\mathbf{u}_{h,p}^{(0,\nu_1)}$:
$$\mathbf{u}_{h,p}^{(0,\nu_1)} := \mathbf{u}_{h,p}^{(0,\nu_1)} + I_{p-1}^{p}\left(\mathbf{e}_{h,p-1}\right). \tag{4.7}$$

5. Apply ν_2 postsmoothing steps described by (4.4) to obtain $\mathbf{u}_{h,p}^{(0,\nu_1+\nu_2)} =: \mathbf{u}_{h,p}^{(1,0)}$.

Recursive application of this scheme until level $p = 1$ is reached, leads to a V-cycle. In contrast to h-multigrid methods, the coarsest problem in p-multigrid methods can still be large for small values of h. However, since we restrict to level $p = 1$, the coarse grid problem corresponds to a standard low-order Lagrange FEM discretization of the problem at hand. Therefore, we use a standard h-multigrid method to solve the coarse grid problem in our p-multigrid scheme, which is known to be optimal (in particular h-independent) in this case. As a smoother, Gauss-Seidel is applied within the h-multigrid method, as it is both cheap and effective for low-order problems. Applying a single W-cycle using canonical prolongation, weighted restriction and a single smoothing step turned out to be sufficient and has therefore been adopted throughout this thesis as coarse grid solver.

DIRECT OR INDIRECT PROJECTION

Note that, for the p-multigrid method, the residual can be projected directly to each p-level or directly to level $p = 1$. We refer to these strategies as an indirect or direct projection, respectively. Figure 4.1 illustrates both coarsening strategies within the p-multigrid method. The use of a direct projection would lead to lower setup costs, as less p-levels have to be considered. Furthermore, the costs of a single p-multigrid cycle would decrease. The question remains, however, to which extend this direct projection influences the obtained convergence rate of the resulting p-multigrid method.
It was shown in [90] that a direct projection hardly affects the performance of the p-multigrid method (in terms of iteration numbers), while the setup costs indeed decrease significantly. In Appendix A, numerical results are presented for the first benchmark confirming this observation. Throughout this dissertation, a direct projection to level $p = 1$ is therefore adopted for the p-multigrid method.

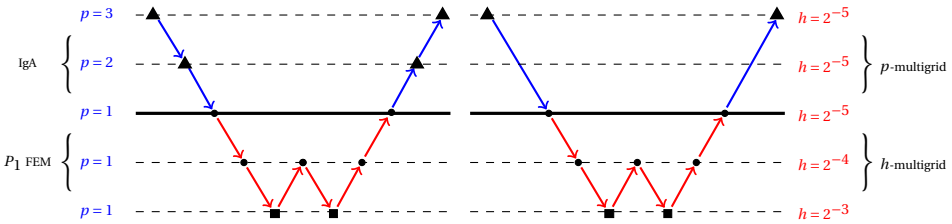


Figure 4.1: Illustration of both an indirect (left) and direct (right) projection scheme within p-multigrid. At $p = 1$, Gauss-Seidel is always adopted as a smoother (•), whereas at the high order level different smoothers will be applied (▲). At the coarsest level, a direct solver is applied to solve the residual equation (■)

PROLONGATION AND RESTRICTION

To transfer coarse grid corrections and residuals between different levels of the p-multigrid hierarchy, prolongation and restriction operators are defined. The prolongation and restriction operator adopted in this thesis are based on a (discrete) L_2 projection and have been used extensively in the literature [91–93]. The coarse grid error at level $p = 1$ (see Section 4.3) is prolongated directly to level p by projection onto the space $\mathcal{V}_{h,p}$. The prolongation operator $I_1^p : \mathcal{V}_{h,1} \rightarrow \mathcal{V}_{h,p}$ is given by

$$I_1^p(\mathbf{v}_1) = (\mathbf{M}_p)^{-1}\mathbf{P}_1^p\,\mathbf{v}_1. \tag{4.8}$$

Here, the mass matrix $\mathbf{M}_p$ and transfer matrix $\mathbf{P}_1^p$ are defined as follows:

$$(\mathbf{M}_p)_{(i,j)} := \int_\Omega \Phi_{i,p}\Phi_{j,p}\,\mathrm{d}\Omega, \qquad (\mathbf{P}_1^p)_{(i,j)} := \int_\Omega \Phi_{i,p}\Phi_{j,1}\,\mathrm{d}\Omega, \tag{4.9}$$

where the value of the integrals is determined by adopting Gaussian quadrature. The residuals are restricted from level p to level 1 by projection onto the space $\mathcal{V}_{h,1}$. The restriction operator $I_p^1 : \mathcal{V}_{h,p} \rightarrow \mathcal{V}_{h,1}$ is defined by

$$I_p^1(\mathbf{v}_p) = (\mathbf{M}_1)^{-1}\mathbf{P}_p^1\,\mathbf{v}_p. \tag{4.10}$$

To prevent the explicit solution of a linear system of equations for each projection step, the consistent mass matrix $\mathbf{M}_p$ in both transfer operators can be replaced by a diagonal matrix $\mathbf{M}_p^L$ by applying row-sum lumping:

$$\mathbf{M}^L_{p_{(i,i)}} = \sum_{j=1}^{N_{\text{dof}}} \mathbf{M}_{p_{(i,j)}}. \tag{4.11}$$

Numerical experiments, presented in Appendix B, show that lumping the mass matrix hardly influences the convergence behaviour of the resulting p-multigrid method. More precisely, the number of iterations needed to reach convergence for the p-multigrid method (using ILUT as a smoother) is identical for all considered configurations. When Gauss-Seidel is applied as a smoother, the number of iterations slightly varies, but remains very similar. It should be noted that the overall accuracy obtained with the p-multigrid method is not affected when the lumped projection is adopted. However, the use of a lumped mass matrix significantly reduces the computational costs of the p-multigrid method, as no explicit solves are necessary when applying the restriction or prolongation operator. Alternatively, one could invert the mass matrix efficiently by exploiting the tensor product structure, see e.g. [94], but this has not been explored in this thesis.

It should be noted that this choice of prolongation and restriction operators yields a non-symmetric coarse grid correction and, hence, a non-symmetric multigrid solver. As a consequence, the multigrid solver will only be applied as a preconditioner for a Krylov method suited for non-symmetric matrices, like Bi-CGSTAB.

Remark: Choosing the prolongation and restriction operator tranpose to each other would restore the symmetry of the multigrid method. Furthermore, only one of the lumped mass matrices $\mathbf{M}_1^L$ or $\mathbf{M}_p^L$ has to be assembled, which leads to a slight decrease of the computational costs. However, numerical experiments, not presented in this thesis, show that this leads to a less robust p-multigrid method. More precisely, the number of iterations needed to reach convergence increases significantly for some of the considered benchmark problems. Choosing the prolongation and restriction operator tranpose to each other has not been considered without lumping the mass matrices, as it would increase the computational costs too much. Therefore, the prolongation and restriction operators are adopted as defined in Equation (4.8) and (4.10), respectively.

SMOOTHER

Within multigrid methods, a basic iterative method (e.g. damped Jacobi) is typically used as a smoother. However, it is known from literature that the performance of classical smoothers such as (damped) Jacobi and Gauss-Seidel decreases significantly for higher values of p. Therefore, apart from Gauss-Seidel, the use of an Incomplete LU Factorization with a dual threshold strategy (ILUT) [95] to approximate the operator $\mathbf{A}_{h,p}$ is investigated. The ILUT factorization considered in this dissertation is closely related to an ILU(0) factorization [96]. This factorization has been applied in the context of Isogeometric Analysis as a preconditioner, showing good convergence behaviour [97].

An ILUT factorization decomposes the operator in a lower and upper triangular part:

$$\mathbf{A}_{h,p} \approx \mathbf{L}_{h,p}\mathbf{U}_{h,p}. \tag{4.12}$$

The ILUT factorization is determined completely by a tolerance τ and fillfactor ρ. Two dropping rules are applied during factorization:

1. All elements smaller (in absolute value) than the dropping tolerance are dropped. The dropping tolerance is obtained by multiplying the tolerance τ with the average magnitude of all elements in the current row.

2. Apart from the diagonal element, only the M largest elements are kept in each row. Here, M is determined by multiplying the fillfactor ρ with the average number of non-zeros in each row of the original operator $\mathbf{A}_{h,p}$.

An efficient implementation of ILUT is available in the Eigen library [98] based on [99]. Once the factorization is obtained, a single smoothing step is applied as follows:

$$\begin{aligned}
\mathbf{e}_{h,p}^{(n)} &= (\mathbf{L}_{h,p}\mathbf{U}_{h,p})^{-1}\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(n)}\right), && (4.13)\\
&= \mathbf{U}_{h,p}^{-1}\mathbf{L}_{h,p}^{-1}\left(\mathbf{f}_{h,p} - \mathbf{A}_{h,p}\mathbf{u}_{h,p}^{(n)}\right), && (4.14)\\
\mathbf{u}_{h,p}^{(n+1)} &= \mathbf{u}_{h,p}^{(n)} + \mathbf{e}_{h,p}^{(n)}, && (4.15)
\end{aligned}$$

where the two matrix inversions in Equation (4.14) amount to forward and backward substitution. Throughout this thesis, the fillfactor $\rho = 1$ is used (unless stated otherwise) and the dropping tolerance equals $\tau = 10^{-12}$. Hence, the number of non-zero elements of $\mathbf{L}_{h,p} + \mathbf{U}_{h,p}$ is similar to the number of non-zero elements of $\mathbf{A}_{h,p}$. Figure 4.2 shows the sparsity pattern of the stiffness matrix $\mathbf{A}_{h,3}$ and $\mathbf{L}_{h,3} + \mathbf{U}_{h,3}$ for the first benchmark and $h = 2^{-5}$. Since a fill-reducing permutation is performed during the ILUT factorization within the Eigen library, sparsity patterns differ significantly. However, the number of non-zero entries of $\mathbf{A}_{h,3}$ and $\mathbf{L}_{h,3} + \mathbf{U}_{h,3}$ is comparable. In fact, due to the considered dropping rules, the number of non-zero entries of $\mathbf{L}_{h,3} + \mathbf{U}_{h,3}$ is even slightly smaller.

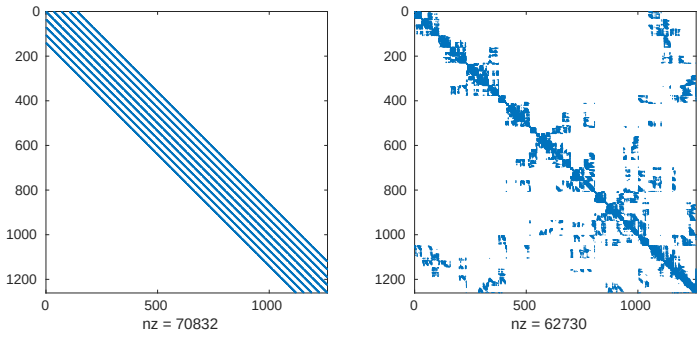


Figure 4.2: Sparsity pattern of $\mathbf{A}_{h,3}$ (left) and $\mathbf{L}_{h,3}+\mathbf{U}_{h,3}$ (right) for $h=2^{-5}$.

4.4. SPECTRAL ANALYSIS

In this section, the performance of the proposed p-multigrid method is analyzed in different ways. First, a spectral analysis is performed to investigate the interplay between the coarse grid correction and the smoother. In particular, we compare the effectiveness of two smoothers (Gauss-Seidel and ILUT) for different values of p. Furthermore, the spectral radius of the iteration matrix is determined to obtain the asymptotic convergence rates of the p-multigrid methods. Throughout this section, the geometries of the first two benchmarks (see Section 4.2) are considered for the analysis.

REDUCTION FACTORS

To investigate the effect of a single smoothing step or coarse grid correction, a spectral analysis [79] is carried out for different values of p. For this analysis, we consider $-\Delta u=0$ with homogeneous Dirichlet boundary conditions and, hence, $u=0$ as its exact solution. Let us define the error reduction factors as follows:

$$r^{S}(\mathbf{u}^0_{h,p})=\frac{||S_{h,p}(\mathbf{u}^0_{h,p})||_2}{||\mathbf{u}^0_{h,p}||_2},\qquad r^{CGC}(\mathbf{u}^0_{h,p})=\frac{||CGC(\mathbf{u}^0_{h,p})||_2}{||\mathbf{u}^0_{h,p}||_2},\tag{4.16}$$

where $S_{h,p}(\cdot)$ denotes a single smoothing step and $CGC(\cdot)$ an exact coarse grid correction. For the coarse grid correction, a direct projection to $p=1$ is considered. As an initial guess, the generalized eigenvectors $\mathbf{v}_i$ are chosen which satisfy

$$\mathbf{A}_{h,p}\mathbf{v}_i=\lambda_i\mathbf{M}_{h,p}\mathbf{v}_i,\quad i=1,\dots,N_{\text{dof}}.\tag{4.17}$$

Here, $\mathbf{M}_{h,p}$ is the consistent mass matrix. The error reduction factors for the first benchmark for both smoothers are shown in Figure 4.3 for $h=2^{-5}$ and different values of p. The reduction factors obtained with Gauss-Seidel as a smoother are shown in the left column, while the plots in the right column show the reduction factors for the ILUT smoother. In general, the coarse grid corrections reduce the coefficients of the eigenvector expansion corresponding to the low-frequency modes, while the smoother reduces

the coefficients associated with high-frequency modes. However, for increasing values of p, the reduction factors of the Gauss-Seidel smoother increase for the high-frequency modes, implying that the smoother becomes less effective. On the other hand, the use of ILUT as a smoother leads to decreasing reduction factors for all modes when the value of p is increased.
Figure 4.4 shows the error reduction factors obtained for the second benchmark, showing similar, but less oscillatory, behaviour. These results indicate that the use of ILUT as a smoother (with $\nu_1 = \nu_2 = 1$) could significantly improve the convergence properties of the p-multigrid method compared to the use of Gauss-Seidel as a smoother.

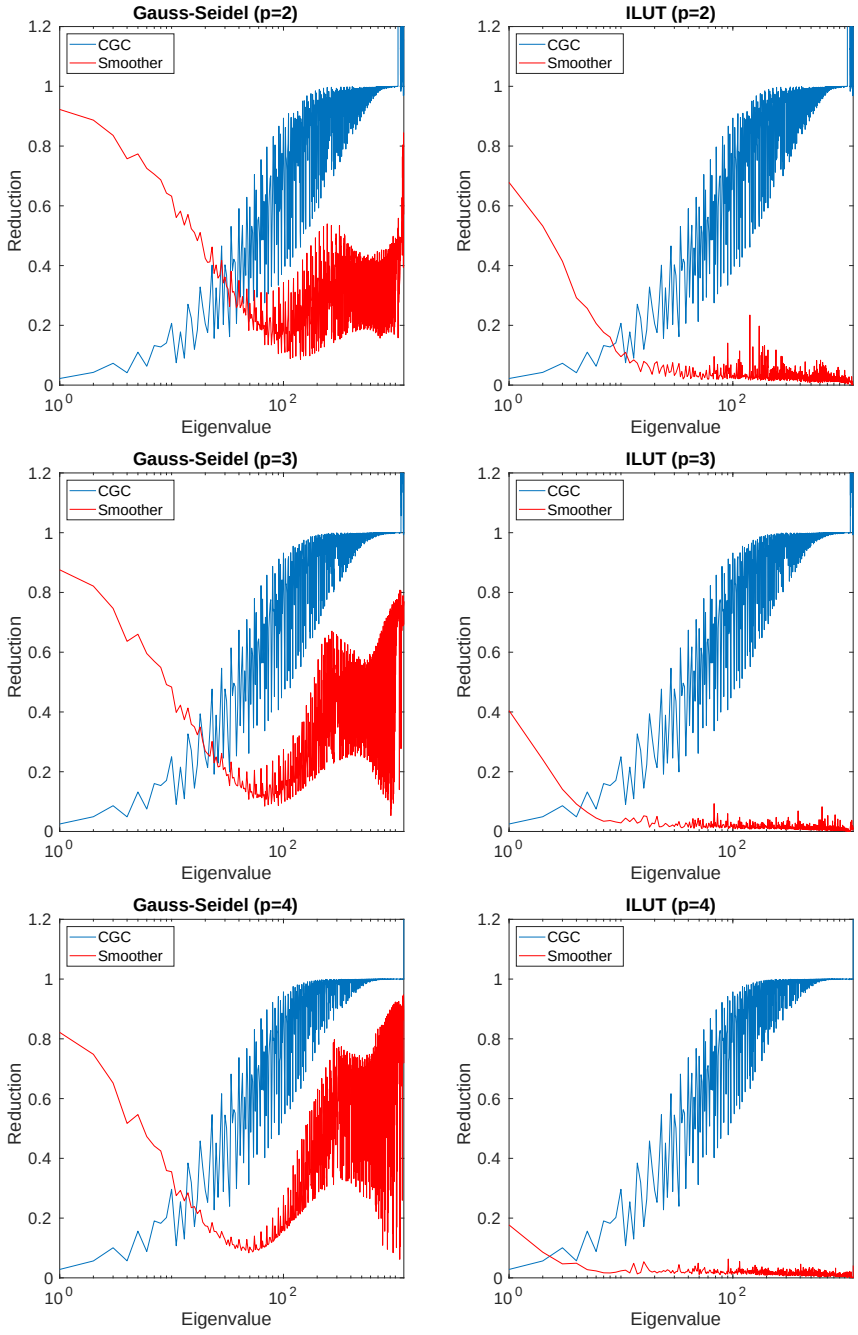


Figure 4.3: Error reduction in $(\mathbf{v}_j)$ for Poisson's equation on the quarter annulus with $p = 2,3,4$ and $h = 2^{-5}$ obtained with Gauss-Seidel (left) and ILUT (right) as a smoother.

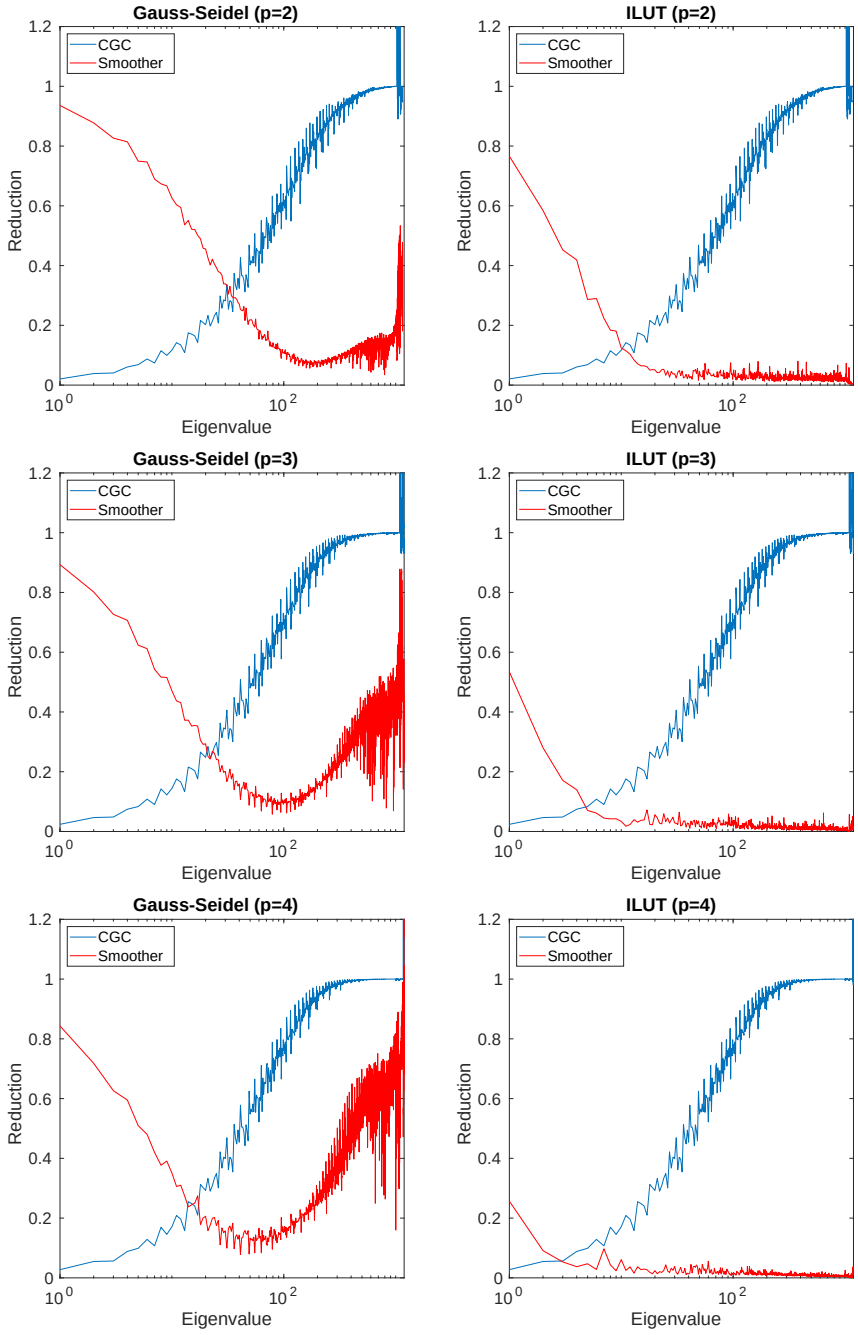


Figure 4.4: Error reduction in $(\mathbf{v}_j)$ for Poisson's equation on the unit square with $p = 2,3,4$ and $h = 2^{-5}$ obtained with Gauss-Seidel (left) and ILUT (right) as a smoother.

ITERATION MATRIX

For any multigrid method, the asymptotic convergence rate is determined by the spectral radius of the iteration matrix. To obtain this matrix explicitly, consider, again, $-\Delta u = 0$ with homogeneous Dirichlet boundary conditions and, hence, $u = 0$ as its exact solution. By applying a single iteration of the p-multigrid method using the unit vector $\mathbf{e}^i_{h,p}$ as initial guess, one obtains the i^{th} column of the iteration matrix [100].

Figure 4.5 shows the spectra for the first benchmark for $h = 2^{-5}$ and different values of p obtained with both smoothers. For reference, the unit circle has been added in all plots. The spectral radius of the iteration matrix, defined as the maximum eigenvalue in absolute value, is then given by the eigenvalue that is the furthest away from the origin. Clearly, the spectral radius significantly increases for higher values of p when adopting Gauss-Seidel as a smoother. The use of ILUT as a smoother results in spectra clustered around the origin, implying fast convergence of the resulting p-multigrid.

The spectra of the iteration matrices for the second benchmark are presented in Figure 4.6. Although the eigenvalues are more clustered with Gauss-Seidel compared to the first benchmark, the same behaviour can be observed.

The spectral radii for both benchmarks, where $\nu_1 = \nu_2 = 1$, are presented in Table 4.1. For Gauss-Seidel, the spectral radius of the iteration matrix is independent of the mesh width h, but depends strongly on the spline degree p. The use of ILUT leads to a spectral radius which is significantly lower for all values of h and p. Although ILUT exhibits a small h-dependence, the spectral radius remains low for all values of h. As a consequence, the p-multigrid method is expected to show essentially both h- and p-independence convergence behaviour when ILUT is adopted as a smoother.

$p=2$	GS	ILUT	$p=3$	GS	ILUT	$p=4$	GS	ILUT
$h=2^{-4}$	0.635	0.014	$h=2^{-4}$	0.849	0.003	$h=2^{-4}$	0.963	0.003
$h=2^{-5}$	0.631	0.039	$h=2^{-5}$	0.845	0.019	$h=2^{-5}$	0.960	0.029
$h=2^{-6}$	0.647	0.058	$h=2^{-6}$	0.844	0.017	$h=2^{-6}$	0.960	0.023

(a) Poisson's equation on quarter annulus

$p=2$	GS	ILUT	$p=3$	GS	ILUT	$p=4$	GS	ILUT
$h=2^{-4}$	0.352	0.043	$h=2^{-4}$	0.703	0.002	$h=2^{-4}$	0.916	0.003
$h=2^{-5}$	0.351	0.037	$h=2^{-5}$	0.699	0.011	$h=2^{-5}$	0.913	0.020
$h=2^{-6}$	0.352	0.042	$h=2^{-6}$	0.699	0.017	$h=2^{-6}$	0.914	0.016

(b) Poisson's equation on unit square

Table 4.1: Spectral radii for Poisson's equation on the quarter annulus and unit square using p-multigrid for different values of the mesh width h and spline degree p.

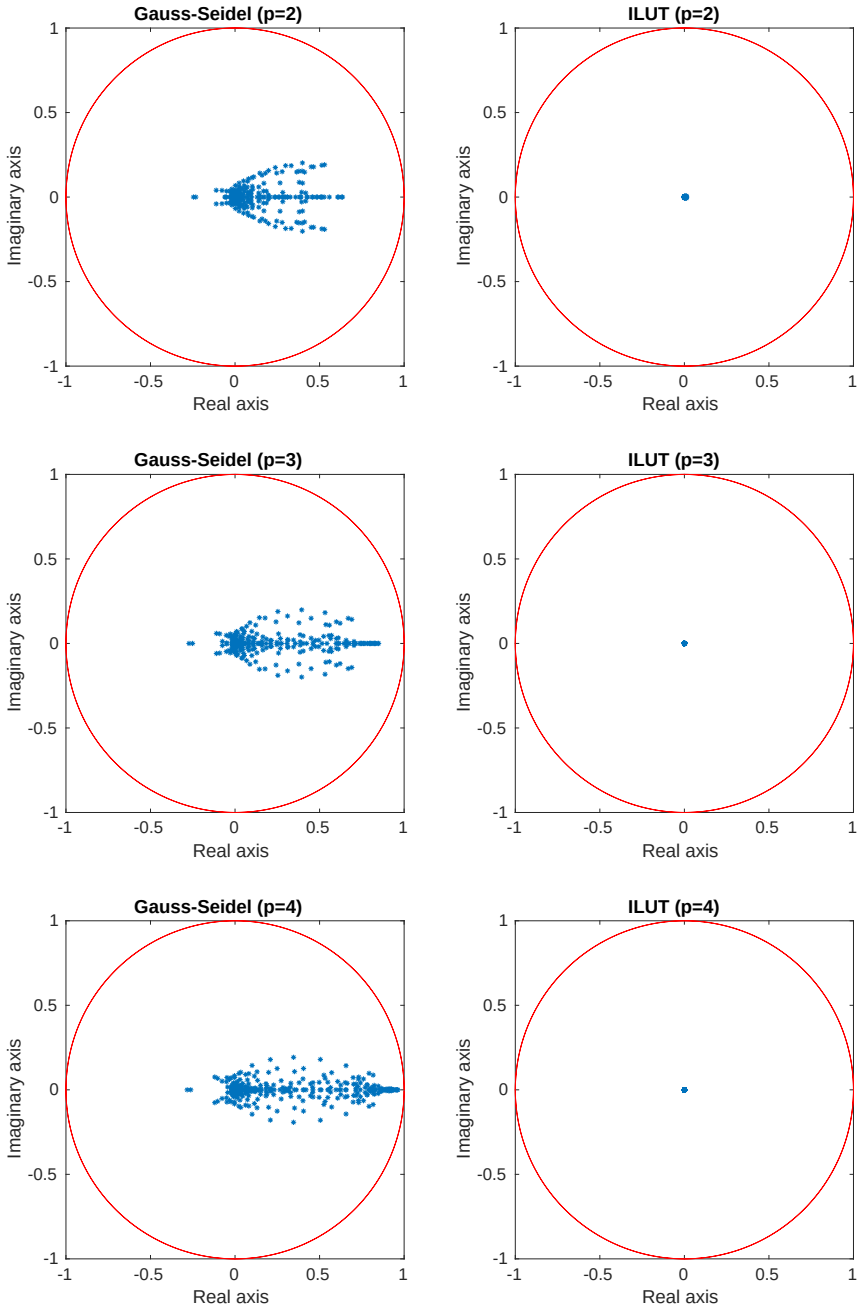


Figure 4.5: Spectra of the iteration matrix for Poisson's equation on the quarter annulus obtained with Gauss-Seidel (left) and ILUT (right).

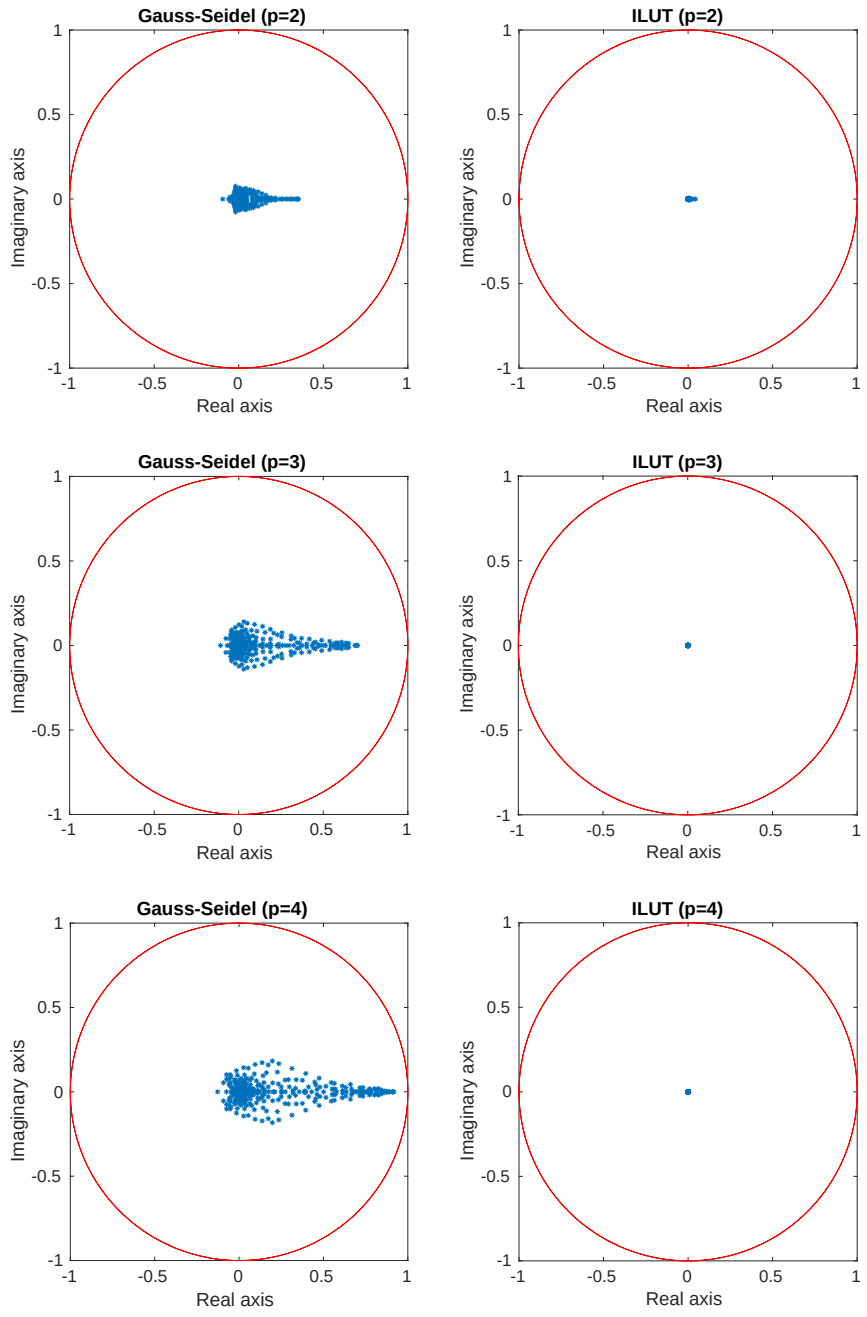


Figure 4.6: Spectra of the iteration matrix for Poisson's equation on the unit square obtained with Gauss-Seidel (left) and ILUT (right).

4.5. NUMERICAL RESULTS

In the previous section, a spectral analysis showed that the use of ILUT as a smoother within a p-multigrid method significantly improves the asymptotic convergence rate compared to the use of Gauss-Seidel as a smoother. In this section, p-multigrid is applied as a stand-alone solver and as a preconditioner within a Bi-CGSTAB method to verify this analysis. Results in terms of iteration numbers and CPU times are obtained using Gauss-Seidel and ILUT as a smoother.

For all numerical experiments, the initial guess $\mathbf{u}_{h,p}^{(0)}$ is chosen randomly, where each entry is sampled from a uniform distribution on the interval $[-1,1]$ using the same seed. Furthermore, we choose $\nu_1 = \nu_2 = 1$ for consistency. Application of multiple smoothing steps, which is in particular common for Gauss-Seidel, decreases the number of iterations until convergence, but does not qualitatively or quantitatively change the p-dependence as long as the number of smoothing steps is kept independent of the spline degree. Furthermore, since the smoother costs dominate when solving the linear systems, CPU times are only mildly affected. The stopping criterion is based on a relative reduction of the initial residual, where a tolerance of $\epsilon = 10^{-8}$ is adopted. Boundary conditions are imposed by eliminating the degrees of freedom associated to the boundary.

p-MULTIGRID AS STAND-ALONE SOLVER

Table 4.2 shows the number of multigrid cycles needed to achieve convergence for the p-multigrid method using different smoothers for the benchmarks considered in this chapter. For the first benchmark, the number of multigrid cycles needed with Gauss-Seidel is in general independent of the mesh width h, but strongly depends on the spline degree p. For the second benchmark, however, the use of Gauss-Seidel leads to a method that diverges, indicated with $(-)$. The p-multigrid method was said to be diverged in case the norm of the relative residual at the end of a V-cycle was significantly higher than at the end of the previous V-cycle. In general, the use of ILUT as a smoother leads to a p-multigrid method which converges for all configurations and exhibits both independence of h and p.

For Poisson's equation on the unit cube, the p-dependence when Gauss-Seidel is adopted as a smoother is stronger compared to the dependence for the two-dimensional benchmarks. Furthermore, the number of iterations slightly decreases for smaller values of the meshwidth h. The number of iterations needed with ILUT as a smoother, is effectively independent of the spline degree p. An h-dependence is visible, however, in particular for higher values of p.

Note that for the first two benchmarks we consider a mesh width up to and including $h = 2^{-9}$, which corresponds to (approximately) a quarter million degrees of freedom. For the three-dimensional problem, a significantly larger mesh width is adopted ($h = 2^{-5}$), leading to around fifty thousand degrees of freedom. The significant higher amount of non-zero entries in the three-dimensional leads, however, to memory issues on the considered platform when a smaller mesh width is considered.

	$p=2$		$p=3$		$p=4$		$p=5$	
	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	61	3	163	3	473
$h=2^{-9}$	5	32	3	61	3	163	3	452

(a) Poisson's equation on quarter annulus

	$p=2$		$p=3$		$p=4$		$p=5$	
	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	–

(b) CDR-equation on unit square

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-2}$	3	65	3	405	3	3255	5	22788
$h=2^{-3}$	3	59	3	339	3	2063	3	8128
$h=2^{-4}$	4	57	3	281	3	1652	5	7152
$h=2^{-5}$	4	55	4	273	6	1361	10	6250

(c) Poisson's equation on the unit cube

Table 4.2: Number of multigrid cycles needed to achieve convergence with p-multigrid using Gauss-Seidel (GS) and ILUT as a smoother. To reproduce these and other tables presenting iteration numbers, we refer to Chapter 7 of this dissertation.

p-MULTIGRID AS A PRECONDITIONER

As an alternative, the p-multigrid method can be applied as a preconditioner within a Bi-CGSTAB method. In the preconditioning phase of each iteration, a single multigrid cycle is applied. Numerical results can be found in Table 4.3. When applying Gauss-Seidel as a smoother, the number of iterations needed with Bi-CGSTAB is significantly lower compared to the number of p-multigrid multigrid cycles and even restores stability for the second bechmark (see Table 4.2). However, a dependence of the iteration numbers on p is still present. When adopting ILUT as a smoother, the number of iterations needed for convergence slightly decreases compared to the number of p-multigrid multigrid cycles for all configurations and benchmarks. Furthermore, the number of iterations is independent of both h and p.

CPU TIMES

Besides iteration numbers, computational times have been determined when adopting p-multigrid as a stand-alone solver. A serial implementation in the C++ library G+Smo [101] is considered on an Intel(R) Core(TM) i7-8650U CPU (1.90GHz). Figure 4.7 illustrates the CPU times obtained for the p-multigrid method for the first benchmark adopting both Gauss-Seidel and ILUT as a smoother. The detailed CPU timings can be found as well in Appendix C. The assembly times denote the CPU time needed to assemble all

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	2	13	2	18	2	41	2	78
$h=2^{-7}$	2	12	2	20	2	41	2	92
$h=2^{-8}$	3	13	2	19	2	43	2	95
$h=2^{-9}$	3	13	2	21	2	41	2	95

(a) Poisson's equation on quarter annulus

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	2	7	2	13	2	29	2	65
$h=2^{-7}$	2	8	2	13	2	29	2	70
$h=2^{-8}$	2	7	2	12	2	29	2	64
$h=2^{-9}$	2	7	2	14	2	28	2	72

(b) CDR-equation on unit square

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-2}$	2	14	2	30	2	94	3	276
$h=2^{-3}$	2	16	2	40	2	105	2	229
$h=2^{-4}$	2	19	2	44	2	119	3	285
$h=2^{-5}$	2	19	2	49	3	136	3	310

(c) Poisson's equation on the unit cube

Table 4.3: Number of iterations needed to achieve convergence with Bi-CGSTAB, using p-multigrid as preconditioner.

operators, including the prolongation and restriction operators, which are independent of the smoother. The setup costs of the smoother, however, differ significantly for both smoothers: Gauss-Seidel does not require any setup costs, while the setup costs with ILUT as a smoother increase significantly for higher values of p. When adopting Gauss-Seidel as a smoother, the time needed to solve the linear systems is significantly higher compared to the use of ILUT. However, since the factorizations costs are relatively high, the p-multigrid method using ILUT as a smoother is faster for only a limited amount of configurations. It should be noted that, if multiple solves are necessary using the same system matrix, the setup costs become relatively seen less important compared to the solver costs, as can be the case with time-dependent problems.

Remark: For all numerical experiments, the 'coarse grid' operators of the multigrid hierarchy have been obtained by rediscretizing the bilinear form. Alternatively, all operators of the h-multigrid hierarchy at level $p = 1$ could be obtained by applying the Galerkin projection. Furthermore, alternative (and more efficient) assembly strategies exist (see Section 2.6). Therefore, the assembly, smoother setup and solving costs are presented separately in this section.

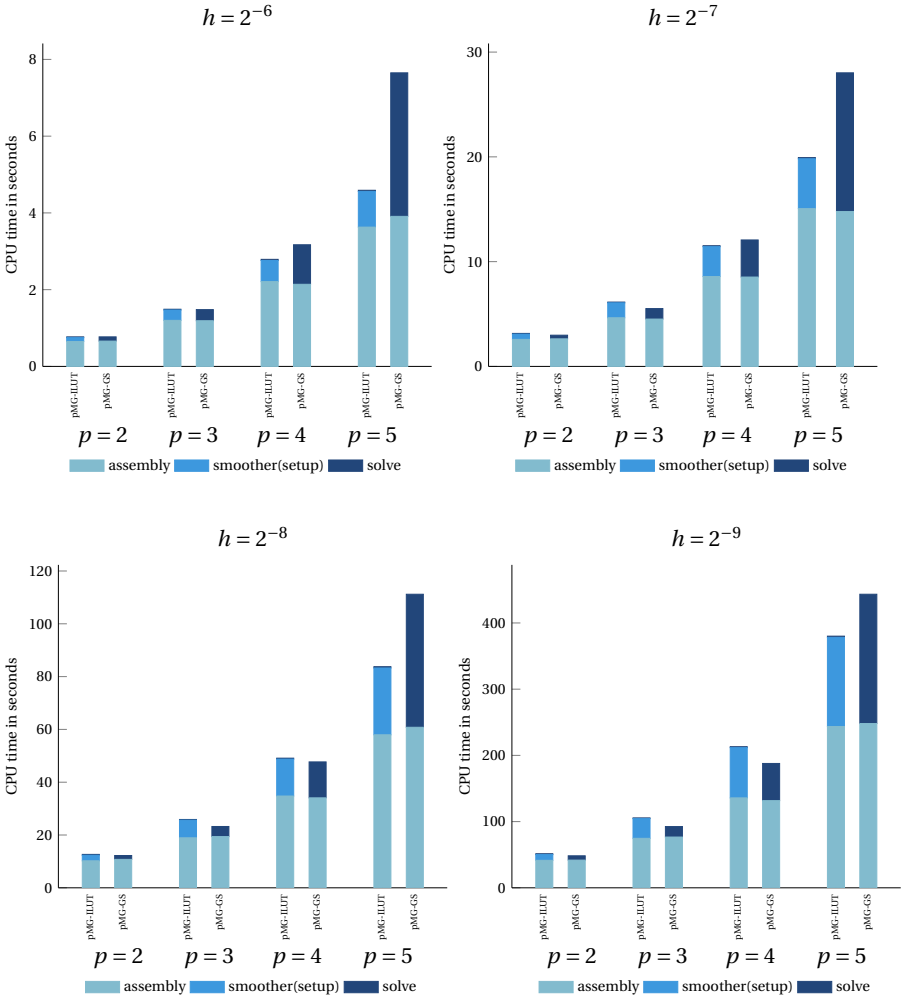


Figure 4.7: CPU timings for p-multigrid (pMG) adopting ILUT and Gauss-Seidel (GS) as a smoother for different values of h for the first benchmark.

4.6. COMPARISON TO h-MULTIGRID METHODS

Throughout this dissertation, the use of ILUT and Gauss-Seidel as a smoother has been investigated within a p-multigrid method. In this section, both smoothers will be applied within h-multigrid methods and compared to the results obtained in the previous section to investigate the effect of the different coarsening strategies. This section starts with a spectral analysis in case h-multigrid is adopted as coarsening strategy. Then, both iteration numbers and CPU times are determined for the h-multigrid method adopting Gauss-Seidel and ILUT as a smoother.

SPECTRAL ANALYSIS

The error reduction factors for the first benchmark for both smoothers and coarsening strategies are shown in Figure 4.8 for $h = 2^{-5}$ and different values of p. The reduction factors obtained with both smoothers, which are independent of the coarsening strategy, are shown in the left column and were obtained in the previous section as well. The plots in the right column show the reduction factors for both coarsening strategies. The coarse grid correction obtained by coarsening in h (e.g. CGC_h) is more effective compared to a correction obtained by coarsening in p. Note that, for higher values of p, both types of coarse grid correction remain effective in reducing the coefficients of the eigenvector expansion corresponding to the low-frequency modes. The oscillatory behaviour of the reduction rates in Figure 4.8 has been observed in the literature for a similar benchmark, see [79], although the exact cause is unknown and requires further investigation. Figure 4.9 shows the error reduction factors obtained for the second benchmark, showing similar, but less oscillatory, behaviour. Note that, the reduction rates of the smoother itself are uniformly smaller than 1 for both benchmarks, implying the smoother is a solver as well.

Figure 4.10 shows the spectra for the first benchmark for $h = 2^{-4}$ and various values of p obtained with both coarsening strategies using Gauss-Seidel and ILUT as a smoother. For reference, the unit circle has been added in all plots. Clearly, the spectral radius significantly increases for higher values of p when adopting Gauss-Seidel as a smoother for h-multigrid methods as well, while the use of ILUT as a smoother results in spectra clustered around the origin, implying fast convergence of the resulting multigrid method.

The spectra of the iteration matrices for the second benchmark are presented in Figure 4.11. Although the eigenvalues are more clustered with Gauss-Seidel compared to the first benchmark, the same behaviour can be observed.

The spectral radii for both benchmarks, where $\nu_1 = \nu_2 = 1$, are presented in Table 4.4. Here, the results for both the p-multigrid method and h-multigrid method are presented for comparison. For Gauss-Seidel, the spectral radius of the iteration matrix is independent of the mesh width h and coarsening strategy, but depends strongly on the spline degree p. The use of ILUT leads to a spectral radius which is significantly lower for all values of h and p. Although ILUT exhibits a small h-dependence, the spectral radius remains low for all values of h and both coarsening strategies. As a consequence, the p-multigrid and h-multigrid method are expected to show essentially both h- and p-independent convergence behaviour when ILUT is adopted as a smoother.

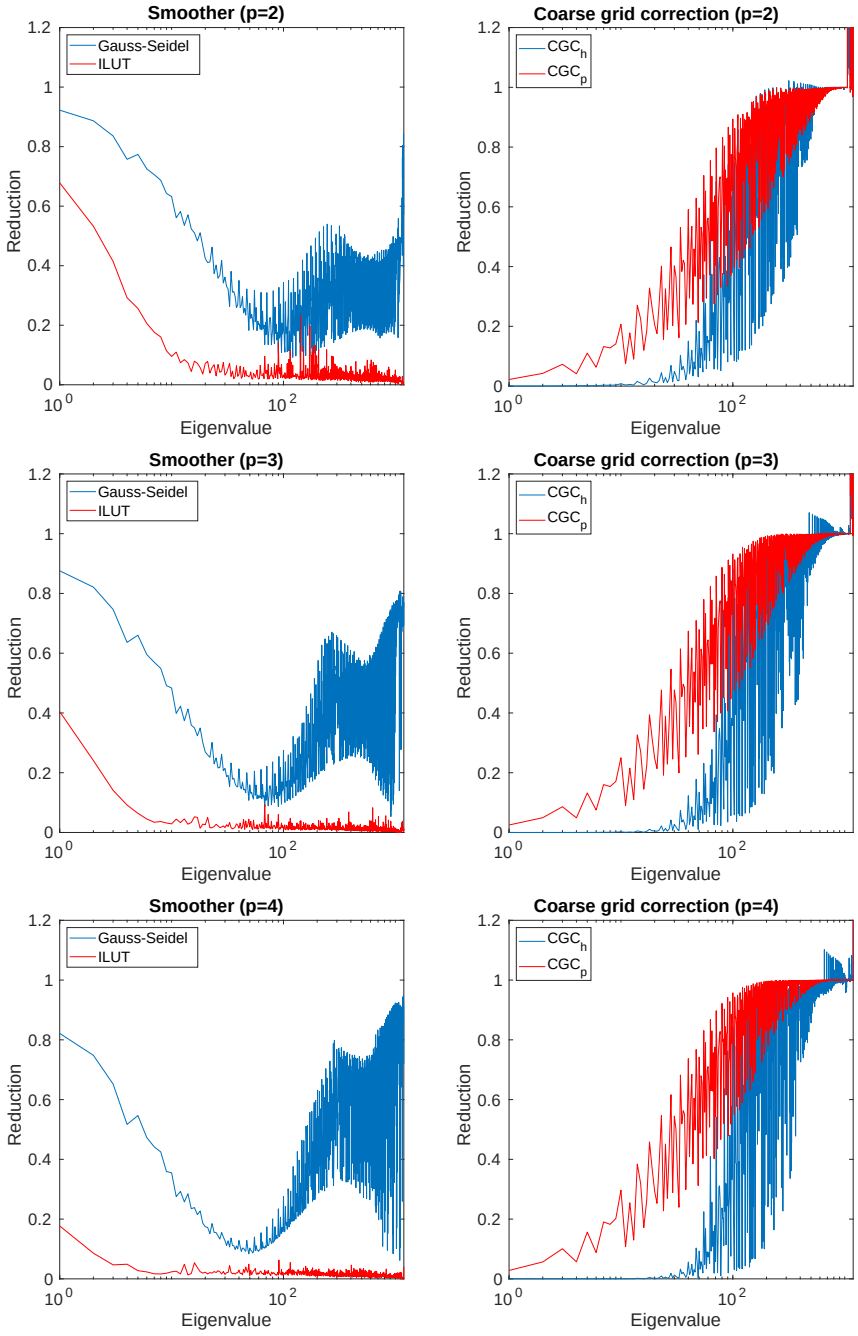


Figure 4.8: Error reduction in $(\mathbf{v}_j)$ for Poisson's equation on the quarter annulus with $p = 2,3,4$ and $h = 2^{-5}$ obtained with different smoothers (left) and coarsening strategies (right).

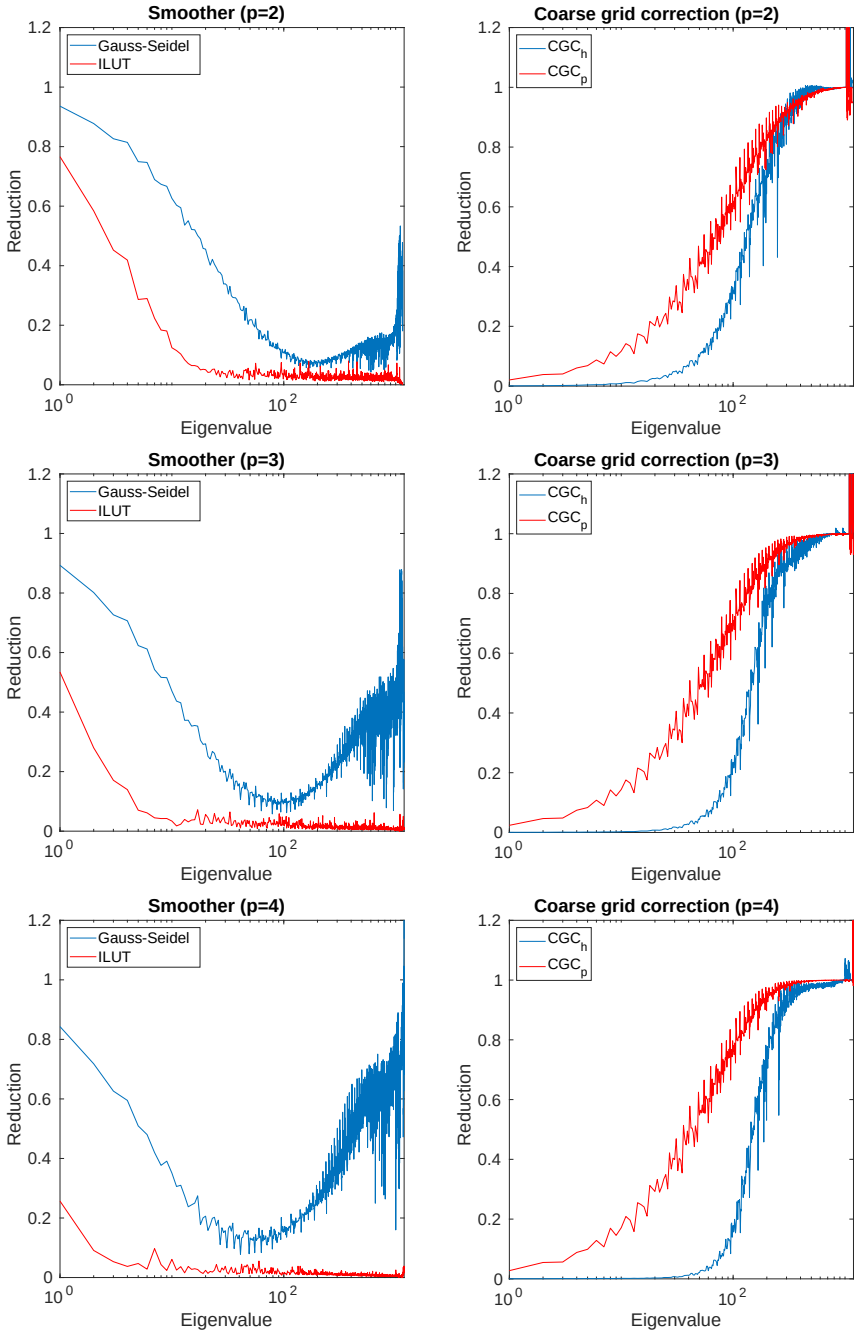


Figure 4.9: Error reduction in $(\mathbf{v}_j)$ for Poisson's equation on the unit square with $p = 2,3,4$ and $h = 2^{-5}$ obtained with different smoothers (left) and coarsening strategies (right).

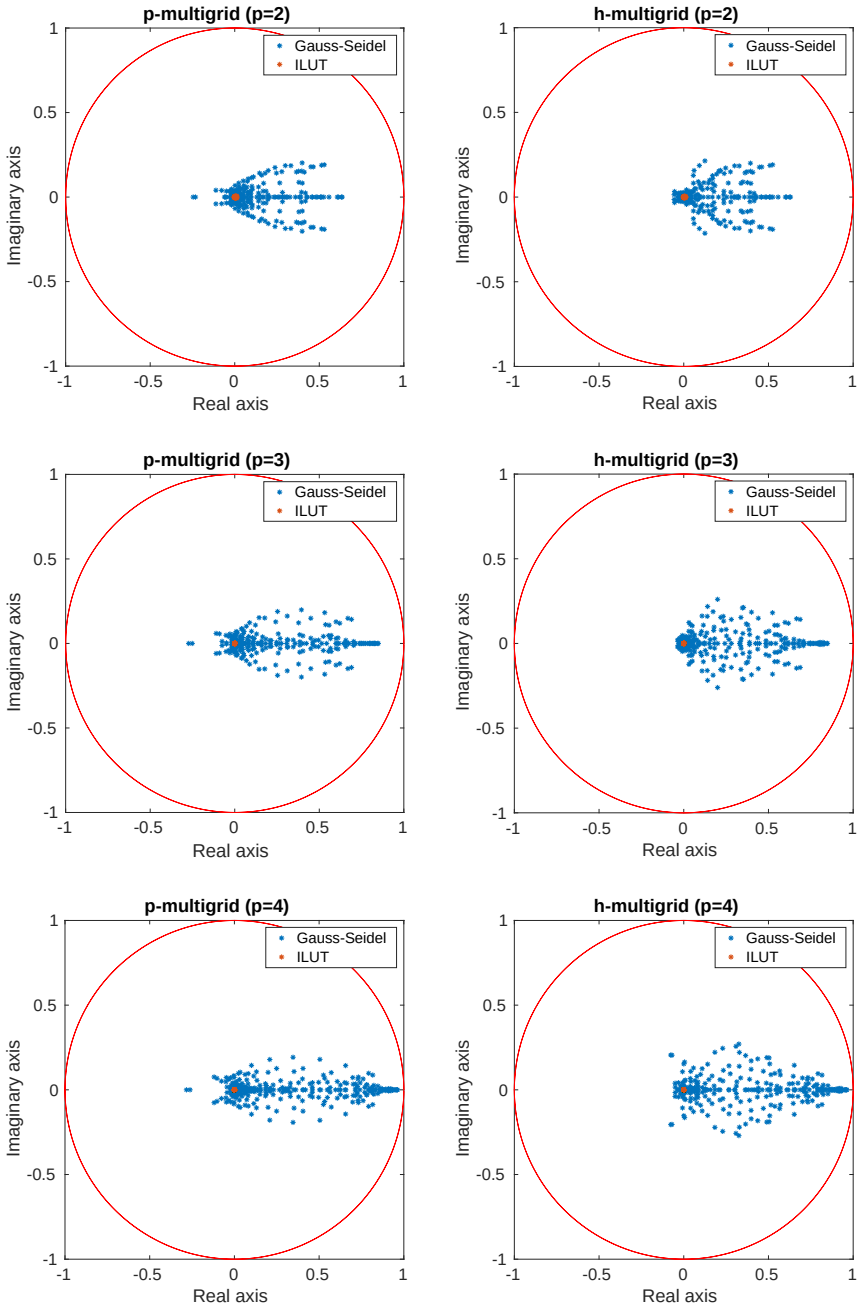


Figure 4.10: Spectra of the iteration matrix for Poisson's equation on the quarter annulus obtained with p-multigrid (left) and h-multigrid (right).

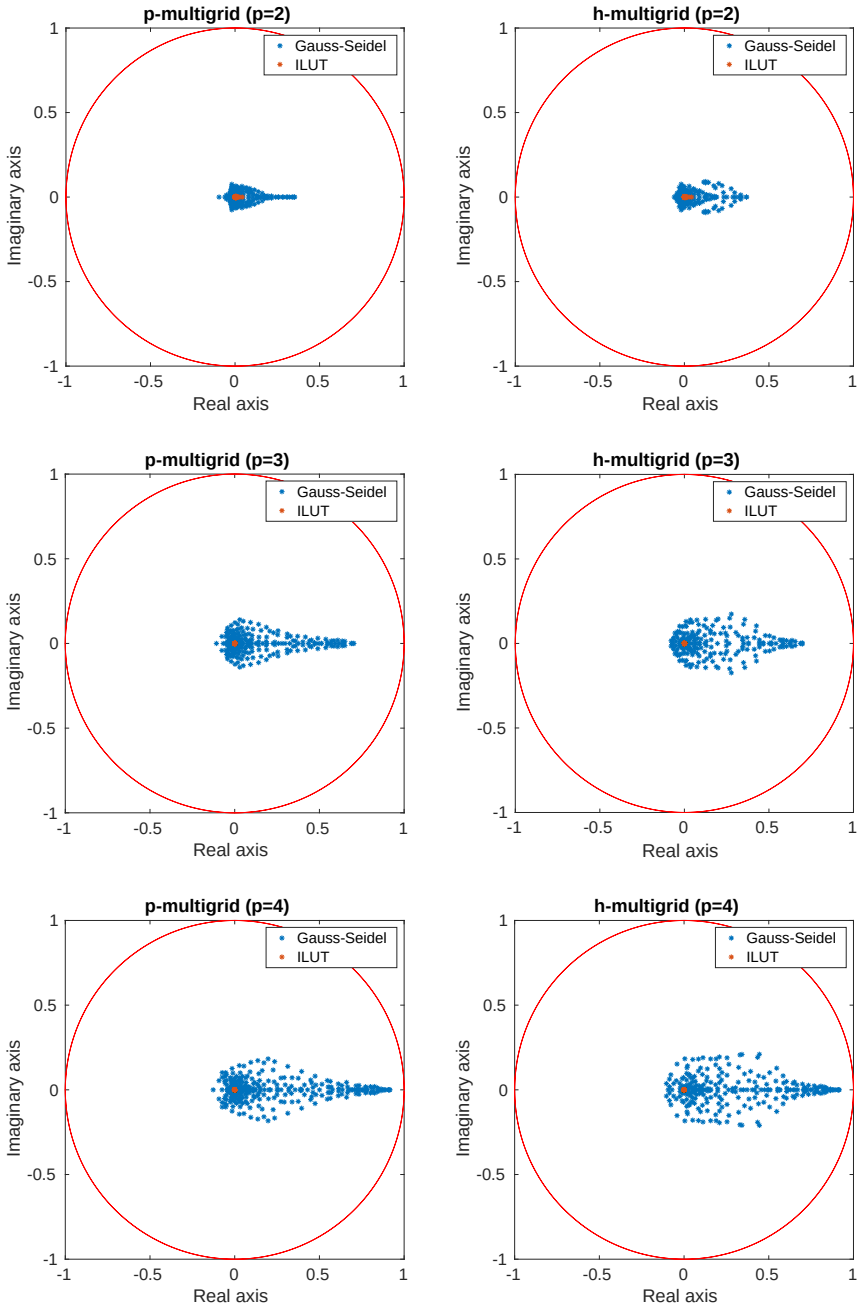


Figure 4.11: Spectra of the iteration matrix for Poisson's equation on the unit square obtained with p-multigrid (left) and h-multigrid (right).

$p=2$	GS	ILUT	$p=3$	GS	ILUT	$p=4$	GS	ILUT
$h=2^{-4}$	0.635	0.014	$h=2^{-4}$	0.849	0.003	$h=2^{-4}$	0.963	0.003
$h=2^{-5}$	0.631	0.039	$h=2^{-5}$	0.845	0.019	$h=2^{-5}$	0.960	0.029
$h=2^{-6}$	0.647	0.058	$h=2^{-6}$	0.844	0.017	$h=2^{-6}$	0.960	0.023

(a) p-multigrid, Poisson's equation on quarter annulus

$p=2$	GS	ILUT	$p=3$	GS	ILUT	$p=4$	GS	ILUT
$h=2^{-4}$	0.630	0.012	$h=2^{-4}$	0.848	0.004	$h=2^{-4}$	0.963	0.003
$h=2^{-5}$	0.627	0.039	$h=2^{-5}$	0.845	0.018	$h=2^{-5}$	0.960	0.029
$h=2^{-6}$	0.646	0.059	$h=2^{-6}$	0.844	0.014	$h=2^{-6}$	0.960	0.023

(b) h-multigrid, Poisson's equation on quarter annulus

$p=2$	GS	ILUT	$p=3$	GS	ILUT	$p=4$	GS	ILUT
$h=2^{-4}$	0.352	0.043	$h=2^{-4}$	0.703	0.002	$h=2^{-4}$	0.916	0.003
$h=2^{-5}$	0.351	0.037	$h=2^{-5}$	0.699	0.011	$h=2^{-5}$	0.913	0.020
$h=2^{-6}$	0.352	0.042	$h=2^{-6}$	0.699	0.017	$h=2^{-6}$	0.914	0.016

(c) p-multigrid, CDR-equation on unit square

$p=2$	GS	ILUT	$p=3$	GS	ILUT	$p=4$	GS	ILUT
$h=2^{-4}$	0.367	0.043	$h=2^{-4}$	0.698	0.002	$h=2^{-4}$	0.916	0.003
$h=2^{-5}$	0.367	0.036	$h=2^{-5}$	0.696	0.008	$h=2^{-5}$	0.913	0.020
$h=2^{-6}$	0.359	0.042	$h=2^{-6}$	0.698	0.006	$h=2^{-6}$	0.913	0.016

(d) h-multigrid, CDR-equation on unit square

Table 4.4: Spectral radii for the first and second benchmark using p-multigrid and h-multigrid for different values of the mesh width h and spline degree p.

ITERATION NUMBERS

Table 4.5 shows the number of iterations needed to reach convergence with an h-multigrid method for the three considered benchmark problems and can be compared to Table 4.2. The number of multigrid cycles needed with Gauss-Seidel is in general independent of the mesh width h, but strongly depends on the spline degree p. The use of ILUT as a smoother leads to an h-multigrid method which converges for all configurations and exhibits both independence of h and p. Furthermore, the number of iterations needed for convergence is significantly lower. Compared to the use of p-multigrid as a method, the results are very similar.

Table 4.6 shows the number of iterations when h-multigrid is applied as a preconditioner. Note that, since the h-multigrid method is symmetric, a Conjugate Gradient (CG) method can be applied as well. In general, a single iteration performed with a Bi-CGSTAB method is twice as expensive compared to a single CG iteration. Results with a CG method have been added between brackets.

With a Bi-CGSTAB method, results obtained with the h-multigrid method are very similar to the use of p-multigrid as a preconditioner. Note that, the use of CG as outer Krylov solver, approximately doubles the number of iterations when ILUT is applied as a smoother. For Gauss-Seidel, the use of Bi-CGSTAB yields significant lower iteration numbers compared to the use of CG, but this does not fully compensate for the higher costs per iteration associated to the Bi-CGSTAB method.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	4	30	3	62	3	175	3	492
$h=2^{-7}$	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	164	3	452

(a) Poisson's equation on quarter annulus

	$p=2$		$p=3$		$p=4$		$p=5$	
	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	–	3	–	3	–	4	–

(b) CDR-equation on unit square

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-2}$	3	65	3	405	3	2269	5	22785
$h=2^{-3}$	3	59	3	339	3	2065	3	8135
$h=2^{-4}$	3	57	3	281	3	1652	5	7148
$h=2^{-5}$	4	55	3	273	5	1361	10	6248

(c) Poisson's equation on the unit cube

Table 4.5: Number of multigrid cycles needed to achieve convergence with h-multigrid.

CPU TIMES

CPU times have been obtained as well in case h-multigrid methods are considered. Figure 4.12 shows the CPU time needed to assemble all operators, including the prolongation and restriction operators, which can be found as well in Appendix D. Compared to p-multigrid methods, less operators have to be assembled. However, most of the operators in the p-multigrid method are assembled at level $p = 1$, where the number of nonzero entries is significantly lower compared to the matrices resulting from high-order discretizations. As a consequence, the total assembly costs are higher with h-multigrid methods for higher values of p. It should be noted, however, that in case of nested spaces, the coarse matrices can be obtained by a Galerkin projection as well. This reduces the assembly costs for the h-multigrid method (and to a lesser extent for the p-multigrid method), making the assembly costs of both methods similar for the considered configurations.

With respect to the setup costs for the smoother, similar observation can be made: For higher values of p, the ILUT factorization costs are significantly higher for the h-multigrid method, due to the increasing number of nonzero entries. The time needed to solve linear systems is slightly lower for h-multigrid methods, since the costs of a single cycle are lower compared to p-multigrid methods. Note that, as with p-multigrid methods, adopting Gauss-Seidel as a smoother leads to significantly higher times needed to solve the linear systems.

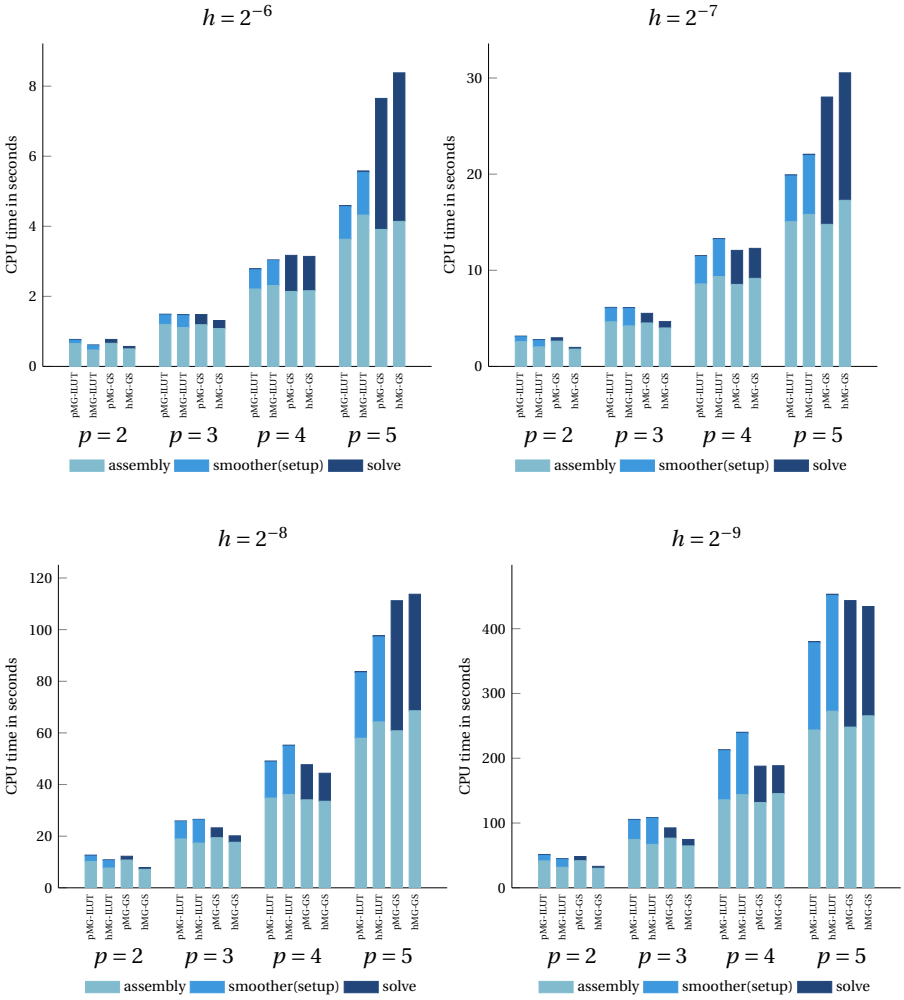


Figure 4.12: CPU timings for p-multigrid (pMG) and h-multigrid (hMG) adopting ILUT and Gauss-Seidel (GS) as a smoother for different values of h for Poisson's equation on the quarter annulus.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	2(4)	8(15)	2(3)	15(23)	2(3)	24(42)	2(4)	43(80)
$h=2^{-7}$	2(4)	9(15)	2(3)	15(23)	2(3)	24(42)	2(4)	47(80)
$h=2^{-8}$	3(5)	9(16)	2(3)	14(23)	2(3)	25(41)	2(4)	41(78)
$h=2^{-9}$	3(5)	10(16)	2(3)	14(23)	2(3)	25(41)	2(4)	43(79)

(a) Poisson's equation on quarter annulus

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	2(4)	6(11)	2(3)	12(18)	2(3)	22(35)	2(4)	36(68)
$h=2^{-7}$	2(4)	7(11)	2(3)	11(18)	2(4)	22(33)	2(4)	40(66)
$h=2^{-8}$	2(4)	7(11)	2(3)	11(18)	2(4)	21(34)	2(4)	39(64)
$h=2^{-9}$	2(4)	6(11)	2(3)	11(18)	2(4)	22(34)	3(4)	40(67)

(b) CDR-equation on unit square

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-2}$	2(3)	16(24)	2(3)	37(58)	2(3)	120(158)	3(7)	590(459)
$h=2^{-3}$	2(3)	16(25)	2(3)	48(63)	2(3)	103(168)	2(3)	221(490)
$h=2^{-4}$	2(3)	17(25)	2(3)	43(65)	2(4)	104(188)	3(8)	286(543)
$h=2^{-5}$	2(3)	17(25)	2(3)	44(67)	3(5)	131(197)	3(12)	264(591)

(c) Poisson's equation on the unit cube

Table 4.6: Number of iterations needed to achieve convergence with Bi-CGSTAB (CG), using h-multigrid as a preconditioner.

4.7. COMPARISON TO ALTERNATIVE SMOOTHER

As mentioned in Chapter 3, alternative smoothers have been developed in recent years for h-multigrid methods in the context of Isogeometric Analysis. An example of such a smoother is a smoother based on stable splittings of spline spaces [82], known as the subspace corrected mass smoother (SCMS). In this section, we compare a p-multigrid and h-multigrid method using this smoother. For this comparison, we consider the CDR-equation on the unit square with:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad R = 1. \tag{4.18}$$

Homogeneous Neumann boundary conditions are applied and the right-hand side is given by:

$$f(x,y) = 2\pi^2 \sin(\pi(x+\frac{1}{2}))\sin(\pi(y+\frac{1}{2})).$$

Table 4.7 shows the number of iterations needed to reach convergence with the p-multigrid method and the h-multigrid method for the ILUT smoother and the subspace corrected mass smoother. For the ILUT smoother, iteration numbers are independent of h and p for both coarsening strategies. The smoother from [82] shows iteration numbers independent of h and p within an h-multigrid method. A slight p-dependency is visible

when this smoother is applied within a p-multigrid method. With the ILUT smoother, the number of iterations needed to reach convergence is significantly lower for all configurations.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS
$h=2^{-6}$	5	40	5	44	5	47	5	52
$h=2^{-7}$	5	40	5	44	4	48	5	53
$h=2^{-8}$	5	40	4	44	5	48	4	53
$h=2^{-9}$	5	40	4	45	5	48	4	53

(a) p-multigrid

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS
$h=2^{-6}$	4	48	4	48	4	48	4	48
$h=2^{-7}$	4	49	3	50	4	49	4	49
$h=2^{-8}$	4	49	3	50	5	50	4	49
$h=2^{-9}$	4	49	3	50	5	50	4	50

(b) h-multigrid

Table 4.7: Number of iterations with ILUT and the smoother from [82] using p-multigrid and h-multigrid.

CPU times for assembly, setting up the smoother and solving the linear system once are presented in Figure 4.13 (left) and can be found in Appendix E and F. Again, a serial implementation in the C++ library G+Smo [101] is considered on an Intel(R) Core(TM) i7-8650U CPU (1.90GHz). The time needed to assemble the operators is comparable for the p-multigrid and h-multigrid method. However, setting up the ILUT smoother is significantly more expensive compared to the smoother from [82]. On the other hand, the CPU time needed to solve the problem is lower when adopting the ILUT smoother. The total solver costs are lower for all configurations when adopting the subspace corrected mass smoother.
However, in case multiple solves are necessary with the same system matrix and different right-hand sides, the ILUT smoother becomes relatively seen more efficient. This is for example the case when 'snapshot' solutions are required to apply Proper Orthogonal Decomposition [102]. Figure 4.13 (right) shows that, already when solving the linear system for 10 different right-hand sides, the CPU times with ILUT as a smoother within the p-multigrid method are lower for all values of p. For the h-multigrid method, however, the use of the smoother from [82] remains more efficient for high values of p.

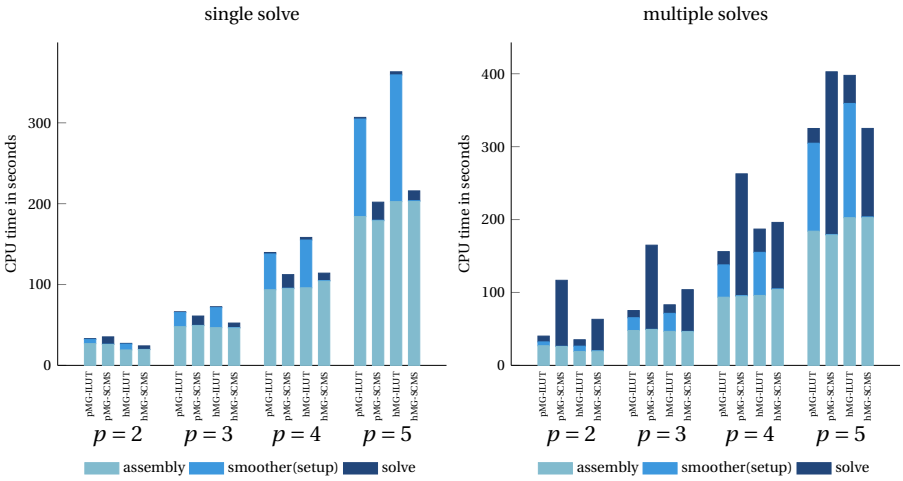


Figure 4.13: CPU timings for p-multigrid and h-multigrid adopting ILUT and the smoother from [82], respectively, for a single solve (left) and 10 solves (right).

4.8. TRUNCATED HIERARCHICAL B-SPLINES (THB-SPLINES)

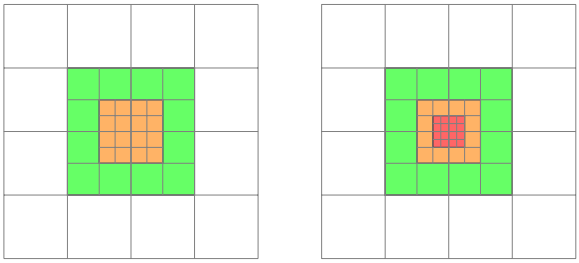
Finally, to illustrate the versatility of the proposed p-multigrid method, we consider discretizations obtained with THB-splines [103]. THB-splines are the result of a local refinement strategy, in which a subset of the basis functions on the fine level are truncated. As a result, not only linear independence and non-negativity are preserved (as with HB-splines, see [104, 105]), but also the partition of unity property.

In the literature, the use of multigrid methods for THB-spline discretizations is an ongoing topic of research [106–108]. We consider Poisson's equation on the unit square, where the exact solution is the same as for the second benchmark. Starting from a tensor product B-spline basis with meshwidth h and degree p, two and three levels of refinement are added as shown in Figure 4.14, leading to a THB-spline basis consisting of, respectively, three and four levels. Figure 4.15 shows the sparsity pattern of the stiffness matrix and the ILUT factorization for $p = 4$ and $h = 2^{-5}$ for configuration (b) (see Figure 4.14). Compared to the (standard) tensor-product B-spline basis the bandwidth of the stiffness matrix significantly increases.

Table 6.8 shows the results obtained with the p-multigrid method (as described in section 4.3) applied as a stand-alone solver, where a stopping criterion is adopted of $\epsilon = 10^{-8}$. The number of iterations needed with p-multigrid (and ILUT as a smoother) depends only mildly on p. Furthermore, the number of iterations are significantly lower compared to the use of Gauss-Seidel as a smoother.

For the configurations denoted in bold, a fillfactor of 2 was adopted, to prevent the p-multigrid from diverging. Figure 4.16 illustrates the reason for it in the case $p = 4$ and $h = 2^{-4}$ for configuration (a). A fillfactor of 1 does not reduce the norm of the (generalized) eigenvectors, while a fillfactor of 2 reduces the eigenvectors over the entire spectrum. In general, a higher fillfactor was necessary for only a limited amount of configurations.

For all numerical experiments, smoothing is performed globally at each level of the multi-



(a) (b)

Figure 4.14: Two hierarchical mesh adopted for THB-Spline basis with the second (green), third (orange) and fourth (red) refinement levels coloured.

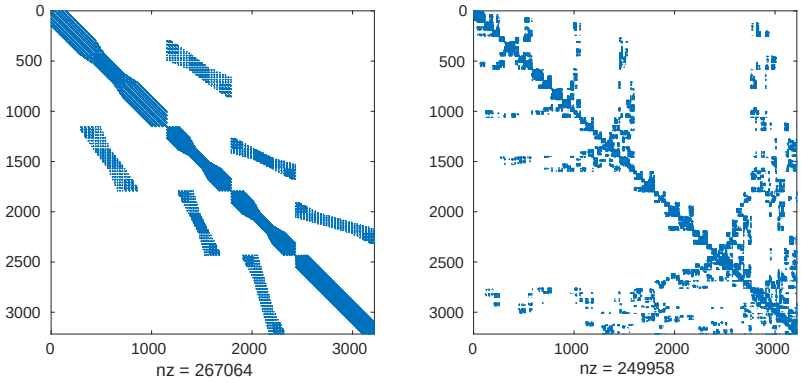


Figure 4.15: Sparsity pattern of the stiffness matrix $\mathbf{A}_{h,4}$ (left) and $\mathbf{L}_{h,4}+\mathbf{U}_{h,4}$ (right).

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-4}$	5	16	6	45	5	178	5	713
$h=2^{-5}$	5	17	6	40	7	182	**5**	882
$h=2^{-6}$	5	17	5	41	7	189	**11**	936

(a) THB-spline basis with three levels of refinement

	$p=2$		$p=3$		$p=4$		$p=5$	
	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

(b) THB-spline basis with four levels of refinement

Table 4.8: Number of multigrid cycles needed for different THB-spline discretizations.

grid hierarchy. Alternatively, local smoothing can be adopted to ensure optimal order of the complexity. In [108] it was shown that adopting Gauss-Seidel as a smoother results in h-independent convergence for h-multigrid methods when applied to both HB-spline and THB-spline discretizations. Results presented in this section should be considered as a first step towards the use of p-multigrid methods for THB-spline discretizations. Future research should focus on more efficient applications of p-multigrid solvers for THB-spline discretizations, considering in particular local smoothing.

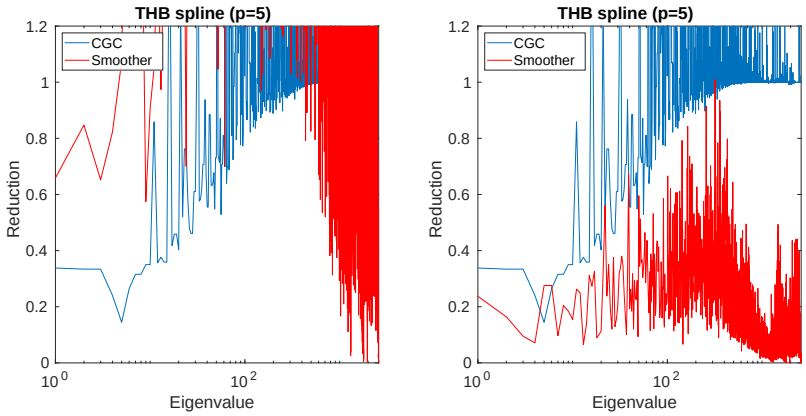


Figure 4.16: Reduction factors obtained for fillfactor 1 (left) and 2 (right).

4.9. CONCLUDING REMARKS

In this chapter, we presented a p-multigrid method to solve linear systems of equations arising in Isogeometric Analysis for single patch geometries. The effectiveness of different smoothers (e.g. Gauss-Seidel and ILUT) have been investigated by means of a spectral analysis. In general, the smoother and coarse grid correction showed to be complementary to each other. The coarse grid correction reduces the coefficients associated with the low-frequency components, while the smoother reduces the high-frequency components. For higher values of p, however, the Gauss-Seidel smoother becomes less and less effective, leading to asymptotic convergence rates which strongly depend on p. The use of ILUT as a smoother, leads to convergence rates which are essentially independent of p.

Numerical results, obtained for the CDR-equation on the unit square and Poisson's equation on a quarter annulus and unit cube, confirm this analysis. Here, p-multigrid has been applied as well as a preconditioner within a Bi-CGSTAB method. Compared to h-multigrid methods, similar iteration numbers were obtained for the considered benchmarks. In terms of CPU times, p-multigrid showed to be competitive with h-multigrid methods, when Gauss-Seidel and ILUT were adopted as a smoother. In fact, p-multigrid with ILUT as a smoother is even competitive to an h-multigrid method adopting a state-of-the-art smoother when multiple solves are required.

5

P-MULTIGRID METHODS FOR MULTIPATCH GEOMETRIES

In this chapter we apply the p-multigrid method as presented in Chapter 4 to benchmarks involving multipatch geometries. In particular, we consider the use of Krylov methods to accelerate the p-multigrid method if the number of patches is increased. Finally, a block ILUT smoother is presented which is specifically designed for multipatch geometries and is suited for parallel computing within an HPC framework.

Parts of this chapter have been published in:
R.Tielen, M. Möller and C.Vuik, *Efficient p-Multigrid based solvers for multipatch geometries in Isogeometric Analysis*, Lecture Notes in Computational Science and Engineering, Springer, **133** (2020)

Parts of this chapter have been published in:
R.Tielen, M. Möller and C.Vuik, *A block ILUT smoother for multipatch geometries in Isogeometric Analysis*, In: Springer INdAM Series, Springer (2021)

5.1. INTRODUCTION

In Chapter 4, we applied the proposed p-multigrid method to a variety of two- and three-dimensional benchmarks on single patch geometries. In general, the use of ILUT as a smoother leads to iteration numbers which are independent of both the knot span size h and spline degree p. In this chapter, we apply the p-multigrid method on benchmarks involving multipatch geometries. Multipatch geometries are adopted when the geometry cannot be described by a single mapping, which is typically the case for more practical applications.

The benchmarks considered throughout this chapter are presented in Section 5.2. In Section 5.3, we apply the p-multigrid method as described in Chapter 4 directly on these benchmarks and investigate the use of a Krylov outer solver to accelerate the overall convergence of the p-multigrid method as well. Section 5.4 then introduces a block ILUT smoother [1] specifically designed for multipatch geometries and presents numerical results obtained with this smoother. In Section 5.5, a p-multigrid method adopting this block ILUT smoother is applied to a more challening benchmark (i.e, a Yeti footprint). This chapter ends with some concluding remarks.

5

5.2. BENCHMARKS

Throughout this chapter, we will consider the following benchmarks to assess the quality of the considered p-multigrid methods:

Benchmark 1. Let Ω be the unit square, i.e., $\Omega = [0,1]^2$. Poisson's equation is considered, where the exact solution u is given by

$$u(x,y) = \sin(\pi x)\sin(\pi y).$$

Benchmark 2. Let Ω be the quarter annulus with an inner and outer radius of 1 and 2, respectively. Again, Poisson's equation is considered, where the exact solution u is given by

$$u(x,y) = -(x^2+y^2-1)(x^2+y^2-4)xy^2.$$

Benchmark 3. Let $\Omega = \{[-1,1]\times[-1,1]\}\backslash\{[0,1]\times[0,1]\}$ be an L-shaped domain. As with the other benchmarks, Poisson's equation is considered, where the exact solution u is given by

$$u(x,y) = \begin{cases} \sqrt[3]{x^2+y^2}\sin\left(\frac{2\text{atan2}(y,x)-\pi}{3}\right) & \text{if } y>0, \\ \sqrt[3]{x^2+y^2}\sin\left(\frac{2\text{atan2}(y,x)+3\pi}{3}\right) & \text{if } y<0, \end{cases}$$

where atan2 is the 2-argument arctangent function. The right-hand side is chosen according to the exact solution. For the first two benchmarks, homogeneous Dirichlet boundary conditions are applied on the entire boundary $\partial\Omega$, while for the third benchmark inhomogeneous Dirichlet boundary conditions are applied. Note that the geometry of each benchmark can be described by a single patch. The multipatch geometries

[1] To make a clear distinction, we will refer to the ILUT smoother discussed in Chapter 4 as global ILUT smoother throughout this chapter.

considered throughout this chapter are obtained by splitting the single patch uniformly in both directions.

5.3. GLOBAL ILUT

Figure 5.1 illustrates a multipatch geometry consisting of 4 patches $\Omega_{(k)}$, $k = 1,\dots,4$, and the resulting block structure of the system matrix $\mathbf{A}_{h,p}$. The first 4 diagonal blocks are associated with the interior degrees of freedom on Ω_i, $i = 1,\dots,4$, while the right-bottom block, shown in grey, denotes the degrees of freedom at the interface Γ. Finally, the off-diagonal blocks denote the coupling between degrees of freedom at the interior and the interface, resulting in an arrowhead matrix. It should be noted that this structure is related to a particular numbering of the degrees of freedom. Note that this block structure is common within domain decomposition (DD) methods [109], which decompose the global domain Ω into subproblems and solve small boundary value problems per subdomain individually using data at the interface as boundary conditions. A coarse problem is then typically solved to exchange information about the solution between the subdomains. When non-overlapping subdomains are considered, the resulting block structure is similar to the structure shown in Figure 5.1.

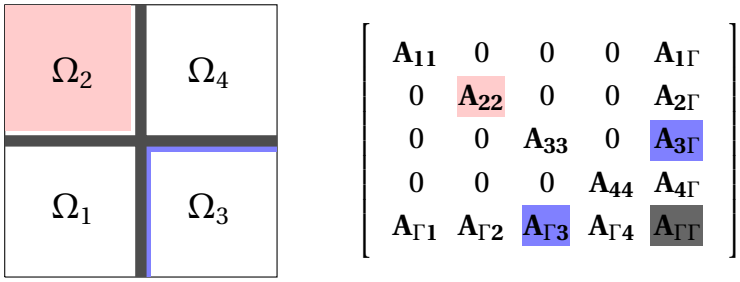


Figure 5.1: A multipatch geometry, consisting of 4 patches and the resulting block structure of the system matrix. The block associated to the interior DOF of Ω_2 is highlighted in red, while the coupling between the interface Γ and Ω_3 is highlighted in blue. Furthermore, the DOF at Γ result in the gray block.

As a first step towards p-multigrid methods for multipatch geometries, we apply the p-multigrid method from Chapter 4 to the different benchmarks. Table 5.1 shows the number of p-multigrid cycles needed to reach convergence with the global ILUT smoother. For some configurations, the application of the global ILUT smoother results in an diverging p-multigrid method (indicated by '$-$'). In general, the iteration numbers are (more or less) independent of h and p. However, a dependency on the number of patches can be observed, in particular for coarser meshes.

To further investigate this dependency, the spectra of the iteration matrices of the resulting p-multigrid methods have been determined. Table 5.2 presents the spectral radius of the iteration matrix for the first benchmark when global ILUT is applied as a smoother. As the spectral radii are obtained numerically, the considered meshes are relatively coarse ($h = 2^{-5}$). Note that, the spectral radius increases when the number of patches is increased. Furthermore, the dependence on p is more erratic. In particular, the resulting p-multigrid method might diverge for one of the configurations, indicated

	$p=2$ # patches			$p=3$ # patches			$p=4$ # patches			$p=5$ # patches		
	4	16	64	4	16	64	4	16	64	4	16	64
$h=2^{-5}$	6	8	11	6	9	15	6	8	15	5	7	14
$h=2^{-6}$	6	7	8	6	8	10	7	9	13	7	8	13
$h=2^{-7}$	6	6	7	6	7	8	7	7	10	6	8	12

(a) CDR-equation on the unit square

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

(b) Poisson's equation on a quarter annulus

	$p=2$ # patches			$p=3$ # patches			$p=4$ # patches			$p=5$ # patches		
	4	16	64	4	16	64	4	16	64	4	16	64
$h=2^{-5}$	6	7	11	5	8	13	5	6	11	4	5	–
$h=2^{-6}$	6	6	8	6	8	10	5	8	12	5	7	10
$h=2^{-7}$	7	7	8	6	7	8	5	6	10	5	7	12

(c) Poisson's equation on an L-shaped domain

Table 5.1: Number of p-multigrid cycles needed to achieve convergence using global ILUT as a smoother.

by a spectral radius larger than 1. Note that, this is indeed the case, see Table 5.1.
Figure 5.2 shows the spectra of the iteration matrix (with $p=3$ and $h=2^{-5}$) for the first benchmark. Note that, for all number of patches, most of the eigenvalues are relatively close to the origin. However, a small number of eigenvalues is positioned significantly further from the origin in case of multiple patches. In case only a limited number of eigenvalues are relatively large, in absolute value, the use of an outer Krylov method to accelerate the overall convergence might be beneficial [100]. Note that, this is also known as using p-multigrid as a preconditioner. Table 5.3 shows the number of iterations needed with a Bi-CGSTAB method to achieve convergence when p-multigrid is applied as preconditioner. Here, a single V-cycle of the p-multigrid method, adopting global ILUT as a smoother, is applied every iteration.

# patches	$p=2$	$p=3$	$p=4$
4	0.094	0.090	0.062
16	0.156	0.187	0.146
64	0.275	0.374	**3.659**

Table 5.2: Asymptotic convergence rate of the p-multigrid adopting a global ILUT for different values of the spline degree p and number of patches.

For a relatively large value of the mesh width h, there is still a mild dependence on the number of patches, see for example $h = 2^{-5}$ and $p = 4$. The number of iterations is, however, independent of the spline degree p. For small values of h, the number of iterations is independent of p and the number of patches. Note that the use of the Bi-CGSTAB method restores stability for configurations that diverged before, but lead to a relatively high number of iterations.

	$p=2$ # patches			$p=3$ # patches			$p=4$ # patches			$p=5$ # patches		
	4	16	64	4	16	64	4	16	64	4	16	64
$h = 2^{-5}$	2	3	3	2	3	4	2	3	13	2	3	353
$h = 2^{-6}$	2	2	3	3	3	4	2	3	4	2	3	3
$h = 2^{-7}$	2	2	3	2	3	3	2	3	3	2	3	4

Table 5.3: Number of Bi-CGSTAB iterations needed to achieve convergence for the Poisson's equation on the quarter annulus. Here, a single p-multigrid cycle is applied as a preconditioner using global ILUT as smoother.

5

5.4. BLOCK ILUT

In the following, we consider a block ILUT smoother as an alternative to the global ILUT smoother in case of a multipatch geometry. This smoother is based on the observation that, in case of a multipatch geometry consisting of K patches, the resulting system matrix can be written as follows (see Figure 5.1):

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & & \mathbf{0} & \mathbf{A}_{1\Gamma} \\ & \ddots & & \vdots \\ \mathbf{0} & & \mathbf{A}_{KK} & \mathbf{A}_{K\Gamma} \\ \mathbf{A}_{\Gamma 1} & \cdots & \mathbf{A}_{\Gamma K} & \mathbf{A}_{\Gamma\Gamma} \end{bmatrix}. \tag{5.1}$$

Note that, in the remainder of this section, we drop the h and p as subscripts to improve readability. In [110] a preconditioner $\mathscr{P}$ is defined consisting of a coarse part $\mathscr{P}_C$ and fine part $\mathscr{P}_F$ in a multiplicative way. That is, the iteration matrix of the resulting preconditioner is obtained by multiplying the iteration matrix of both preconditioners:

$$\begin{aligned} \mathbf{I} - \mathscr{P}\mathbf{A} &= (\mathbf{I} - \mathscr{P}_C\mathbf{A})(\mathbf{I} - \mathscr{P}_F\mathbf{A}), && (5.2) \\ &= \mathbf{I} - \mathscr{P}_F\mathbf{A} - \mathscr{P}_C\mathbf{A} + \mathscr{P}_F\mathbf{A}\mathscr{P}_C\mathbf{A} && (5.3) \end{aligned}$$

and hence:

$$\mathscr{P} = \mathscr{P}_F + \mathscr{P}_C - \mathscr{P}_F\mathbf{A}\mathscr{P}_C. \tag{5.4}$$

The coarse part consists of a subdomain deflation approach [111] and takes care of the low-frequency components. However, it is known from multigrid that the coarse grid correction decreases the low-frequency components of the error. We therefore only adopt the fine part of the preconditioner (i.e., $\mathscr{P}_F$) in our block ILUT smoother and ignore $\mathscr{P}_C$. Hence, we have $\mathscr{P} = \mathscr{P}_F$ and will use the linear iteration with respect to $\mathscr{P}$ as

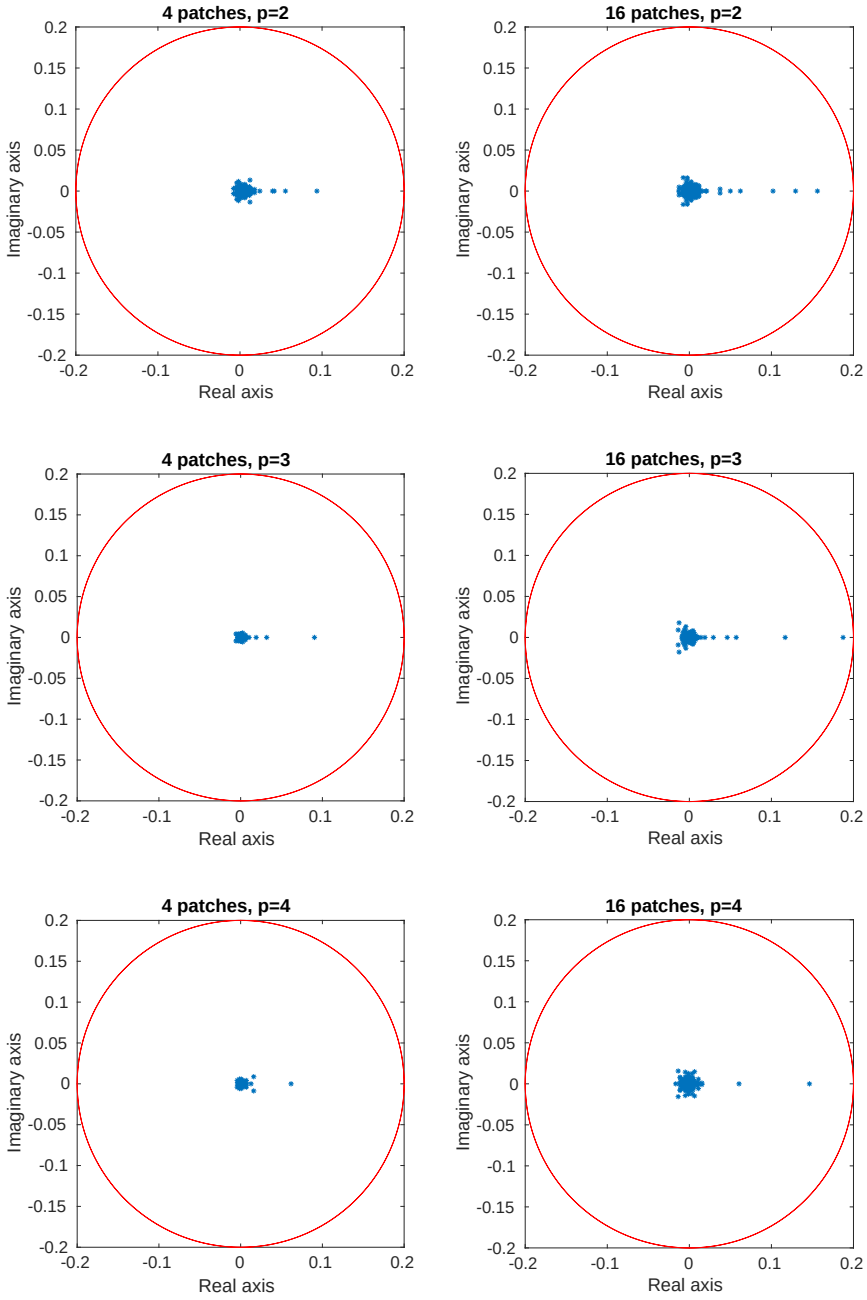


Figure 5.2: Spectra of the iteration matrix for Poisson's equation on the unit square obtained with p-multigrid ($h = 2^{-5}$). Here, global ILUT is applied as a smoother.

our smoother. Based on these considerations, the smoother $\mathscr{P}$ is then defined by observing that $\mathbf{A}$ can be written as follows:

$$\mathbf{A} = \mathbf{L}\mathbf{U} = \begin{bmatrix} \mathbf{L}_1 & & & \\ & \ddots & & \\ & & \mathbf{L}_K & \\ \mathbf{B}_1 & \cdots & \mathbf{B}_K & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U}_1 & & & \mathbf{C}_1 \\ & \ddots & & \vdots \\ & & \mathbf{U}_K & \mathbf{C}_K \\ & & & \mathbf{S} \end{bmatrix}. \tag{5.5}$$

Here $\mathbf{A}_{ii} = \mathbf{L}_i\mathbf{U}_i$, based on a complete LU factorization. Furthermore, we have $\mathbf{B}_i = \mathbf{A}_{\Gamma i}\mathbf{U}_i^{-1}$ and $\mathbf{C}_i = \mathbf{L}_i^{-1}\mathbf{A}_{i\Gamma}$. Finally, we have $\mathbf{S} = \mathbf{A}_{\Gamma\Gamma} - \sum_{i=1}^{K}\mathbf{B}_i\mathbf{C}_i$ to ensure that $\mathbf{A} = \mathbf{L}\mathbf{U}$.
A smoother can now be obtained by replacing the LU factorizations by ILUT factorizations:

$$\mathbf{A} \approx \tilde{\mathbf{L}}\tilde{\mathbf{U}} = \begin{bmatrix} \tilde{\mathbf{L}}_1 & & & \\ & \ddots & & \\ & & \tilde{\mathbf{L}}_K & \\ \tilde{\mathbf{B}}_1 & \cdots & \tilde{\mathbf{B}}_K & \mathbf{I} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{U}}_1 & & & \tilde{\mathbf{C}}_1 \\ & \ddots & & \vdots \\ & & \tilde{\mathbf{U}}_K & \tilde{\mathbf{C}}_K \\ & & & \tilde{\mathbf{S}} \end{bmatrix}. \tag{5.6}$$

Here $\mathbf{A}_{ii} \overset{(!)}{\approx} \tilde{\mathbf{L}}_i\tilde{\mathbf{U}}_i$ and all other blocks ($\tilde{\mathbf{B}}_i = \mathbf{A}_{\Gamma i}\tilde{\mathbf{U}}_i^{-1}$, $\tilde{\mathbf{C}}_i = \tilde{\mathbf{L}}_i^{-1}\mathbf{A}_{i\Gamma}$ and $\tilde{\mathbf{S}}$) are based on this ILUT factorization. Figure 5.3 depicts the sparsity pattern of the operator $\mathbf{A}_{h,p}$ and the block ILUT factorization $\mathbf{L}_{h,p} + \mathbf{U}_{h,p}$ for $p = 4$, $h = 2^{-4}$ and 4 patches. Note that, compared to the global ILUT factorization, the factorizations are obtained within each block. As a consequence, all non-zero entries of the global matrix $\mathbf{L}_{h,p} + \mathbf{U}_{h,p}$ stay within these blocks.

COMPUTATIONAL COSTS

In this subsection, the computational costs of the p-multigrid method adopting both smoothers are discussed. First, we discuss the assembly costs of the considered operators after which both setup and application costs of the block ILUT and global ILUT smoother are presented.
Assuming an element-based assembly loop with standard Gauss-quadrature, the assembly costs at level p for the stiffness matrix and transfer matrix are $\mathscr{O}(N_{\text{dof}}p^{3d})$ floating point operations (flops) [97]. Note that the stiffness matrix has to be assembled both at the high-order level and level $p = 1$ as a direct projection is considered within the p-multigrid method (see Section 4.3). Furthermore, the (variationally) lumped mass matrices have to be assembled for the prolongation and restriction operator, at the cost of $\mathscr{O}(N_{\text{dof}})$ flops. It was shown in [112] that the assembly costs form a significant part of the total computational costs within the p-multigrid method. Alternative (and more efficient) assembly techniques exist [69–71], but will not be explored in this dissertation.
At the low-order level, Gauss-Seidel is applied as a smoother, which costs $\mathscr{O}(N_{\text{dof}})$ flops per smoothing step and has no setup costs. The costs of setting up the block ILUT and global ILUT smoother at the high-order level differ significantly. The global ILUT factorization of $\mathbf{A}_{h,p}$ costs $\mathscr{O}(N_{\text{dof}}p^{2d})$ flops, provided that $\rho = 1$ and $\tau = 10^{-13}$ (i.e., no fill-in is allowed). Under these assumptions, applying the global ILUT smoother costs $\mathscr{O}(N_{\text{dof}}p^{d})$ flops. Setting up the block ILUT smoother consists of different steps. Here, we assume

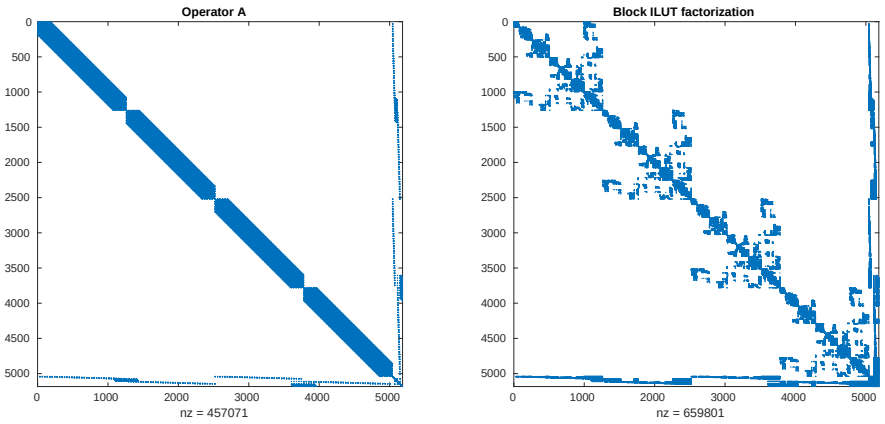


Figure 5.3: Sparsity pattern of $\mathbf{A}_{h,p}$ (left) and $\mathbf{L}_{h,p}+\mathbf{U}_{h,p}$ (right) for $p=4$, $h=2^{-5}$ and 4 patches for Poisson's equation on the quarter annulus.

that the total number of degrees of freedom N_{dof} is given by $N_{\text{dof}} = KN_{\text{patch}} + N_{\text{interface}}$, where N_{patch} is the number of degrees of freedom associated to the interior of a single patch and $N_{\text{interface}}$ denotes the total number of degrees of freedom associated to the interface. In general, N_{patch} is significantly higher compared to $N_{\text{interface}}$.

First, an ILUT factorization is determined for all patches, which costs $\mathcal{O}(N_{\text{patch}}p^{2d})$ for each patch. The matrices $\tilde{\mathbf{B}}_i, \tilde{\mathbf{C}}_i$ $(i = 1,\dots,K)$ are obtained by solving the following sytems, respectively, with multiple matrices:

$$\tilde{\mathbf{U}}_i^\top \tilde{\mathbf{B}}_i^\top = \mathbf{A}_{i\Gamma}^\top, \qquad \tilde{\mathbf{L}}_i \tilde{\mathbf{C}}_i = \mathbf{A}_{\Gamma i}.$$

Note that these solves only require forward substitutions and can therefore be performed efficiently and in parallel at the cost of $\mathcal{O}(N_{\text{patch}}p^d)$. Finally, $\tilde{\mathbf{S}}$ can be determined at the costs of $\mathcal{O}(N_{\text{interface}}^3)$, assuming the worst-case scenario of a full matrix $\mathbf{S}$. However, it holds that the off-diagonal entries $\mathbf{S}$ can only be non-zero if two basis functions belong to the same patch or neighbouring patches. Hence, for a large number of patches, $\mathbf{S}$ will not be a full matrix, but the number of non-zero entries per row grows when refinement is applied within a patch. Applying the block ILUT smoother $\mathcal{O}(N_{\text{patch}}p^d)$ flops, assuming the smoother is applied in parallel. The analysis of the computational costs show that the competitiveness of the block ILUT smoother in terms of CPU timings depends heavily on a parallel implementation of this smoother. This chapter, however, places the focus on the analysis of the above block ILUT smoother when applied to multipatch discretizations in Isogeometric Analysis and not on its efficient parallel implementation. Future research will focus on a parallel and distributed implementation of the considered block ILUT smoother.

SPECTRAL ANALYSIS

To analyze the asymptotic convergence rate of the p-multigrid method adopting the block ILUT smoother, the spectrum of the iteration matrix has been determined. Table 5.4 shows the spectral radius obtained for the first benchmark for different values of p with the block ILUT smoother for $h = 2^{-5}$. For comparison, the spectral radii obtained with the global ILUT smoother have been added as well.

	$p=2$		$p=3$		$p=4$	
# patches	Global	Block	Global	Block	Global	Block
4	0.094	0.015	0.090	0.011	0.062	0.005
16	0.156	0.019	0.187	0.015	0.146	0.005
64	0.275	0.039	0.374	0.067	**3.659**	0.058

Table 5.4: Asymptotic convergence rate of the p-multigrid adopting a global ILUT and block ILUT smoother for different values of the spline degree p and number of patches.

For both smoothers, the spectral radius increases when the number of patches is increased. For higher values of p, the spectral radius decreases when block ILUT is applied as a smoother. Note that, for all configurations, the spectral radius obtained with the block ILUT smoother is significantly lower compared to the one obtained with the global ILUT smoother. As a consequence, the p-multigrid method (using block ILUT as a smoother) is expected to show superior convergence behaviour. Figure 5.4 shows the spectrum of the iteration matrix for the p-multigrid method ($h = 2^{-5}$) adopting the global and block ILUT smoother. For the block ILUT smoother, the spectra are more clustered around the origin, resulting in a lower spectral radius. Furthermore, the dependency on the number of patches is hardly visible in the spectra, although it is mildly visible in Table 5.4. Therefore, the use of an outer Krylov solver is not expected to significantly decrease the dependency on the number of patches when block ILUT is applied as a smoother.

ITERATION NUMBERS

Based on the spectral analysis in the previous subsection, the use of the block ILUT smoother is expected to show improved iteration numbers compared to the p-multigrid method equiped with the global ILUT smoother. In this section, the number of p-multigrid cycles needed to achieve convergence is determined using the block ILUT smoother. As before, a reduction of the relative residual by 10^8 is chosen as a stopping criterion. We consider a random vector as an initial guess, where each entry is sampled from a uniform distribution on the interval $[-1,1]$ using the same seed.

Table 5.5 shows the number of multigrid cycles needed to achieve convergence for the considered benchmarks with block ILUT as a smoother. The number of iterations needed to achieve convergence with block ILUT as a smoother is, in general, independent of h. For higher values of p, the number of iterations slightly decreases. For one configuration, however, Poisson's equation on the quarter annulus, where $p = 5$ and $h = 2^{-5}$, the p-multigrid method is diverging (indicated by '$-$'). A small dependence on the number of patches is visible on the coarse mesh, but this depedency becomes negligible for smaller values of h. Note that, compared to global ILUT, the use of block ILUT leads to a

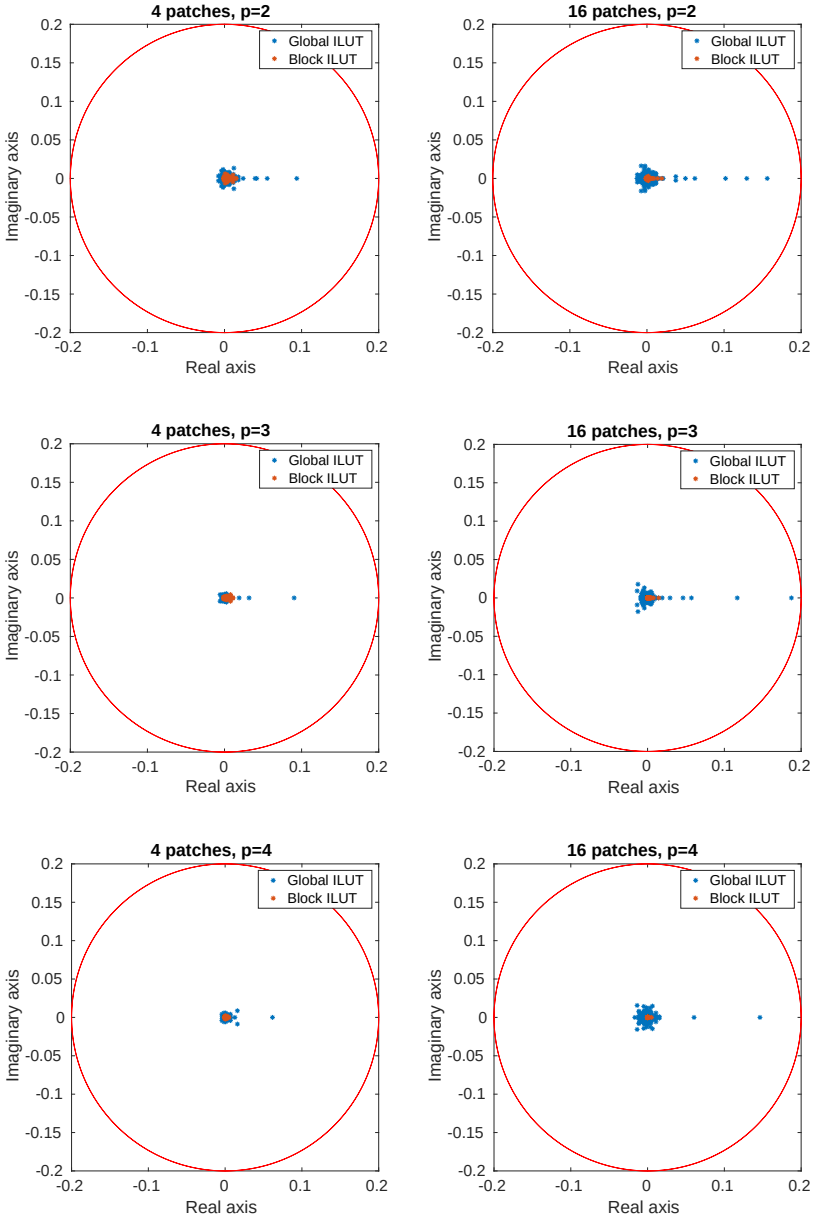


Figure 5.4: Spectra of the iteration matrix for Poisson's equation on the unit square obtained with p-multigrid ($h = 2^{-5}$). Here, global ILUT and block ILUT are applied as a smoother.

lower number of iterations for all configurations.

	$p=2$ # patches			$p=3$ # patches			$p=4$ # patches			$p=5$ # patches		
	4	16	64	4	16	64	4	16	64	4	16	64
$h=2^{-5}$	4	4	7	3	3	5	2	3	5	2	2	4
$h=2^{-6}$	4	4	5	3	3	4	3	3	4	3	3	3
$h=2^{-7}$	4	4	4	3	3	3	3	3	3	4	3	3

(a) CDR-equation on the unit square

	$p=2$ # 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	4	4	3	3	4	2	2	4	2	2	–
$h=2^{-6}$	3	3	4	3	3	4	3	3	3	3	3	3
$h=2^{-7}$	3	3	3	3	3	3	3	3	3	–	6	3

(b) Poisson's equation on a quarter annulus

	$p=2$ # 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	3	4	2	3	4	2	2	3	2	2	2
$h=2^{-6}$	3	3	3	3	3	3	2	2	3	2	2	2
$h=2^{-7}$	3	3	3	2	3	3	2	2	3	2	2	3

(c) Poisson's equation on an L-shaped domain

Table 5.5: Number of p-multigrid cycles needed to achieve convergence using block ILUT as a smoother for the considered benchmarks.

As the number of p-multigrid cycles when adopting (block) ILUT as a smoother is very low, the use of (block) ILUT as a stand-alone solver has been investigated as well. Table 5.6 shows the number of iterations needed for the second benchmark when both block ILUT and global ILUT are applied as a solver. For all configurations, the number of iterations is significantly higher compared to the number of p-multigrid cycles (see Table 5.5 and 5.1). Furthermore, a dependency on the mesh width h can be observed, making the use of (block) ILUT as a solver even less efficient for smaller values of h.
Alternatively, a single p-multigrid cycle can be applied as a preconditioner within a Bi-CGSTAB method. Table 5.7 shows the number of Bi-CGSTAB iterations needed to achieve convergence for the second benchmark, Poisson's equation on the quarter annulus. Compared to the use of p-multigrid as a solver, the number of Bi-CGSTAB iterations is only slightly lower when block ILUT is applied as a smoother. However, for the global ILUT smoother, a significant reduction of the iteration numbers can be achieved, as observed in the previous section. Note that the use of a Krylov solver restores stability for configurations that diverged before, but lead to a relatively high number of iterations.

	$p=2$ # patches			$p=3$ # patches			$p=4$ # patches			$p=5$ # patches		
	4	16	64	4	16	64	4	16	64	4	16	64
$h=2^{-5}$	28	22	16	14	10	14	8	6	10	4	4	–
$h=2^{-6}$	112	90	68	48	40	26	24	18	12	14	10	8
$h=2^{-7}$	406	386	312	178	162	130	86	64	54	48	38	26

(a) Block ILUT smoother

	$p=2$ # patches			$p=3$ # patches			$p=4$ # patches			$p=5$ # patches		
	4	16	64	4	16	64	4	16	64	4	16	64
$h=2^{-5}$	56	64	100	22	30	56	14	16	–	8	12	–
$h=2^{-6}$	220	218	234	84	84	100	40	42	56	24	26	36
$h=2^{-7}$	710	700	782	276	302	296	138	138	148	80	80	76

(b) Global ILUT smoother

Table 5.6: Number of iterations needed to achieve convergence for Poisson's equation on the quarter annulus. Here, block ILUT and global ILUT are used as a solver.

	$p=2$ # patches			$p=3$ # patches			$p=4$ # patches			$p=5$ # patches		
	4	16	64	4	16	64	4	16	64	4	16	64
$h=2^{-5}$	2	2	2	2	2	2	1	1	2	1	1	50
$h=2^{-6}$	2	2	2	2	2	2	2	2	2	2	2	2
$h=2^{-7}$	2	2	2	2	2	2	2	2	2	34	3	2

(a) Block ILUT smoother

	$p=2$ # patches			$p=3$ # patches			$p=4$ # patches			$p=5$ # patches		
$h=2^{-5}$	2	3	3	2	3	4	2	3	13	2	3	353
$h=2^{-6}$	2	2	3	3	3	4	2	3	4	2	3	3
$h=2^{-7}$	2	2	3	2	3	3	2	3	3	2	3	4

(b) Global ILUT smoother

Table 5.7: Number of Bi-CGSTAB iterations needed to achieve convergence for Poisson's equation on the quarter annulus. Here, a single p-multigrid cycle is applied as a preconditioner using block ILUT and global ILUT as smoother.

5.5. APPLICATION: YETI FOOTPRINT

In the previous sections, we applied a global ILUT and the proposed block ILUT smoother on benchmarks defined on geometries consisting of a different number of patches. In general, the application of both smoothers within a p-multigrid method resulted in iteration numbers independent of h and p. A small dependency on the number of patches was noticeable, however, in particular on coarser meshes.

In this section, the block ILUT and global ILUT smoother are applied on a more challenging geometry. We consider Poisson's equation on a two-dimensional Yeti footprint,

where the exact solution is given by $u(x,t) = \sin(5\pi x)\sin(5\pi y)$:

$$\begin{aligned} -\Delta u &= 50\pi^2 \sin(5\pi x)\sin(5\pi y), && (x,y) \in \Omega, \\ u &= 0, && (x,y) \in \partial\Omega. \end{aligned}$$

This benchmark has been considered in the literature in the context of a parallel multigrid method for Isogeometric Analysis, see [86]. Figure 5.5 shows the Yeti footprint multipatch geometry (left) and its exact solution (right) consisting of 21 patches.

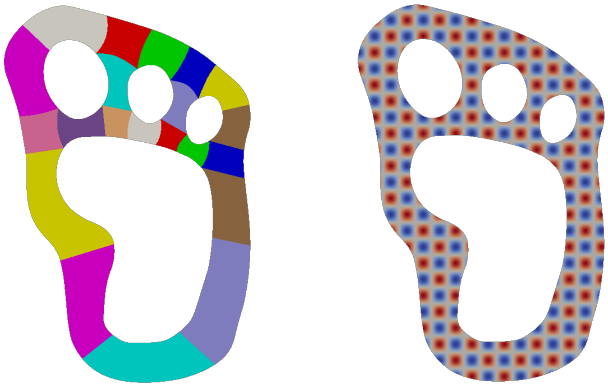


Figure 5.5: Underlying multipatch geometry (left) and the exact solution (right) of the Yeti footprint.

Table 5.8 shows the number of multigrid cycles needed to reach convergence for the p-multigrid method adopting both smoothers and the aforementioned convergence criterion. For all configurations, the use of block ILUT leads to a lower number of cycles compared to the use of global ILUT. Furthermore, the number of iterations needed to reach convergence is, in general, lower for higher values of the spline degree p. Results are, to a large extent, comparable with the ones presented in the previous section. A small h-dependence can be observed, however, for smaller values of p when applying the global ILUT smoother.

	$p=2$		$p=3$		$p=4$		$p=5$	
	Global	Block	Global	Block	Global	Block	Global	Block
$h=2^{-3}$	5	4	4	2	4	2	4	2
$h=2^{-4}$	8	4	5	3	5	3	4	2
$h=2^{-5}$	8	4	6	3	5	3	5	3

Table 5.8: Number of p-multigrid cycles needed to reach convergence for the Yeti footprint with global and block ILUT as a smoother.

5.6. CONCLUDING REMARKS

In this chapter, we applied the p-multigrid method presented in Chapter 4 to benchmarks involving multipatch geometries. The direct use of this p-multigrid method using global ILUT as a smoother leads to iteration numbers which are independent of the mesh width h and spline degree p. On coarser meshes, a dependency on the number of patches can be observed, which can be successfully mitigated by the use of an outer Krylov solver. Furthermore, we presented a block ILUT smoother as an alternative smoother in the multipatch setting. The block ILUT smoother is based on a preconditioner, which has been developed in the context of domain decomposition methods [110]. The key idea is to make use of the block structure of the resulting system matrix when setting up the smoother. As a consequence, the ILUT factorizations are only obtained for smaller blocks, where the number of blocks depends on the number of patches describing the geometry. A spectral analysis showed that this smoother is more effective when applied within a p-multigrid method compared to a global ILUT smoother. Numerical results, obtained for a variety of two-dimensional benchmarks, indeed showed that the number of iterations needed to achieve convergence is lower for all considered configurations when adopting the block ILUT smoother. Future research will focus on the comparison of the presented block ILUT smoother with the global ILUT smoother in terms of computational efficiency.

6

Multigrid Reduced in Time for Isogeometric Analysis

In this chapter, we combine the Multigrid Reduced in Time (MGRIT) method with Isogeometric Analysis to solve time-dependent partial differential equations parallel-in-time. In particular, we investigate the convergence of MGRIT for different geometries, time integration schemes and cycle types adopting a standard iterative method for the spatial solves. Then, we replace this standard solver with our p-multigrid method to further reduce the computational costs, in particular for higher values of p. Finally, we determine the scaling properties (i.e., weak and strong scaling) of the resulting method on modern architectures.

Parts of this chapter are based on:
R. Tielen, M. Möller and C. Vuik, *Multigrid Reduced in Time for Isogeometric Analysis*, in The Proceedings of the Young Investigators Conference (2021)

Parts of this chapter are based on:
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 [math.NA]

6.1. INTRODUCTION

For time-dependent partial differential equations (PDEs), Isogeometric Analysis is often combined with a time integration scheme within the method of lines. However, as with all traditional time integration schemes, the latter part is sequential by design and, hence, a bottleneck for parallelization in numerical simulations. When the spatial resolution is increased to improve accuracy, the time step size might have to be reduced as well to reduce the overall discretization error. At the same time, processors' clock speeds are no longer increasing, but the core count goes up, which calls for the parallelization of the calculation process to benefit from modern computer hardware. As traditional time integration schemes are sequential by nature, new parallel-in-time methods are needed to resolve this problem.

The Multigrid Reduced in Time (MGRIT) method [113] is a parallel-in-time algorithm based on multigrid reduction (MGR) techniques [114] showing many similarities with the Parareal algorithm [115]. In contrast to space-time methods, in which time is considered as an extra spatial dimension, traditional time stepping is still necessary within MGRIT, but is performed in parallel. Space-time methods have been combined in the literature with Isogeometric Analysis [116–119]. Although very successful, a drawback of such methods is the fact that they are more intrusive on existing codes, while MGRIT just requires a routine to integrate the fully discrete problem between two time instances. Over the years, MGRIT has been studied in detail (see [120–124]) and applied to a variety of problems, including those arising in optimization [125, 126] and power networks [127, 128].

In this chapter we combine the MGRIT method with Isogeometric Analysis to numerically solve time-dependent PDEs. First, we introduce the considered model problem and its discretization in Section 6.2 after which we describe the MGRIT method in Section 6.3. Numerical results, including CPU timings, obtained for different geometries and time integration schemes are presented for different configurations of the MGRIT method in Sections 6.4 and 6.5. In particular, we consider the use of a standard iterative method and the p-multigrid method for the spatial solves within the MGRIT method. Furthermore, the scaling properties (i.e., weak and strong scaling) of the resulting method on modern architectures are investigated in Section 6.6. Finally, conclusions are drawn in Section 6.7.

6.2. MODEL PROBLEM AND DISCRETIZATION

As a model problem, we consider the transient diffusion equation:

$$\partial_t u(\mathbf{x},t) - \kappa \Delta u(\mathbf{x},t) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, t \in [0,T]. \tag{6.1}$$

Here, κ denotes a constant diffusion coefficient, $\Omega \subset \mathbb{R}^d$ a connected, Lipschitz domain in d dimensions and $f \in L^2(\Omega)$ a source term. The above equation is complemented by initial conditions and homogeneous Dirichlet boundary conditions:

$$u(\mathbf{x},0) = u^0(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{6.2}$$

$$u(\mathbf{x},t) = 0, \quad \mathbf{x} \in \partial\Omega, t \in [0,T]. \tag{6.3}$$

First, we discretize Equation (6.1) in time by dividing the time interval $[0, T]$ in N_t subintervals of size Δt and applying the θ-scheme to the temporal derivative, which leads to the following equation to be solved at every time step $k = 0, \dots, N_t$:

$$\frac{u(\mathbf{x})^{k+1} - u(\mathbf{x})^k}{\Delta t} = \kappa\theta\Delta u(\mathbf{x})^{k+1} + \kappa(1-\theta)\Delta u(\mathbf{x})^k + f(\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{6.4}$$

Depending on the choice of θ, this scheme leads to the backward Euler ($\theta = 1$), forward Euler ($\theta = 0$) or the second-order in time Crank-Nicolson ($\theta = 0.5$) method, which will all be adopted throughout this chapter. By rearranging the terms, the discretized equation can be written as follows:

$$u(\mathbf{x})^{k+1} - \kappa\Delta t\theta\Delta u(\mathbf{x})^{k+1} = u(\mathbf{x}^k) + \kappa\Delta t(1-\theta)\Delta u(\mathbf{x})^k + \Delta t f(\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{6.5}$$

To obtain the variational formulation, let $\mathcal{V} = H_0^1(\Omega)$ be the space of functions in the Sobolev space $H^1(\Omega)$ that vanish on the boundary $\partial\Omega$. Equation (6.5) is multiplied with a test function $v \in \mathcal{V}$ and the result is then integrated over the domain Ω:

$$\int_\Omega u^{k+1} v - \kappa\Delta t\theta\Delta u^{k+1} v \,\mathrm{d}\Omega = \int_\Omega u^k v + \kappa\Delta t(1-\theta)\Delta u^k v + \Delta t f v \,\mathrm{d}\Omega. \tag{6.6}$$

Applying integration by parts on the second term on both sides of the equation results in

$$\int_\Omega u^{k+1} v + \kappa\Delta t\theta\nabla u^{k+1}\cdot\nabla v \,\mathrm{d}\Omega = \int_\Omega u^{k+1} v - \kappa\Delta t(1-\theta)\nabla u^k\cdot\nabla v + \Delta t f v \,\mathrm{d}\Omega, \tag{6.7}$$

where the boundary integral vanishes since $v = 0$ on $\partial\Omega$. As described in Chapter 2, a geometry function $\mathbf{F}$ (or a family of functions in case of a multipatch geometry) is defined to parameterize the physical domain Ω. Furthermore, the solution u is approximated at every time step by a linear combination of multivariate B-spline basis functions. Denoting the total number of multivariate B-spline basis functions $\Phi_{i,p}$ by N_{dof}, the solution u is thus approximated at every time step as follows:

$$u(\mathbf{x}) \approx u_{h,p}(\mathbf{x}) = \sum_{i=1}^{N_{\text{dof}}} u_i \Phi_{i,p}(\mathbf{x}), \quad u_{h,p} \in \mathcal{V}_{h,p}, \tag{6.8}$$

where we omit the superscript denoting the time step to improve readability. Here, the spline space $\mathcal{V}_{h,p}$ is defined, using the inverse of the geometry mapping $\mathbf{F}^{-1}$ as pull-back operator, as follows:

$$\mathcal{V}_{h,p} := \operatorname{span}\left\{\Phi_{i,p}\circ\mathbf{F}^{-1}\right\}_{i=1,\dots,N_{\text{dof}}}. \tag{6.9}$$

By setting $v = \Phi_{j,p}$, Equation (6.7) can be written as follows:

$$(\mathbf{M} + \kappa\Delta t\theta\mathbf{K})\,\mathbf{u}^{k+1} = (\mathbf{M} - \kappa\Delta t(1-\theta)\mathbf{K})\,\mathbf{u}^k + \Delta t\mathbf{f}, \quad k = 0, \dots, N_t, \tag{6.10}$$

where $\mathbf{M}$ and $\mathbf{K}$ denote the mass and stiffness matrix, respectively:

$$\mathbf{M}_{i,j} = \int_\Omega \Phi_{i,p}\Phi_{j,p} \,\mathrm{d}\Omega, \qquad \mathbf{K}_{i,j} = \int_\Omega \nabla\Phi_{i,p}\cdot\nabla\Phi_{j,p} \,\mathrm{d}\Omega, \quad i,j = 1,\dots,N_{\text{dof}}. \tag{6.11}$$

A traditional (i.e., sequential) time integration scheme would solve Equation (6.10) for $k = 0, \dots, N_t$ to obtain the numerical solution at each time instance. In this chapter, however, we apply the MGRIT method to solve Equation (6.10) parallel-in-time.

6.3. MULTIGRID REDUCED IN TIME (MGRIT)

In this section, we discuss the Multigrid Reduced in Time (MGRIT) approach based on the description presented in [113] and [129]. For the ease of notation, we set $\theta = 1$ in Equation (6.10) throughout the remainder of this section. Let $\Psi = (\mathbf{M} + \kappa \Delta t \mathbf{K})^{-1}$ denote the inverse of the left-hand side operator. Equation (6.10) can then be written as follows:

$$\mathbf{u}^{k+1} = \Psi \mathbf{M} \mathbf{u}^k + \mathbf{g}^{k+1},\ k = 0, \dots, N_t, \tag{6.12}$$

where $\mathbf{g}^{k+1} = \Psi \Delta t \mathbf{f}$. Setting $\mathbf{g}^0$ equal to the initial condition $u^0(\mathbf{x})$ projected on the spline space $\mathcal{V}_{h,p}$, the sequence of all time-integration steps can be written as a linear system of equations:

$$\mathbf{A}\mathbf{u} = \begin{bmatrix} I & & & \\ -\Psi\mathbf{M} & I & & \\ & \ddots & \ddots & \\ & & -\Psi\mathbf{M} & I \end{bmatrix} \begin{bmatrix} \mathbf{u}^0 \\ \mathbf{u}^1 \\ \vdots \\ \mathbf{u}^{N_t} \end{bmatrix} = \begin{bmatrix} \mathbf{g}^0 \\ \mathbf{g}^1 \\ \vdots \\ \mathbf{g}^{N_t} \end{bmatrix} = \mathbf{g}. \tag{6.13}$$

A sequential time integration scheme would correspond to a block-forward solve of this linear system of equations. Throughout this chapter, we use MGRIT to solve Equation (6.13). First, we introduce the two-level MGRIT method, showing similarities with the well-known Parareal algorithm [115], which has been investigated in detail in the literature [130–134]. In fact, MGRIT can be considered as a generalization of the Parareal method [123]. Afterwards, we will present the multilevel variant of MGRIT in more detail.

The two-level MGRIT method combines the use of a computationally cheap coarse-level time integration method with an accurate but more expensive fine-level one, which can be performed in parallel. That is, Equation (6.13) can be solved iteratively by introducing a coarse temporal mesh with time step size $\Delta t_C = m \Delta t_F$. Here, the step size of the fine temporal mesh Δt_F coincides with the Δt from the previous sections and m denotes the coarsening factor. Figure 6.1 illustrates both the fine and coarse temporal meshes.

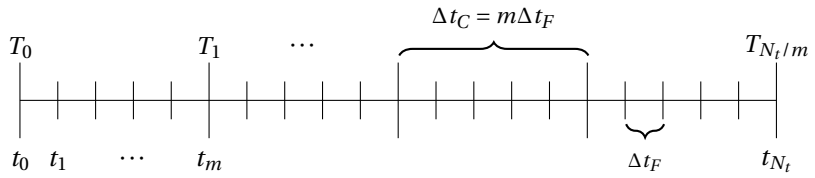


Figure 6.1: Coarse and fine temporal mesh from 0 to T.

The time instances $T_0, T_1, \dots, T_{N_t/m}$ are referred to as coarse points (or C-points), while the remaining points are called fine points (or F-points). By applying a numbering strategy that first numbers the F-points and then the C-points, we can write Equation (6.13) as follows:

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FC} \\ \mathbf{A}_{CF} & \mathbf{A}_{CC} \end{bmatrix} \begin{bmatrix} \mathbf{u}_F \\ \mathbf{u}_C \end{bmatrix} = \begin{bmatrix} \mathbf{g}_F \\ \mathbf{g}_C \end{bmatrix}, \tag{6.14}$$

where the matrix $\mathbf{A}$ can be decomposed as follows:

$$\begin{bmatrix} \mathbf{A}_{FF} & \mathbf{A}_{FC} \\ \mathbf{A}_{CF} & \mathbf{A}_{CC} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_F & \mathbf{0} \\ \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1} & \mathbf{I}_C \end{bmatrix} \begin{bmatrix} \mathbf{A}_{FF} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I}_F & \mathbf{A}_{FF}^{-1}\mathbf{A}_{FC} \\ \mathbf{0} & \mathbf{I}_C \end{bmatrix}, \tag{6.15}$$

where $\mathbf{I}_C$ and $\mathbf{I}_F$ are identity matrices. The 'ideal' restriction and prolongation operator are then defined as follows:

$$R = \begin{bmatrix} \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1} & \mathbf{I}_C \end{bmatrix}, \qquad P = \begin{bmatrix} -\mathbf{A}_{FF}^{-1}\mathbf{A}_{FC} \\ \mathbf{I}_C \end{bmatrix}. \tag{6.16}$$

Within MGRIT, the 'ideal' prolongation operator P is typically adopted, while the 'ideal' restriction operator is replaced by $\tilde{R} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_C \end{bmatrix}$. The matrix $\mathbf{A}_{FF}$ is given by:

$$\mathbf{A}_{FF} = \begin{bmatrix} \mathbf{A}_\Psi & & \\ & \ddots & \\ & & \mathbf{A}_\Psi \end{bmatrix}, \quad \mathbf{A}_\Psi = \begin{bmatrix} I & & & \\ -\Psi\mathbf{M} & I & & \\ & \ddots & \ddots & \\ & & -\Psi\mathbf{M} & I \end{bmatrix}. \tag{6.17}$$

Note that each solve with $\mathbf{A}_\Psi$ corresponds to a single time step within a coarse interval, which is a completely independent process for each coarse interval and can therefore be performed in parallel. The Schur complement matrix $\mathbf{S}$ in Equation (6.15) is given by:

$$\mathbf{S} = \mathbf{A}_{CC} - \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1}\mathbf{A}_{FC} = \begin{bmatrix} I & & & \\ -(\Psi\mathbf{M})^m & I & & \\ & \ddots & \ddots & \\ & & -(\Psi\mathbf{M})^m & I \end{bmatrix}. \tag{6.18}$$

Instead of solving for $\mathbf{S}$ directly, MGRIT solves for a modified matrix $\tilde{\mathbf{S}}$ by replacing the operator $(\Psi\mathbf{M})^m \approx \Phi\mathbf{M}$, which corresponds to applying a single time step at the coarse level. As a true multigrid method, the building blocks of the MGRIT method consist of *relaxation* (= fine time stepping) and a *coarse grid correction* (= coarse time stepping). Relaxation involves solving a linear system of the form

$$\mathbf{A}_{FF}\mathbf{u}_F = \mathbf{g}_F - \mathbf{A}_{FC}\mathbf{u}_C, \tag{6.19}$$

where $\mathbf{u}_F$ and $\mathbf{u}_C$ denote the solution at all F-points and C-points, respectively. Within relaxation, the solution is updated at the F-points based on the given values at the C-points. This time stepping from a coarse point C to all neighbouring fine points is also referred to as F-relaxation [113]. On the other hand, time stepping to a C-point from the previous F-point is referred to as C-relaxation. It should be noted that both types of relaxation are highly parallel and can be combined leading to so-called CF- or FCF-relaxation. Figure 6.2 illustrates both C- and F-relaxation methods.
The coarse grid correction involves solving the linear system of equations

$$\tilde{\mathbf{S}}\mathbf{u}_C = \tilde{R}\left(\mathbf{g} - \mathbf{A}\mathbf{u}\right), \tag{6.20}$$

which is a sequential procedure by design, but is much cheaper compared to the fine time integration (which can be performed in parallel). Here, the vector $\mathbf{u}_C$ is obtained by applying $\tilde{R}$ on $\mathbf{u}$.

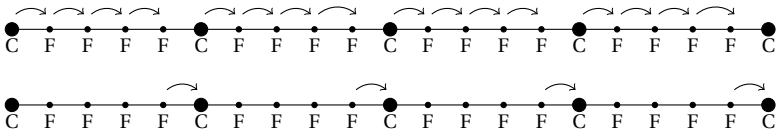


Figure 6.2: Illustration of F-relaxation (top) and C-relaxation (bottom).

MULTILEVEL MGRIT METHOD

The solution procedure described above can be extented to a true multilevel MGRIT method. First, we define a hierarchy of L temporal meshes, where the time step size for the discretization at level l ($l = 0, 1, \dots L$) is given by $\Delta t_F m^l$. The total number of levels L is related to the coarsening factor m and the total number of fine steps Δt_F by $L = \log_m(N_t)$. Let $\mathbf{A}^{(l)}\mathbf{u}^{(l)} = \mathbf{g}^{(l)}$ denote the linear system of equations based on the considered time step size at level l. The MGRIT method can then be written as follows:

MGRIT

1. Apply F-relaxation (= fine time stepping) on $\mathbf{A}^{(l)}\mathbf{u}^{(l)} = \mathbf{g}^{(l)}$:
$$\mathbf{A}^{(l)}_{FF}\mathbf{u}^{(l)}_F = \mathbf{g}^{(l)}_F - \mathbf{A}^{(l)}_{FC}\mathbf{u}^{(l)}_C.$$
2. Determine the residual at level l and restrict it to level $l+1$ using the restriction operator $\tilde{R}$:
$$\mathbf{r}^{(l+1)} = \tilde{R}\left(\mathbf{g}^{(l)} - \mathbf{A}^{(l)}\mathbf{u}^{(l)}\right).$$
3. Solve Equation (6.20) (= coarse time stepping) to obtain $\mathbf{u}^{(l+1)}$:
$$\tilde{\mathbf{S}}\mathbf{u}^{(l+1)} = \mathbf{r}^{(l+1)}.$$
4. Prolongate the correction using the 'ideal' interpolation operator P and update the solution at level l:
$$\mathbf{u}^{(l)} := \mathbf{u}^{(l)} + P\mathbf{u}^{(l+1)}.$$

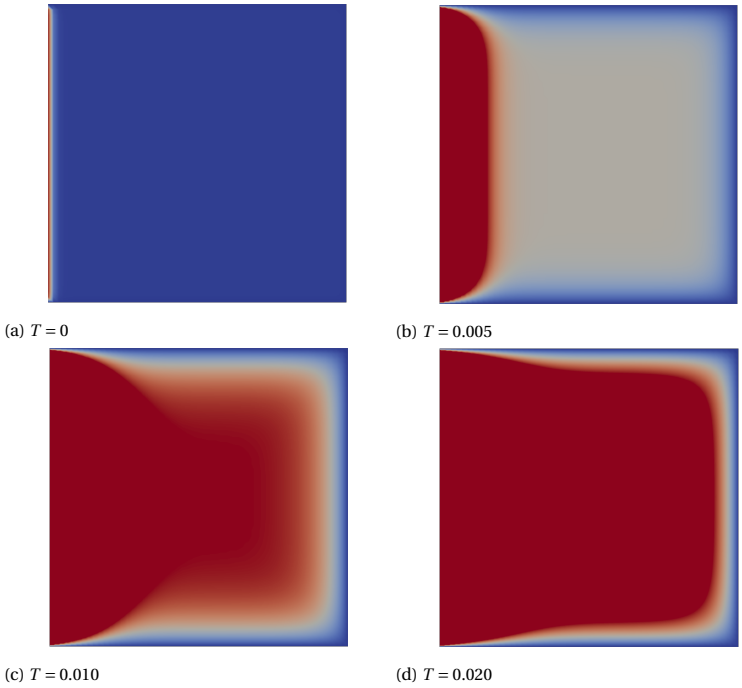
Recursive application of this scheme until the coarsest level is reached, leads to a so-called V-cycle. However, as with standard multigrid methods, alternative cycle types (i.e. W-cycles, F-cycles) can be defined. At all levels of the multigrid hierarchy, the operators are obtained by rediscretizing Equation (6.1) using a different time step size.

6.4. NUMERICAL RESULTS (CONJUGATE GRADIENT METHOD)

To assess the effectiveness of the MGRIT method when applied in combination with Isogeometric Analysis, we solve Equation (6.1) on the time domain $T = [0, 0.1]$, where the initial condition is chosen equal to zero and the right-hand side equal to one. The MGRIT method is said to have reached convergence if the relative residual at the end of an iteration is smaller than or equal to 10^{-10}, unless stated otherwise. Figure 6.3 shows the resulting solution u at different time instances for $\Omega = [0,1]^2$. Here, an inhomogeneous Neumann boundary condition is applied at the left boundary.

Throughout this section, the MGRIT hierarchy, the domain of interest Ω and the time integration scheme are varied. The MGRIT hierarchies that will be adopted are a two-level method, a V-cycle and an F-cycle. As a domain, we consider the unit square (i.e., $\Omega = [0,1]^2$), a quarter annulus defined in the first quadrant with an inner radius of 1 and an outer radius of 2 and a multipatch geometry, see Figure 6.4. As a time integration scheme, we consider a value of θ within the θ-scheme of 0, 0.5 and 1 throughout this section, which corresponds to forward Euler, Crank-Nicolson and backward Euler, respectively.

All numerical experiments in the remainder of this chapter have been performed using XBraid, a parallel-in-time software package that implements the MGRIT method. More information on reproducing all numerical experiments can be found in Chapter 7.



(a) $T = 0$ (b) $T = 0.005$

(c) $T = 0.010$ (d) $T = 0.020$

Figure 6.3: Solution to Equation (6.1) on the unit square at different times T using a inhomogeneous Neumann boundary condition at the left boundary using quadratic B-spline basis functions.

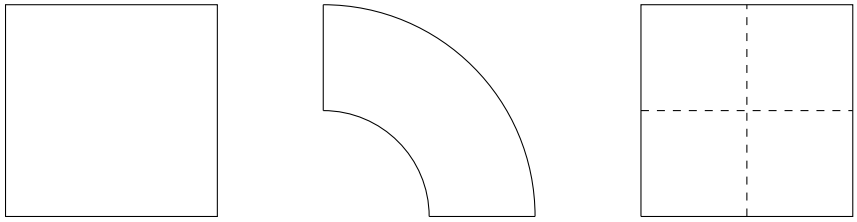


Figure 6.4: Spatial domains Ω considered throughout this section.

MGRIT HIERARCHIES

First, we consider the MGRIT method using different hierarchies when backward Euler is adopted for the time integration to solve Equation (6.1). At each time step, the linear system (i.e., Equation (6.10)) is approximately solved by the Conjugate Gradient method combined with a diagonal preconditioner (where $\epsilon = 10^{-12}$). Table 6.1 shows the number of MGRIT iterations for different values of N_t and p when a two-level method, V-cycles or F-cycles are considered as MGRIT hierarchy. Varying the number of time steps is in particular interesting as MGRIT is a parallel-in-time method, where speed-ups will primarily come from parallelization in the temporal component. Here, F-relaxation is applied at all levels of the MGRIT hierarchy. The mesh width for all configurations equals $h = 2^{-6}$. For all three hierarchies, the number of MGRIT iterations needed to reach convergence is independent of N_t and p. The results obtained with a two-level method or F-cycles are identical and lead to a lower number of iterations compared to the use of V-cycles for all configurations.

	$p=2$			$p=3$			$p=4$			$p=5$		
	TL	V	F	TL	V	F	TL	V	F	TL	V	F
$N_t = 250$	7	10	7	7	10	7	7	10	7	7	10	7
$N_t = 500$	7	10	7	7	10	7	7	10	7	7	10	7
$N_t = 1000$	7	11	7	7	11	7	7	11	7	7	11	7
$N_t = 2000$	7	11	7	7	11	7	7	11	7	7	11	7

Table 6.1: Number of MGRIT iterations for solving the model problem when adopting a two-level (TL) method, V-cycles (V) and F-cycles (F), respectively.

Instead of increasing the number of time steps, the mesh width can be varied as well. Table 6.2 shows the number of MGRIT iterations adopting different hierarchies for a fixed number of time steps ($N_t = 100$) and different values of p and h. For all configurations, the use of a two-level method or F-cycles leads to a lower number of iterations compared to the use of V-cycles. In particular, the number of iterations are independent of the mesh width for all MGRIT hierarchies and comparable to the ones obtained when considering different values of the number of time steps.

	$p=2$			$p=3$			$p=4$			$p=5$		
	TL	V	F	TL	V	F	TL	V	F	TL	V	F
$h=2^{-4}$	7	9	7	7	9	7	7	9	7	7	9	7
$h=2^{-5}$	7	9	7	7	9	7	7	9	7	7	9	8
$h=2^{-6}$	8	9	8	8	9	8	8	9	8	8	9	8
$h=2^{-7}$	8	9	8	8	9	8	8	9	8	8	9	8

Table 6.2: Number of MGRIT iterations for solving the model problem when adopting a two-level (TL) method, V-cycles (V) and F-cycles (F), respectively.

VARYING GEOMETRIES

Next, we apply MGRIT on a curved and a multipatch geometry, respectively. Table 6.3 shows the number of V-cycles needed with MGRIT when backward Euler is adopted for the time integration. Results can be compared to the ones presented in Table 6.1, showing identical iteration numbers for all geometries.

	Quarter Annulus				Multipatch			
	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
$N_t=250$	10	10	10	10	10	10	10	10
$N_t=500$	10	10	10	10	10	10	10	10
$N_t=1000$	11	11	11	11	11	11	11	11
$N_t=2000$	11	11	11	11	11	11	11	11

Table 6.3: Number of MGRIT iterations for solving Equation (6.1) on a quarter annulus and multipatch geometry when adopting V-cycles for varying time step sizes.

Table 6.4 shows the results when the number of time steps is kept constant ($N_t = 100$) for the quarter annulus and multipatch geometry when adopting V-cycles. Results can be compared to Table 6.2 and are (again) identical for all three geometries.

	Quarter Annulus				Multipatch			
	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
$h=2^{-4}$	9	9	9	9	9	9	9	9
$h=2^{-5}$	9	9	9	9	9	9	9	9
$h=2^{-6}$	9	9	9	9	9	9	9	9
$h=2^{-7}$	9	9	9	9	9	9	9	9

Table 6.4: Number of MGRIT iterations for solving Equation (6.1) on a quarter annulus and multipatch geometry when adopting V-cycles for varying mesh widths.

TIME INTEGRATION SCHEMES

In addition to the implicit backward Euler scheme, we have considered alternative time integration schemes as well. In this subsection, we investigate the use of the forward Euler and (second-order accurate) Crank-Nicolson method. The use of explicit time integration schemes in the context of parallel-in-time integration is on the one hand highly relevant, as the required number of time steps needed to ensure stability is relatively high. On the other hand, coarsening with respect to the time step size might still exhibit stability issues at coarser levels. Therefore, explicit-implicit methods are often considered, where explicit time integration is applied on the finest temporal level, while implicit methods are adopted at the coarser levels. The question remains to which extent the resulting MGRIT algorithm remains robust in the mesh width and/or spline degree. Table 6.5 shows the number of MGRIT iterations for different numbers of time steps when adopting V-cycles and a mesh width of $h = 2^{-4}$. Here, forward Euler/Crank-Nicolson is applied at the fine level, while backward Euler is applied at the coarse levels. For some of the considered configurations, the resulting MGRIT method does not converge within 100 iterations for forward Euler (indicated by '–'). It should be noted, however, that for these configurations, forward Euler applied as a sequential time integration scheme does not converge either, which is a direct consequence of the CFL condition. When the Crank-Nicolson method is applied the resulting MGRIT method converges in a relatively low number of iterations.

	Forward Euler				Crank-Nicolson			
	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
$N_t = 250$	–	–	–	–	11	11	14	24
$N_t = 500$	13	–	–	–	11	11	11	12
$N_t = 1000$	13	13	–	–	11	11	11	11
$N_t = 2000$	13	13	13	–	11	11	11	11

Table 6.5: Number of MGRIT iterations for solving Equation (6.1) on the unit square using forward Euler and Crank-Nicolson when adopting V-cycles.

Table 6.6 shows the number of MGRIT iterations for a varying mesh width and 1000 time steps for both time integration methods. For many configurations, MGRIT using forward Euler does not converge, while the Crank-Nicolson method converges for all configurations. A small dependency on h and p is, however, visible.

	Forward Euler				Crank-Nicolson			
	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
$h = 2^{-3}$	13	13	13	14	11	11	11	12
$h = 2^{-4}$	13	13	–	–	11	11	11	11
$h = 2^{-5}$	–	–	–	–	11	11	13	23
$h = 2^{-6}$	–	–	–	–	13	28	52	88

Table 6.6: Number of MGRIT iterations for solving Equation (6.1) on the unit square using forward Euler and Crank-Nicolson when adopting V-cycles.

CPU TIMINGS

Up to now, focus has been on the number of iterations needed to reach convergence with MGRIT. In this section, we will focus on the computational efficiency of the MGRIT method for different values of the number of time steps N_t and spline degree p. Here, we adopt V-cycles, a mesh width of $h = 2^{-6}$ and the unit square as our domain of interest. Note that the corresponding iteration numbers can be found in Table 6.1. The computations are performed on three nodes, which consist each of an Intel(R) i7-10700 (@ 2.90GHz) processor with 8 cores.

Figure 6.5 shows the CPU time needed to reach convergence with MGRIT for a different number of time steps N_t and spline degree p. Here the total number of cores is varied between 3 and 24, but are always evenly distributed over the three nodes. As can be observed in all figures, doubling the number of time steps roughly doubles the time needed to reach convergence for all values of p.

Furthermore, it can be observed that the CPU times significantly increase for higher values of p which is related to the spatial solves required at every time step. As standard iterative solvers (like the Conjugate Gradient method) have a deteriorating performance for increasing values of p, more iterations are required to reach convergence for each spatial solve, resulting in higher computational costs of the MGRIT method.

When focussing on the results obtained with 3 and 6 cores, it can be seen that doubling the number of cores significantly reduces the CPU time needed to reach convergence. More precisely, a reduction of $45-50\%$ can be observed when doubling the number of cores to 6, implying the MGRIT algorithm is highly parallelizable. When the number of cores is further increased (i.e., 12 and 24) this reduction is, however, significantly lower. Most likely, this is related to the amount of data that has to be transferred during the computations, which is a relatively slow process (1 Gbit/s) on the considered cluster. Therefore, it is expected that better speed-ups can be obtained when a larger mesh width is considered.

Figure 6.6 shows, again, the CPU time needed to reach convergence with MGRIT for a different number of time steps N_t and spline degree p. In contrast to Figure 6.5, where a mesh width was chosen equal to $h = 2^{-6}$, we consider a mesh width of $h = 2^{-5}$ instead. Doubling the number of time steps roughly doubles the time needed to reach convergence and the deteriorating performance of the Conjugate Gradient method can be observed as well for higher values of p. However, as smaller linear systems of equations are considered, the overall CPU times are lower compared to those presented in Figure 6.5. Furthermore, a higher reduction of the CPU time can be observer when the number of cores is increased from 12 to 24.

In both figures, a strong dependency of the CPU times on the spline degree p is visible, related to the use of the Conjugate Gradient method for all spatial solves. To mitigate this dependency, we will apply our p-multigrid method, as presentented in Chapter 4, for all spatial solves in MGRIT in the next section.

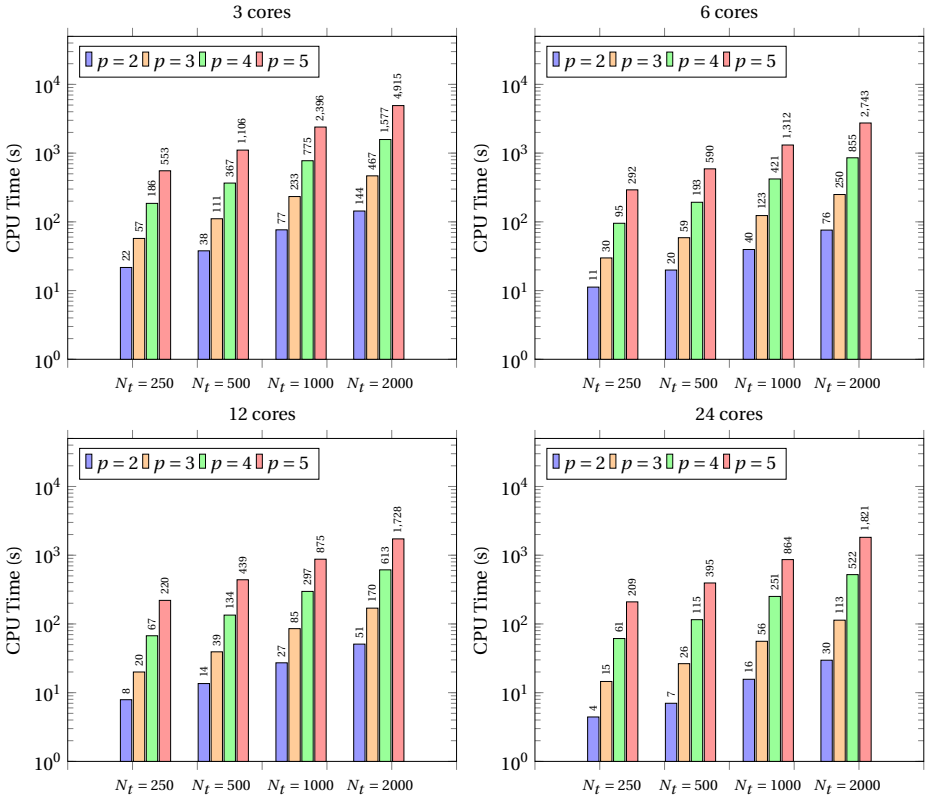


Figure 6.5: CPU times for MGRIT using V-cycles and backward Euler on the unit square for a fixed problem size ($h = 2^{-6}$) adopting a different number of cores.

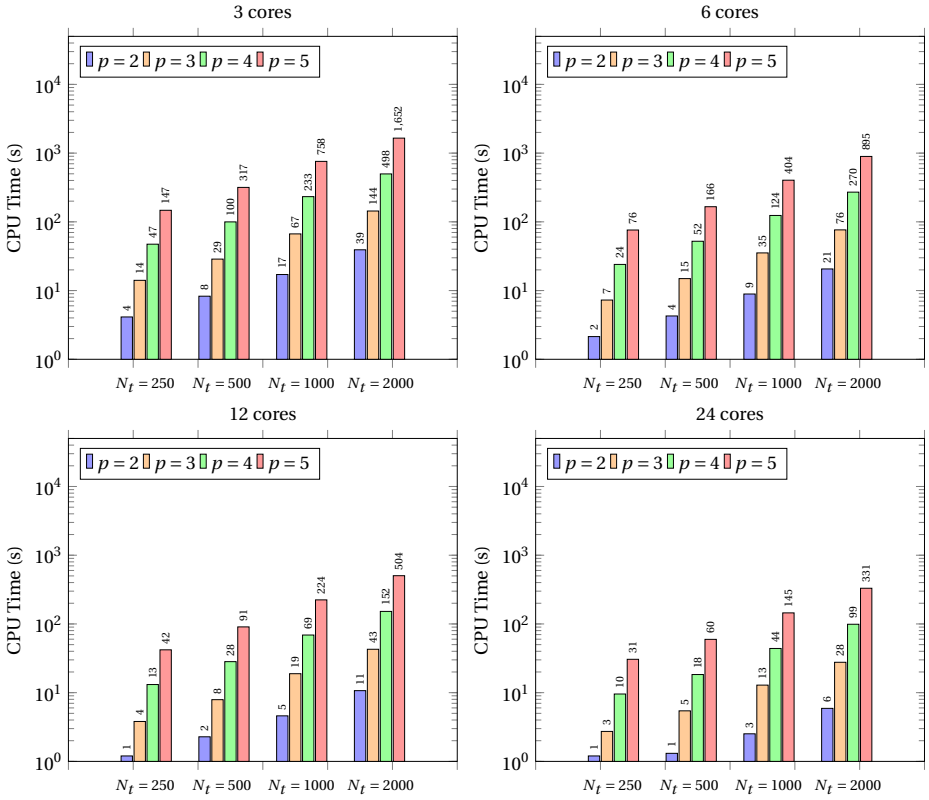


Figure 6.6: CPU times for MGRIT using V-cycles and backward Euler on the unit square for a fixed problem size ($h = 2^{-5}$) adopting a different number of cores.

6.5. NUMERICAL RESULTS (p-MULTIGRID METHOD)

Within the MGRIT method, linear systems of equations have to be solved to account for the fine- and coarse-level time stepping. In the previous section, a standard preconditioned Conjugate Gradient method was adopted leading to a significant increase of the computational costs for higher values of p. Instead of adopting a Conjugate Gradient method, we will adopt our p-multigrid method, as presented in Chapter 4, for the spatial solves within MGRIT throughout the remainder of this section to mitigate this dependency. That is, a direct projection to level $p = 1$ is considered and ILUT is adopted as a smoother. At $p = 1$, a single W-cycle of a standard h-multigrid method is applied to approximately solve the low-order system, where Gauss-Seidel is adopted as a smoother.

ITERATION NUMBERS

As a first step, we compare the number of MGRIT iterations to reach convergence when the p-multigrid method is adopted while keeping all other parameters equal. Table 6.7 shows the results when the mesh width is kept constant ($h = 2^{-6}$) for the unit square and quarter annulus when adopting V-cycles. Results can be compared to Table 6.1 and 6.3. For both benchmarks and all configurations, the number of iterations needed with MGRIT to reach convergence is identical.

	Unit Square				Quarter Annulus			
	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
$N_t = 250$	10	10	10	10	10	10	10	10
$N_t = 500$	10	10	10	10	10	10	10	10
$N_t = 1000$	11	11	11	11	11	11	11	11
$N_t = 2000$	11	11	11	11	11	11	11	11

Table 6.7: Number of MGRIT iterations for solving Equation (6.1) on the unit square and a quarter annulus when adopting V-cycles for a varying number of time steps. Here p-multigrid is adopted for the spatial solves.

Table 6.8 shows the results when the number of time steps is kept constant ($N_t = 100$) for the unit square and quarter annulus when adopting V-cycles. Results can be compared to Table 6.2 and 6.4 and show, again, a similar number of iterations.

	Unit Square				Quarter Annulus			
	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
$h = 2^{-4}$	9	9	9	9	9	9	9	9
$h = 2^{-5}$	9	9	9	9	9	9	9	9
$h = 2^{-6}$	9	9	9	9	9	9	9	9
$h = 2^{-7}$	9	9	9	9	9	9	9	9

Table 6.8: Number of MGRIT iterations for solving Equation (6.1) on the unit square and a quarter annulus when adopting V-cycles for varying mesh widths. Here p-multigrid is adopted for the spatial solves.

CPU TIMINGS

CPU timings have been obtained when a p-multigrid method is adopted for the spatial solves within MGRIT as well. As in the previous section, we adopt V-cycles, a mesh width of $h = 2^{-6}$ and the unit square as our domain of interest. Note that the corresponding iteration numbers can be found in Table 6.7. The computations are performed on three nodes, which consist each of an Intel(R) i7-10700 (@ 2.90GHz) processor with 8 cores.
Figure 6.7 shows the CPU time needed to reach convergence for a varying number of cores, a different number of time steps and different values of p. As with the use of the Conjugate Gradient method, doubling the number of time steps leads to an increase of the CPU time by a factor of two. However, the dependency of the CPU times on the spline degree is significantly mitigated, which leads to a serious decrease of the CPU times compared to the use of the Conjugate Gradient method when higher values of p are considered. Again, increasing the number of cores from 3 to 6, reduces the CPU time needed to reach convergence by $45-50\%$. This does, however, not hold anymore when the number of cores is further increased. Results obtained for a mesh width of $h = 2^{-5}$ are shown in Figure 6.8, showing similar results compared to the ones presented in the previous section when adopting the Conjugate Gradient method.
The results obtained in this and the previous section indicate that MGRIT combined with a p-multigrid method leads to an overall efficient method. However, for an increased number of cores, the observed speed-up significantly decreases on the considered cluster. Therefore, a large computer cluster will be considered in the next section to further investigate the scalability of MGRIT (i.e. weak and strong scalability) when combined with a p-multigrid method.

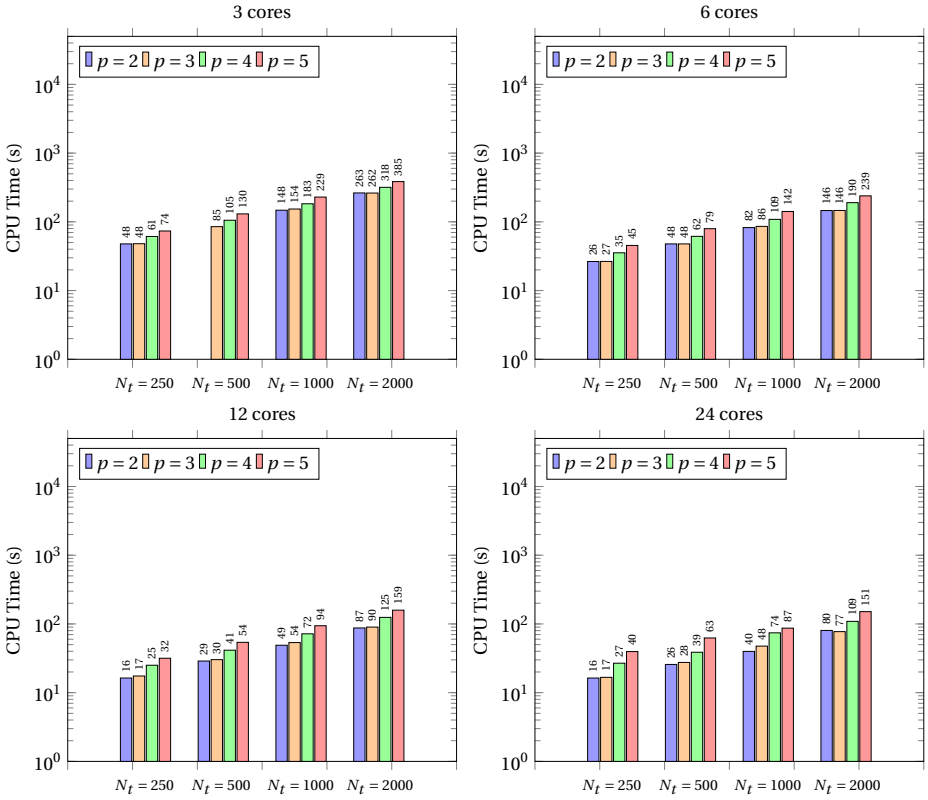


Figure 6.7: CPU times for MGRIT using V-cycles and backward Euler on the unit square for a fixed problem size ($h = 2^{-6}$) adopting a different number of cores. Here p-multigrid is used for all spatial solves within MGRIT.

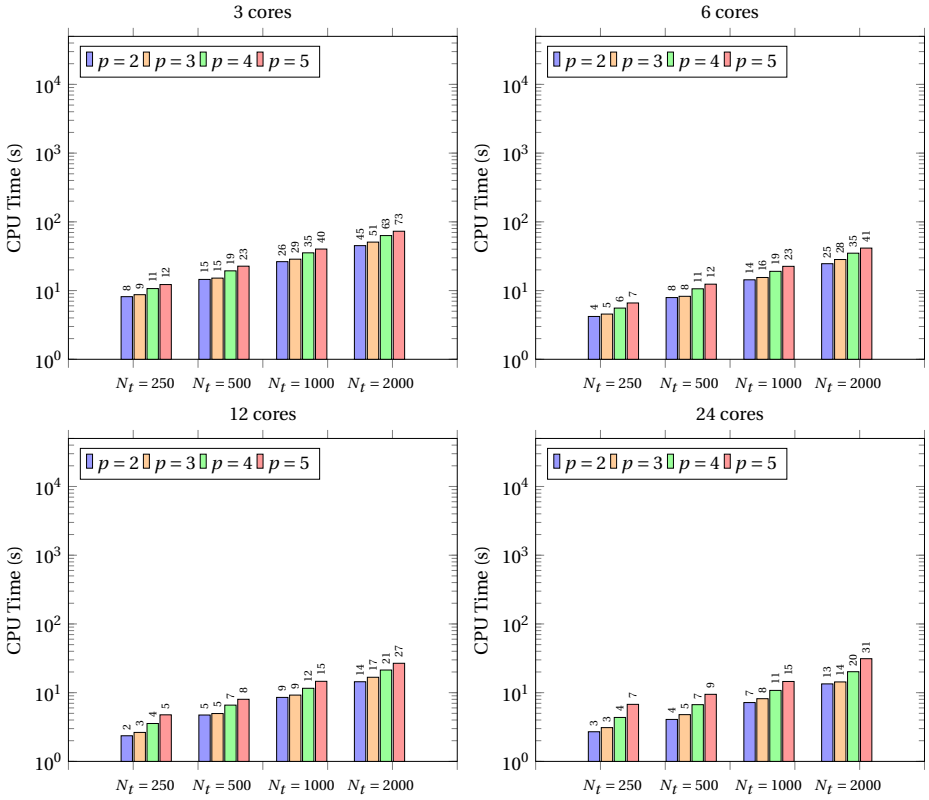


Figure 6.8: CPU times for MGRIT using V-cycles and backward Euler on the unit square for a fixed problem size ($h = 2^{-5}$) adopting a different number of cores. Here p-multigrid is used for all spatial solves within MGRIT.

6.6. SCALABILITY

In the previous sections, we applied MGRIT on a relatively small cluster with up to 24 cores on 3 nodes. Here, it was shown that the use of a p-multigrid method significantly reduces the dependency of the CPU timings on the spline degree. In this section, we investigate the scalability of MGRIT (combined with a p-multigrid method) on a modern architecture. More precisely, we will investigate both strong and weak scalability on the Lisa computer system, one of the nationally used clusters of the Netherlands [1]. Although the general description for each multigrid method is identical, each type of multigrid method is defined by a specific choice of the coarse grid operator, the prolongation and restriction operator and smoother. Throughout this thesis, we will distinguish two main types of multigrid methods: h-multigrid and p-multigrid methods.

First, we fix the total problem size and increase the number of cores (i.e., strong scalability). That is, we consider the same benchmark problem as in the previous sections, but with a mesh width of $h = 2^{-6}$ and a number of time steps N_t of 10.000. As before, backward Euler is applied for the time integration and V-cycles are adopted as MGRIT hierarchy. Figure 6.9 shows the CPU times needed to reach convergence for a varying number of Intel Xeon Gold 6130 (@ 2.10GHz) processors, where each processor consists of 16 cores. For all values of p, increasing the number of processors leads to significant speed-ups which illustrates the parallizability of the MGRIT method. In contrast to the previous section, the speed-up is not limited for a higher number of cores, but we observe scalability up to 2048 cores (128 nodes).

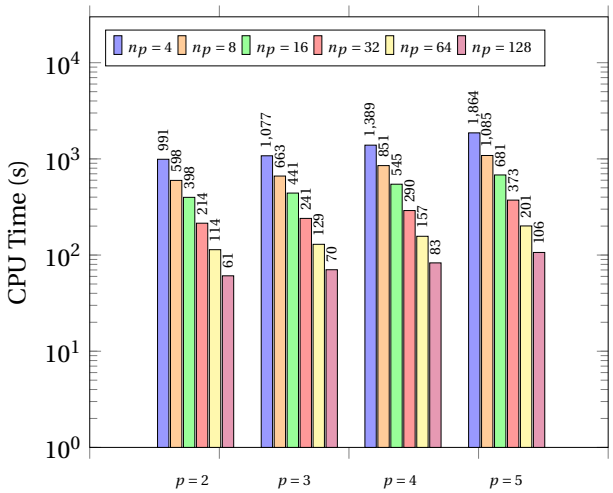


Figure 6.9: Strong scalability study for MGRIT using V-cycles and backward Euler on the unit square. Here p-multigrid is used for all spatial solves within MGRIT.

Figure 6.10 shows the obtained speed-ups as a function of the number of processors for different values of p based on the results presented in Figure 6.9. As a comparison, the

[1] https://userinfo.surfsara.nl/systems/lisa

ideal speed-up has been added, assuming a perfect parallizability of the MGRIT method. The obtained speed-ups remain high, even when the number of processors is further increased to 128, and is independent of the spline degree p.

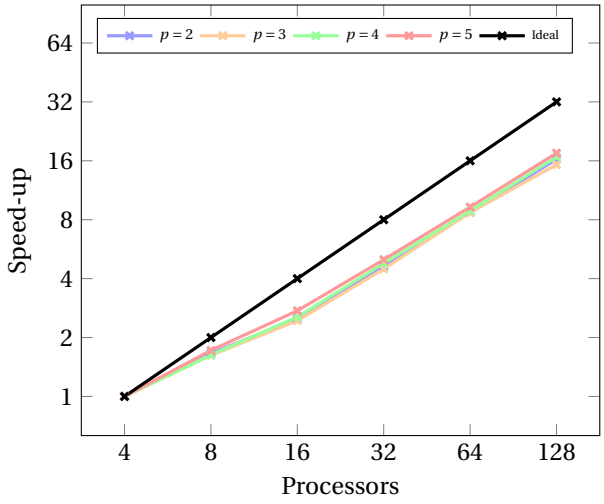


Figure 6.10: Speed-up with MGRIT using V-cycles and backward Euler on the unit square. Here p-multigrid is used for all spatial solves within MGRIT.

As a next step, we consider the same benchmark problem but keep the problem size per processor fixed (i.e., weak scalability). In case of four processors, the number of time steps equals 1000 and is adjusted based on the number of cores. Figure 6.11 shows the CPU time needed to reach convergence for a different number of processors and different values of p. Clearly, the CPU times vary slightly when the number of processors is increased, but overall the MGRIT method shows decent scalability. Although the CPU times slightly increase for higher values of p, the strong p-dependency observed with the Conjugate Gradient method is clearly mitigated.

6.7. CONCLUDING REMARKS

In this chapter, we combined the Multigrid Reduced in Time (MGRIT) method with Isogeometric Analysis to solve time-dependent partial differential equations. In particular, we investigated the convergence for different geometries, time integration schemes and multigrid hierarchies. Results show that the MGRIT method converges independent of the mesh width h, spline degree p and number of time steps N_t. CPU times show that the overall method is highly parallelizable, but the use of standard iterative methods (i.e., the Conjugate Gradient method) leads to a significant dependency of the CPU time on the degree of the B-spline basis functions. Therefore, our p-multigrid method has been adopted to successfully mitigate this dependency. Finally, the scaling properties of the MGRIT method have been determined on a modern architecture, illustrating the scalability of the overall method.

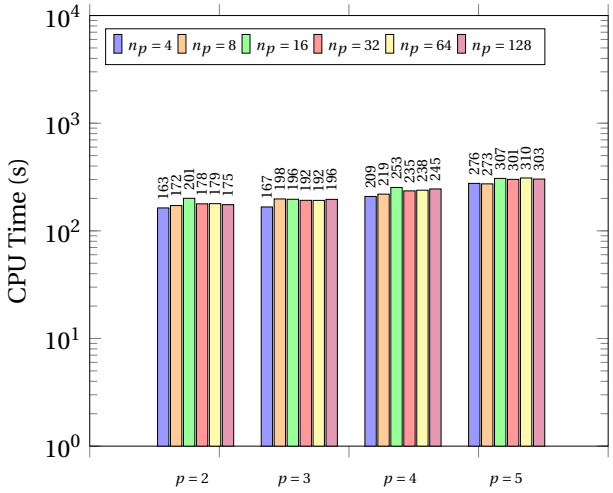


Figure 6.11: Weak scalability study for MGRIT using V-cycles and backward Euler on the unit square. Here p-multigrid is used for all spatial solves within MGRIT.

7

P-MULTIGRID METHODS IN G+SMO

This chapter provides a short introduction to the open-source library G+Smo (Geometry plus Simulation modules), which has been used and extended throughout this thesis. Furthermore, we give detailed instructions on how to reproduce the results presented in this thesis.

7.1. INTRODUCTION

G+Smo (Geometry + Simulation Modules, pronounced "gismo") [101] is an open-source C++ library which has been developed specifically for Isogeometric Analysis and aims to bring together the mathematical tools for geometric design and numerical simulation. The G+Smo library is object-oriented and partitioned into modules (e.g. the PDE discretization or adaptive spline module). All numerical experiments presented in this thesis have been performed using and extending this library.
This chapter provides a short introduction to the G+Smo library in Section 7.2. The p-multigrid class, which was developed during this PhD research is then discussed in Section 7.3. Finally, we describe in Secion 7.4 and 7.5 how the numerical results (i.e., iteration numbers) presented in thesis can be reproduced in G+Smo.

7.2. G+SMO: AN INTRODUCTION

G+Smo makes use of the version control system git, allowing to keep track of changes in the code over time. The latest version of G+Smo can be obtained from GitHub in the following way:

```
$ git clone https://github.com/gismo/gismo.git
```

Here, $ denotes the command prompt. The terminal command above clones the latest revision of the code on your personal device. To compile the library, there are some minimal prerequisites (e.g. a C++ compiler and CMake) which can be found on the GitHub page. Configuring and building, enabling the use of MPI and Xbraid, is done as follows:

```
$ mkdir gismo_build
$ cd gismo_build
$ cmake ../gismo/gismo.git -DGISMO_WITH_MPI=ON -DGISMO_WITH_XBRAID=ON
- Build files have been written to:  home/gismo_build
$ make poisson_example
$ ./bin/poisson_example
```

This builds the G+Smo library in the folder "gismo_build" and executes `poisson_example`. Within this example, the Poisson equation is solved on the unit square, where the right-hand side is chosen such that the exact solution can be expressed in terms of sine functions (i.e. the method of manufactured solutions). After assembling the stiffness matrix and the right-hand side, the resulting linear system of equations is then solved by means of a Conjugate Gradient method.

7.3. THE p-MULTIGRID CLASS

Within G+Smo, a base class (called `pMultigridBase`) has been implemented, consisting of all different aspects of the p-multigrid as described in this thesis. In particular, a generic (recursive) multigrid cycle is implemented in this base class.
Apart from the multigrid cycle, the restriction and prolongation function are implemented as well in the base class, while all other functions are pure virtual. As a consequence, the remaining functions have to be implemented explicitly in the derived `pMultigrid` class. This derived class has three template arguments:

- The data type considered, typically a real number (i.e., `real_t`).
- The solver to be applied at the coarsest level (e.g. `gsSparseSolver<real_t>::LU`).
- The assembler adopted (e.g. `gsCDRAssembler<real_t>`).

Apart from these template arguments, the `pMultigrid` class is defined by three arguments, namely a multipatch object `gsMultiPatch<T>` "mp", a low-order basis `gsMultiBasis<T>` "basisL" and the boundary conditions `gsBoundaryConditions<T>` "bcInfo". Hence, a p-multigrid structure can be declared as follows:

```
pMultigrid<T, Coarsesolver , Assembler> My_MG(mp, basisL, bcInfo);
```

The derived p-multigrid structure has two important functions:

- **setup**: Here, all operators are assembled and the smoother is set up.
- **solve**: This functions solves the model problem using the base class.

The `pMultigrid` class can be used to solve a given model problem by calling the **setup** and **solve** functions. Next to those two functions, all pure virtual functions (restriction, prolongation etc.) from the base class still have to implemented in the `pMultigrid` class. Of course, the `pMultigrid_example` already contains versions of the prolongation and resitriction functions as presented in this dissertation.
From all arguments the **solve** and **setup** require, we present the most important ones. In the following table, we provide the names of the arguments, the flags that can be used and a short description. Here, we distinguish between parameters that describe or define the model problem to be solved and the arguments that define the multigrid hierarchy. More information on these parameters can be found in the example folder of G+Smo in the `pMultigrid_example` file.
All geometries considered in this thesis, apart from the Yeti footprint, are based on the class `gsNurbsCreator`, which provides the NURBS geometry for each benchmark problem. In particular, the specific geometry function for each geometry can be found in the source code of this class.

p-multigrid parameters

numBenchmark	b	Adopt as a benchmark: • CDR-equation, unit square (-b 1) • Poisson's equation, quarter annulus (-b 2) • Poisson's equation, quarter annulus (-b 3) • Poisson's equation, L-shaped domain (-b 4) • Poisson's equation, unit cube (-b 5) • Poisson's equation, Yeti footprint (-b 6)
typeSolver	s	Apply *p*-multigrid: • as a stand-alone solver (-s 1) • within a Conjugate Gradient method (-s 2) • within a Bi-CGSTAB method (-s 3)
typeSmoother	S	Adopt as a smoother: • Gauss-Seidel (-S 1) • ILUT (-S 2) • SCMS (-S 3) • block ILUT (-S 4)
typeProjection	D	Consider: • a direct projection to $p = 1$ (-D 1) • an indirect projection (-D 2)
typeLumping	L	Adopt for the prolongation/restriction: • a lumped L_2 projection (-L 1) • a consistent L_2 projection (-L 2)
typeBChandling	d	Apply the boundary conditions: • by eliminating the DOFs (-d 1) • by means of Nitsche's method (-d 2)
numSmoothing	v	Number of pre- and postsmoothing steps
numLevels	l	Degree of levels in the multigrid hierarchy

7.4. pMULTIGRID_EXAMPLE

In this section, we discuss how results obtained in this thesis (with respect to iteration numbers) can be obtained by any G+Smo user as well. To obtain the code adopted in this dissertation, a permalink [1] can be used which leads directly to the source code.

This version of the code contains the `pMultigrid_example` within the examples folder and can reproduce results from this thesis regarding iteration numbers. As an example, one can run the following code in the terminal:

```
$ ./bin/pMultigrid_example -p 2 -r 6 -l 4 -S 1 -b 2 -z "hhp"
```

Here, the parameter `-z` provides the coarsening strategy by entering a string of length $l-1$ which defines if p-coarsening ("p") or h-coarsening ("h") is applied. The above code leads to the following terminal output:

```
|| Benchmark information ||
Poisson equation on the quarter annulus (1)
Exact solution:  -(x*x+y*y-1)*(x*x+y*y-4)*x*y*y
Right hand side solution:  2*x*(22*x*x*y*y+21*y*y*y*y-45*y*y+x*x*x*x-5*x*x+4)
Number of patches:  1

p-multigrid is applied as stand-alone solver

|| Multigrid hierarchy ||
Level 1, Degree:  1, Ndof:  306
Level 1, Degree:  1, Ndof:  1122
Level 1, Degree:  1, Ndof:  4290
Level 1, Degree:  2, Ndof:  4422

|| Setup timings ||
Total Assembly time:  0.253446
Total ILUT factorization time:  0.0677094
Total block ILUT factorization time:  2.38419e-07
Total SCMS time:  0
Total setup time:  0.405733

|| Solver information ||
Solver converged in 0.0217087 seconds!
Solver converged in 4 iterations!
Residual after solving:  2.42863e-05
```

[1] https://archive.softwareheritage.org/swh:1:dir:d3c61b86c91a9f031dda3b9bcbbbcff0ca80a821;origin=https://github.com/roeltielen24/gismo_pmultigrid_xbraid;visit=swh:1:snp:5e1a54efb88fa0bf85aa4efc012192862a981d0d;anchor=swh:1:rev:523519f0e0691d576d8581e243905d84e241e14d

That is, Poisson's equation is solved on a quarter annulus using a p-multigrid method combined with an ILUT smoother. Note that the output in the terminal describes the benchmark problem, the p-multigrid hierarchy and details about the solver. By typing

```
$ ./bin/pMultigrid_example -p <INT> -r <INT> -l <INT> -S <INT> -b 2 -z <STRING>
```

one can obtain the results for different discretizations. Table 7.1 shows the results for different values of p and h which can be reproduced by varying the different parameters in the terminal. The red colored 4 is the result corresponding to the output above.

	$p=2$		$p=3$		$p=4$		$p=5$	
	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	61	3	163	3	473
$h=2^{-9}$	5	32	3	61	3	163	3	452

Table 7.1: Number of multigrid cycles needed to achieve convergence with p-multigrid using Gauss-Seidel (GS) and ILUT as a smoother.

7.5. XBRAID_HEATEQUATION_EXAMPLE

In a similar way, results obtained with MGRIT can be obtained as well. The corresponding source file `xbraid_heatEquation_example` can be found in the following way:

```
$ cd extensions/gsXbraid/examples
```

One can then run the following code in the terminal:

```
$ mpirun -np 4 ./bin/xbraid_heatEquation_example -n 100 -i 2 -r 4
```

This command runs the MGRIT method (using 4 processors) adopting 100 time steps, a mesh width of $h=2^{-4}$ and spline degree equal to 2. The obtained output provides all the details with respect to the settings of the MGRIT method and the p-multigrid method. A part of the output shows the convergence of the MGRIT method:

```
Braid:  Begin simulation, 100 time steps
Braid:  || r_0 || not available, wall time = 5.56e-02
Braid:  || r_1 || = 1.572381e-01, conv factor = 1.00e+00, wall time = 1.61e-01
Braid:  || r_2 || = 5.147275e-03, conv factor = 3.27e-02, wall time = 3.31e-01
Braid:  || r_3 || = 3.185626e-04, conv factor = 6.19e-02, wall time = 5.56e-01
Braid:  || r_4 || = 2.021004e-05, conv factor = 6.34e-02, wall time = 7.40e-01
Braid:  || r_5 || = 1.209229e-06, conv factor = 5.98e-02, wall time = 8.87e-01
Braid:  || r_6 || = 7.119758e-08, conv factor = 5.89e-02, wall time = 1.04e+00
Braid:  || r_7 || = 4.077797e-09, conv factor = 5.73e-02, wall time = 1.18e+00
```

```
Braid:  || r_8 || = 2.237164e-10, conv factor = 5.49e-02, wall time = 1.32e+00
Braid:  || r_9 || = 1.143291e-11, conv factor = 5.11e-02, wall time = 1.47e+00
```

By typing

```
$ mpirun -np 4 ./bin/xbraid_heatEquation_example -n <INT> -i <INT> -r <INT>
```

one can obtain the results for different discretizations. It should be noted that this example makes use of an XML-file (heat2d_square_ibvp1.xml) which provides all the MGRIT and p-multigrid settings and should be adjusted accordingly. Table 7.2 shows the results for different values of p and h which can be reproduced in this example. Here, the red colored 9 is the result corresponding to the output above.

	Unit Square				Quarter Annulus			
	$p=2$	$p=3$	$p=4$	$p=5$	$p=2$	$p=3$	$p=4$	$p=5$
$h=2^{-4}$	9	9	9	9	9	9	9	9
$h=2^{-5}$	9	9	9	9	9	9	9	9
$h=2^{-6}$	9	9	9	9	9	9	9	9
$h=2^{-7}$	9	9	9	9	9	9	9	9

Table 7.2: Number of MGRIT iterations for solving Equation (6.1) on the unit square and a quarter annulus when adopting V-cycles for varying mesh widths. Here p-multigrid is adopted for the spatial solves.

8

CONCLUSIONS AND FUTURE WORK

In this thesis, we presented a p-multigrid method to solve linear systems of equations arising in Isogeometric Analysis. The use of p-multigrid methods was motivated by the fact that low-order (i.e., $p = 1$) discretizations in IgA coincide with standard P_1 Lagrange finite elements. Therefore, well-established solution techniques can be adopted from standard FEM to obtain the coarse grid correction at the low-order level. Furthermore, the smaller support of P_1 basis functions leads to less non-zero entries in the system matrices, resulting in lower assembly costs and memory requirements compared to h-multigrid methods for certain configurations when applied within Isogeometric Analysis.

In practice, a single W-cycle of a standard h-multigrid method using Gauss-Seidel as a smoother is adopted to solve the residual equation at level $p = 1$. To project residuals and corrections from the high-order level onto the low-order level (and vice versa), a row-sum lumped L_2 projection is used. Here, the row-sum lumping prevents the explicit solution of a linear system of equations at each projection step, while remaining accurate enough to have an overall efficient numerical method. It should be noted that from the high-order level a direct projection to level $p = 1$ is adopted, which has shown to be more efficient in terms of computational times, while keeping (more or less) the same number of iterations compared to an indirect projection, where all p levels are traversed.

For single patch geometries, we applied the proposed p-multigrid method for a variety of two- and three-dimensional benchmark problems. Here, both Gauss-Seidel and ILUT have been applied as a smoother at the high-order level. As with h-multigrid methods, the use of Gauss-Seidel leads to h-independent, but p-dependent convergence rates of the resulting p-multigrid method. When ILUT is applied as a smoother, however, the number of iterations needed to achieve convergence is independent of both h and p. Compared to h-multigrid methods, the observed convergence rates with p-multigrid methods are comparable for all smoothers. Finally, we compared the p-multigrid method (using ILUT as a smoother) with an h-multigrid method adopting a subspace corrected mass smoother and applied it to THB-spline based discretizations.

For multipatch geometries, the dependency of the convergence rate of the p-multigrid

method on the number of patches has been investigated. For sufficiently fine meshes, the number of iterations is constant in the number of patches, while the number of iterations slightly grows for a higher number of patches on coarser meshes. Although the use of an outer Krylov solver reduces the number of iterations needed to achieve convergence, the use of p-multigrid method as a stand-alone solver might be more benificial when considering the total computational time. As a second improvement, we considered the use of a block ILUT smoother that makes use of the typical block structure of the resulting system matrix in case of a multipatch geometry. The block ILUT smoother can be set up (and applied) in parallel and shows an improved convergence rate compared to the use of a global ILUT smoother.
In Chapters 4 and 5, the focus lied on time-independent benchmark problems. For time-dependent bechmark problems, a time-integration scheme is typically adopted for the temporal discretization. A drawback of this approach is, however, the sequential nature of these schemes. Therefore, we adopted a Multigrid Reduced in Time (MGRIT) method in the context of Isogeometric Analysis for the first time in the literature, which allows us to parallelize the time integration. The spatial solves arising in the MGRIT method have been solved by the proposed p-multigrid method. Finally, we described how the numerical results presented in this thesis can be (easily) reproduced using the open-source C++ library G+Smo (Geometry + Simulation modules).

These findings lead to the following conclusions of this dissertation:

- p-multigrid methods enhanced with an ILUT smoother are robust in the spline degree p and the mesh width h. Moreover, combined with an h-multigrid method at level $p = 1$, they are competitive in terms of computational efficiency to state-of-the-art multigrid methods for Isogeometric Analysis.
- The use of a block ILUT smoother within the p-multigrid method is an effective approach for multipatch geometries in Isogeometric Analysis and leads to better convergence rates compared to the use of a global ILUT smoother.
- Multigrid Reduced in Time (MGRIT) methods have been successfully combined with p-multigrid methods to obtain a scalable and robust solver for Isogeometric Analysis.

FUTURE WORK

In the future, further progress can be made regarding the algorithmic design of the p-multigrid method, the application of p-multigrid methods to more challenging benchmarks and the theoretical insight on the effectiveness of p-multigrid methods. For each of these categories, we briefly describe potential directions for future research.

ALGORITHMIC IMPROVEMENTS

- Currently, we apply a direct projection from the high-order level to level $p = 1$, after which a standard h-multigrid method is applied. That is, we first coarsen (aggressively) in p, after which successive h-coarsening is applied. Alternatively, one can

combine both h- and p-coarsening to obtain a coarser system before applying the h-multigrid method, thereby reducing the computational costs of the p-multigrid method. In fact, this procedure has been persued in a recent preprint [135], where a local Fourier analysis (LFA) showed that this indeed improves the computational efficiency of the multigrid method in case a multiplicative Schwarz smoother is adopted.

- In this thesis, we considered Gauss-Seidel, ILUT and the subspace corrected mass smoother (SCMS) as smoothers within the p-multigrid methods. However, a variety of smoothers have been developed for h-multigrid methods in recent years. In particular, we suggest to consider the smoother presented in [84, 135], as it has been applied in a variant of the p-multigrid method presented in this thesis.

- We applied the p-multigrid method presented in this thesis to THB-spline discretizations in Chapter 4, where smoothing was applied globally, making the overall complexity suboptimal. Alternatively, local smoothing can be pursued [108] to ensure optimal order of complexity. Therefore, we suggest to consider local smoothing strategies when the p-multigrid method is applied in the context of THB-spline discretizations.

- In Chapter 5 of this thesis, we presented a block ILUT smoother for multipatch geometries, suited for parallel computations. In future work, we see high potential in implementing this smoother explicitly in parallel to obtain a parallel p-multigrid method. Potentially, this approach can be combined with MGRIT, to obtain a full parallel method (in space and time).

Applications

- The p-multigrid method presented in this thesis has been applied to a variety of benchmark problems in two and three dimensions. For these benchmarks, the solution is found in a subspace of the space $H^1(\Omega)$, implying C^0 continuity of the basis functions is sufficiently smooth. For PDEs involving fourth-order derivatives (e.g. the biharmonic equation), as often encountered in engineering, the solution should be at least C^1, implying that a low-order correction found at level $p = 1$ is not sufficiently smooth. Therefore, for these type of benchmark problems, a p-multigrid where the low-order level corresponds to $p = 2$ would be worthwhile investigating.

- Typically, multigrid methods are applied on diffusion-dominated (i.e., Poisson-type) problems. However, it might be interesting to investigate to which extend the p-multigrid method can be applied to more convection-dominated problems. In fact, the p-multigrid method presented in this thesis has already been applied successfully to projection-type problems within the Material Point Method (MPM), see [136].

- Recently, the use of Isogeometric Analysis to discretize the Helmholtz equation has shown to significantly reduce the pollution error. Iterative solvers to solve the resulting linear systems of equations have obtained increased interest in recent years

[137]. In particular, the use of deflation combined with the approximated Complex Shifted Laplacian Preconditioner (CSLP) has shown to lead to wave-number independent convergence rates [138]. Here, the CSLP is approximated by a fixed number of V-cycles using a standard h-multigrid method adopting (damped) Jacobi as a smoother. The use of p-multigrid methods combined with a suitable smoother could further improve the convergence rates, in particular for higher values of p.

THEORETICAL INSIGHT

- Throughout this thesis, we analyzed the spectral properties of the p-multigrid method by determining the reduction factors of the coarse grid correction and smoother on the generalized eigenvectors. Furthermore, the spectrum of the iteration matrix was obtained numerically to determine the asymptotic convergence rate and investigate to which extend the use of an outer Krylov solver would further improve the overall convergence. Alternatively, a local Fourier analysis (LFA) can be performed to analyze the (expected) performance of a multigrid method. In [135], a LFA has been applied to a variant of the p-multigrid method presented in this thesis.

- In this thesis, numerical experiments are used to show the effectiveness, robustness and efficiency of the proposed p-multigrid method. However, mathematical theory on p-multigrid methods is non trivial, in particular when rediscretization is applied to obtain the coarse grid operators. Future research could focus on a theoretical foundation to quantify the effectiveness of p-multigrid methods in the context of Isogeometric Analysis.

REFERENCES

[1] T. Hughes, J. Cottrell, and Y. Bazilevs, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement,* Computer Methods in Applied Mechanics and Engineering **194**, 4135 (2005).

[2] J. Cottrell, A. Reali, Y. Bazilevs, and T. Hughes, *Isogeometric analysis of structural vibrations,* Computer Methods in Applied Mechanics and Engineering **195(41-43)**, 5257 (2006).

[3] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner, *Isogeometric shell analysis with Kirchhoff-Love elements,* Computer Methods in Applied Mechanics and Engineering **198**, 3902 (2009).

[4] J. Kiendl, Y. Bazilevs, M.-C. Hsu, R. Wüchner, and K.-U. Bletzinger, *The bending strip method for isogeometric analysis of Kirchhoff-Love shell structures comprised of multiple patches,* Computer Methods in Applied Mechanics and Engineering **199**, 2403 (2010).

[5] D. Benson, Y. Bazilevs, M.-C. Hsu, and T. Hughes, *Isogeometric shell analysis: The Reissner-Mindlin shell,* Computer Methods in Applied Mechanics and Engineering **199**, 276 (2010).

[6] N. Nguyen-Thanh, J. Kiendl, H. Nguyen-Xuan, R. Wüchner, K.-U. Bletzinger, Y. Bazilevs, and T. Rabczuk, *Rotation free isogeometric thin shell analysis using PHT-splines,* Computer Methods in Applied Mechanics and Engineering **200**, 3410 (2011).

[7] W. Dornisch, S. Klinkel, and B. Simeon, *Isogeometric Reissner-Mindlin shell analysis with exactly calculated director vectors,* Computer Methods in Applied Mechanics and Engineering **253**, 491 (2013).

[8] J. Kiendl, M.-C. Hsu, M. Wu, and A. Reali, *Isogeometric Kirchhoff-Love shell formulations for general hyperelastic materials,* Computer Methods in Applied Mechanics and Engineering **291**, 280 (2015).

[9] Y. Bazilevs, V. Calo, Y. Zhang, and T. J. R. Hughes, *Isogeometric fluid–structure interaction analysis with applications to arterial blood flow,* Computational Mechanics **38**, 310 (2006).

[10] G. Moutsanidis, C. C. Long, and Y. Bazilevs, *IGA-MPM: The isogeometric material point method,* Computer Methods in Applied Mechanics and Engineering **372**, 113346 (2020).

[11] Y. Gan, Z. Sun, Z. Chen, X. Zhang, and Y. Liu, *Enhancement of the material point method using B-spline basis functions,* International Journal for Numerical Methods in Engineering **113**, 411 (2017).

[12] R. Tielen, E. Wobbes, M. Möller, and L. Beuth, *A high order material point method,* Procedia Engineering **175**, 265 (2017).

[13] Y. G. Motlagh and W. M. Coombs, *An implicit high-order material point method,* Procedia Engineering **175**, 8 (2017).

[14] M. Steffen, R. M. Kirby, and M. Berzins, *Analysis and reduction of quadrature errors in the material point method (MPM),* International Journal for Numerical Methods in Engineering **76**, 922 (2008).

[15] M. Bucelli, M. Salvador, L. Dede', and A. Quarteroni, *Multipatch isogeometric analysis for electrophysiology: Simulation in a human heart,* Computer Methods in Applied Mechanics and Engineering **376** (2021).

[16] W. A. Wall, M. A. Frenzel, and C. Cyron, *Isogeometric structural shape optimization,* Computer Methods in Applied Mechanics and Engineering **197**, 2976 (2008).

[17] X. Qian, *Full analytical sensitivities in NURBS based isogeometric shape optimization,* Computer Methods in Applied Mechanics and Engineering **199**, 2059 (2010).

[18] Y.-D. Seo, H.-J. Kim, and S.-K. Youn, *Shape optimization and its extension to topological design based on isogeometric analysis,* International Journal of Solids and Structures **47**, 1618 (2010).

[19] K. Li and X. Qian, *Isogeometric analysis and shape optimization via boundary integral,* Computer-Aided Design **43**, 1427 (2011).

[20] D. M. Nguyen, A. Evgrafov, and J. Gravesen, *Isogeometric shape optimization for electromagnetic scattering problems,* Progress In Electromagnetics Research B **45**, 117 (2012).

[21] L. Blanchard, R. Duvigneau, A.-V. Vuong, and B. Simeon, *Shape gradient for isogeometric structural design,* Journal of Optimization Theory and Applications **161**, 361 (2014).

[22] P. Bornemann and F. Cirak, *A subdivision-based implementation of the hierarchical B-spline finite element method,* Computer Methods in Applied Mechanics and Engineering **253**, 584 (2013).

[23] D. Fußeder, B. Simeon, and A.-V. Vuong, *Fundamental aspects of shape optimization in the context of isogeometric analysis,* Computer Methods in Applied Mechanics and Engineering **286**, 313 (2015).

[24] L. Beirão da Veiga, D. Cho, L. Pavarino, and S. Scacchi, *Overlapping Schwarz methods for isogeometric analysis,* SIAM Journal on Numerical Analysis **50**, 1394 (2012).

[25] L. Beirão da Veiga, D. Cho, L. Pavarino, and S. Scacchi, *BDDC preconditioners for isogeometric analysis,* Mathematical Models and Methods in Applied Sciences **23**, 1099 (2013).

[26] L. Beirão da Veiga, D. Cho, L. Pavarino, and S. Scacchi, *Isogeometric Schwarz preconditioners for linear elasticity systems,* Computer Methods in Applied Mechanics and Engineering **253**, 439 (2013).

[27] N. Collier, L. Dalcin, and V. Calo, *On the computational efficiency of isogeometric methods for smooth elliptic problems using direct solvers,* International Journal for Numerical Methods in Engineering **100**, 620 (2014).

[28] M. Bercovier and I. Soloveichik, *Overlapping non matching meshes domain decomposition method in isogeometric analysis,* (2015), arXiv:1502.03756 [math.NA] .

[29] G. Sangalli and M. Tani, *Isogeometric preconditioners based on fast solvers for the Sylvester equation,* SIAM Journal on Scientific Computing **38**, 3644 (2016).

[30] D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, N. Collier, and V. Calo, *The value of continuity: Refined isogeometric analysis and fast direct solvers,* Computer Methods in Applied Mechanics and Engineering **316**, 586 (2017).

[31] D. Cho and R. Vázquez, *BPX preconditioners for isogeometric analysis using analysis-suitable t-splines,* IMA Journal of Numerical Analysis **40**, 764 (2018).

[32] G. Loli, M. Montardini, G. Sangalli, and M. Tani, *An efficient solver for space–time isogeometric Galerkin methods for parabolic problems,* Computers & Mathematics with Applications **80**, 2586 (2020).

[33] D. Cho, *Overlapping Schwarz methods for isogeometric analysis based on generalized B-splines,* Computer Methods in Applied Mechanics and Engineering **372**, 113430 (2020).

[34] D. Cho, *Optimal multilevel preconditioners for isogeometric collocation methods,* Mathematics and Computers in Simulation **168**, 76 (2020).

[35] M. Bosy, M. Montardini, G. Sangalli, and M. Tani, *A domain decomposition method for isogeometric multi-patch problems with inexact local solvers,* Computers & Mathematics with Applications **80**, 2604 (2020).

[36] V. Calo, M. Łoś, Q. Deng, I. Muga, and M. Paszyński, *Isogeometric residual minimization method (iGRM) with direction splitting preconditioner for stationary advection-dominated diffusion problems,* Computer Methods in Applied Mechanics and Engineering **373**, 113214 (2021).

[37] G. Loli, G. Sangalli, and M. Tani, *Easy and efficient preconditioning of the isogeometric mass matrix,* Computers & Mathematics with Applications (2021).

[38] H. Horníková, C. Vuik, and J. Egermaier, *A comparison of block preconditioners for isogeometric analysis discretizations of the incompressible Navier-Stokes equations,* International Journal for Numerical Methods in Fluids (2021).

[39] A. Brandt, *Multi-level adaptive solutions to boundary-value problems,* Mathematics of Computation **31**, 333 (1977).

[40] A. Brandt, *Guide to multigrid development,* in *Multigrid Methods*, edited by W. Hackbusch and U. Trottenberg (Springer Berlin Heidelberg, Berlin, Heidelberg, 1982) pp. 220–312.

[41] A. Brandt, S. McCormick, and J. Ruge, *Multigrid methods for differential eigenproblems,* SIAM Journal on Scientific and Statistical Computing **4**, 244 (1983).

[42] A. Brandt and O. E. Livne, *Multigrid Techniques* (Society for Industrial and Applied Mathematics, 2011).

[43] D. Braess and W. Hackbusch, *A new convergence proof for the multigrid method including the v-cycle,* SIAM journal on numerical analysis **20**, 967 (1983).

[44] W. Hackbusch, *On the multi-grid method applied to difference equations,* Computing **20**, 291 (1978).

[45] W. Hackbusch and H. D. Mittelmann, *On multi-grid methods for variational inequalities,* Numerische Mathematik **42**, 65 (1983).

[46] W. Hackbush, *Multi-grid methods and applications* (Springer, Berlin, Germany, 1985).

[47] K. Stüben and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and applications,* in *Multigrid Methods*, edited by W. Hackbusch and U. Trottenberg (Springer Berlin Heidelberg, Berlin, Heidelberg, 1982) pp. 1–176.

[48] S. R. Fulton, P. E. Ciesielski, and W. H. Schubert, *Multigrid methods for elliptic problems: A review,* Monthly Weather Review **114**, 943 (1986).

[49] K. Fidkowski, J. L. T.A. Oliver, and D. Darmofal, *p-multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations,* Journal of Computational Physics **207**, 92 (2005).

[50] H. Luo, J. Baum, and R. Löhner, *A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids,* Journal of Computational Physics **211**, 767 (2006).

[51] H. Luo, J. Baum, and R. Löhner, *Fast p-multigrid discontinuous Galerkin method for compressible flows at all speeds,* AIAA Journal **46**, 635 (2008).

[52] B. Helenbrook, D. Mavriplis, and H. Atkins, *Analysis of p-multigrid for continuous and discontinuous finite element discretizations,* in *AIAA Computational FLuid Dynamics Conference* (2003).

[53] C. Liang, R. Kannan, and Z. Wang, *A p-multigrid spectral difference method with explicit and implicit smoothers on unstructured triangular grids,* Computers & Fluids **38**, 254 (2009).

[54] F. Bassi, A. Ghidoni, S. Rebay, and P. Tesini, *High order accurate p-multigrid discontinuous Galerkin solution of the Euler equations,* International Journal for Numerical Methods in Fluids **60**, 847 (2009).

[55] S. Premasuthan, C. Liang, A. Jameson, and Z. Wang, *A p-multigrid spectral difference method for viscous compressible flow using 2d quadrilateral meshes,* in *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition* (2009) p. 950.

[56] V. Dolean, R. Pasquetti, and F. Rapetti, *p-multigrid for Fekete spectral element method,* in *Domain Decomposition Methods in Science and Engineering XVII*, edited by U. Langer, M. Discacciati, D. Keyes, O. Widlund, and W. Zulehner (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008) pp. 485–492.

[57] C. Liang, A. Chan, and A. Jameson, *A p-multigrid spectral difference method for two-dimensional unsteady incompressible Navier-Stokes equations,* Computers & fluids **51**, 127 (2011).

[58] D. Weber, J. Mueller-Roemer, C. Altenhofen, A. Stork, and D. Fellner, *Deformation simulation using cubic finite elements and efficient p-multigrid methods,* Computers & Graphics **53**, 185 (2015).

[59] P. van Slingerland and C. Vuik, *Fast linear solver for diffusion problems with applications to pressure computation in layered domains,* Computational Geosciences **18**, 343 (2014).

[60] A. M. Rueda-Ramírez, J. Manzanero, E. Ferrer, G. Rubio, and E. Valero, *A p-multigrid strategy with anisotropic p-adaptation based on truncation errors for high-order discontinuous Galerkin methods,* Journal of Computational Physics **378**, 209 (2019).

[61] M. Franciolini, L. Botti, A. Colombo, and A. Crivellini, *p-multigrid matrix-free discontinuous Galerkin solution strategies for the under-resolved simulation of incompressible turbulent flows,* Computers & Fluids **206**, 104558 (2020).

[62] C. R. Nastase and D. J. Mavriplis, *High-order discontinuous Galerkin methods using an hp-multigrid approach,* Journal of Computational Physics **213**, 330 (2006).

[63] P. F. Antonietti, M. Sarti, and M. Verani, *Multigrid algorithms for hp-discontinuous Galerkin discretizations of elliptic problems,* SIAM Journal on Numerical Analysis **53**, 598 (2015).

[64] W. F. Mitchell, *The hp-multigrid method applied to hp-adaptive refinement of triangular grids,* Numerical Linear Algebra with Applications **17**, 211 (2010).

[65] J. van der Vegt and S. Rhebergen, *hp-multigrid as smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows: Part I. multilevel analysis,* Journal of computational physics **231**, 7537 (2012).

[66] J. van der Vegt and S. Rhebergen, *hp-multigrid as smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows: Part II. optimization of the Runge-Kutta smoother,* Journal of Computational Physics **231**, 7537 (2012).

[67] T. Hughes, A. Reali, and G. Sangalli, *Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: Comparison of p-method finite elements with k-method NURBS,* Computer Methods in Applied Mechanics and Engineering **197**, 4104 (2008).

[68] C. de Boor, *A practical guide to splines* (Springer-Verlag, New York, USA, 1978).

[69] F. Calabro, G. Sangalli, and M. Tani, *Fast formation of isogeometric Galerkin matrices by weighted quadrature,* Computer Methods in Applied Mechanics and Engineering **316**, 606 (2017).

[70] P. Antolin, A. Buffa, F. Calabrò, M. Martinelli, and G. Sangalli, *Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization,* Computer Methods in Applied Mechanics and Engineering **285**, 817 (2015).

[71] A. Mantzaflaris, B. Jüttler, B. N. Khoromskij, and U. Langer, *Low rank tensor methods in Galerkin-based isogeometric analysis,* Computer Methods in Applied Mechanics and Engineering **316**, 1062 (2017).

[72] D. Drzisga, B. Keith, and B. Wohlmuth, *The surrogate matrix methodology: Accelerating isogeometric analysis of waves,* Computer Methods in Applied Mechanics and Engineering **372** (2020).

[73] D. Drzisga, B. Keith, and B. Wohlmuth, *The surrogate matrix methodology: A reference implementation for low-cost assembly in isogeometric analysis,* MethodsX **7** (2020).

[74] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems,* Journal of Research of the National Bureau of Standards **49(6)**, 409 (1952).

[75] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems,* SIAM Journal on Scientific and Statistical Computing **13**, 631 (1992).

[76] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems,* SIAM Journal on Scientific and Statistical Computing **7**, 856 (1986).

[77] P. Sonneveld and M. B. van Gijzen, *IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations,* SIAM Journal on Scientific Computing **31**, 1035 (2009).

[78] K. Gahalaut, J. Kraus, and S. Tomar, *Multigrid methods for isogeometric discretizations,* Computer Methods in Applied Mechanics and Engineering **253**, 413 (2013).

[79] C. Hofreither and W. Zulehner, *Spectral analysis of geometric multigrid methods for isogeometric analysis,* Numerical Methods and Applications **8962**, 123 (2015).

[80] C. Hofreither and W. Zulehner, *On full multigrid schemes for isogeometric analysis,* in *Lecture Notes in Computational Science and Engineering* (Springer International Publishing, 2016) pp. 267–274.

[81] C. Hofreither, S. Takacs, and W. Zulehner, *A robust multigrid method for isogeometric analysis in two dimensions using boundary correction,* Computer Methods in Applied Mechanics and Engineering **316**, 22 (2017).

[82] C. Hofreither and S. Takacs, *Robust multigrid for isogeometric analysis based on stable splittings of spline spaces,* SIAM Journal on Numerical Analysis **4**, 2004 (2017).

[83] A. de la Riva, C. Rodrigo, and F. Gaspar, *A robust multigrid solver for isogeometric analysis based on multiplicative Schwarz smoothers,* SIAM Journal on Scientific Computing **41**, 321 (2019).

[84] A. de la Riva, F. Gaspar, and C. Rodrigo, *On the robust solution of an isogeometric discretization of bilaplacian equation by using multigrid methods,* Computers & Mathematics with Applications **80**, 386 (2020).

[85] M. Donatelli, C. Garoni, C. Manni, S. Capizzano, and H. Speleers, *Symbol-based multigrid methods for Galerkin B-spline isogeometric analysis,* SIAM Journal on Numerical Analysis **55**, 31 (2017).

[86] C. Hofer and S. Takacs, *A parallel multigrid solver for multi-patch isogeometric analysis,* in *Advanced Finite Element Methods with Applications: Selected Papers from the 30th Chemnitz Finite Element Symposium 2017*, edited by T. Apel, U. Langer, A. Meyer, and O. Steinbach (Springer International Publishing, Cham, 2019) pp. 205–219.

[87] S. Takacs, *Robust approximation error estimates and multigrid solvers for isogeometric multi-patch discretizations,* Mathematical Models and Methods in Applied Sciences **28**, 1899 (2018).

[88] S. Takacs, *Fast multigrid solvers for conforming and non-conforming multi-patch isogeometric analysis,* Computer Methods in Applied Mechanics and Engineering **371**, 113301 (2020).

[89] J. Sogn and S. Takacs, *Robust multigrid solvers for the biharmonic problem in isogeometric analysis,* Computer Methods in Applied Mechanics and Engineering **77**, 105 (2019).

[90] R. Tielen, M. Möller, and C. Vuik, *A direct projection to low-order level for p-multigrid methods in isogeometric analysis,* in *The Proceedings of the European Numerical Mathematics and Advanced Applications Conference* (2019).

[91] S. Brenner and L. Scott, *The mathematical theory of finite element methods* (Springer, New York, USA, 1994).

[92] W. Briggs, V. E. Henson, and S. McCormick, *A Multigrid Tutorial 2nd edition* (SIAM, Philadelphia, USA, 2000).

[93] R. Sampath and G. Biros, *A parallel geometric multigrid method for finite elements on octree meshes,* SIAM Journal on Scientific Computing **32**, 1361 (2010).

[94] L. Gao and V. Calo, *Fast isogeometric solvers for explicit dynamics,* Computer Methods in Applied Mechanics and Engineering **274**, 19 (2014).

[95] Y. Saad, *ILUT: A dual threshold incomplete LU factorization,* Numerical Linear Algebra with Applications **1**, 387 (1994).

[96] Y. Saad, *Iterative Methods for Sparse Linear Systems* (Society for Industrial and Applied Mathematics, 2003).

[97] N. Collier, L. Dalcin, D. Pardo, and V. Calo, *The costs of continuity: performance of iterative solvers on isogeometric finite elements,* SIAM Journal on Scientific Computing **35**, 767 (2013).

[98] G. Guennebaud *et al.*, *Eigen v3,* http://eigen.tuxfamily.org (2010).

[99] Y. Saad, *Sparskit: a basic tool kit for sparse matrix computations,* (1994).

[100] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid* (Academic Press, London, UK, 2001).

[101] A. Mantzaflaris *et al.*, *G+Smo (Geometry plus Simulation modules) v0.8.1,* http://github.com/gismo (2018).

[102] G. D. Cortes, C. Vuik, and J. Jansen, *On POD-based deflation vectors for DPCG applied to porous media problems,* Journal of Computational and Applied Mathematics **330**, 193 (2018).

[103] C. Gianelli, B. Jüttler, and H. Speleers, *THB-splines: The truncated basis for hierarchical splines,* Computer Aided Geometric Design **29**, 485 (2012).

[104] R. Kraft, *Adaptive and linearly independent multilevel B-splines* (Vanderbilt University Press, Nashville, USA, 1997).

[105] A. Vuong, C. Giannelli, B. Jüttler, and B. Simeon, *A hierarchical approach to adaptive local refinement in isogeometric analysis,* Computer Methods in Applied Mechanics and Engineering **200**, 3554 (2011).

[106] C. Hofreither, B. Jüttler, G. Kiss, and W. Zulehner, *Multigrid methods for isogeometric analysis with THB-splines,* Computer Methods in Applied Mechanics and Engineering **308**, 96 (2016).

[107] C. Bracco, D. Cho, C. Giannelli, and R. Vázquez, *BPX preconditioners for isogeometric analysis using (truncated) hierarchical B-splines,* Computer Methods in Applied Mechanics and Engineering **379** (2021).

[108] C. Hofreither, L. Mitter, and H. Speleers, *Local multigrid solvers for adaptive isogeometric analysis in hierarchical spline spaces,* IMA Journal of Numerical Analysis (2021).

[109] J. M. Tang and Y. Saad, *Domain-decomposition-type methods for computing the diagonal of a matrix inverse,* SIAM Journal on Scientific Computing **33**, 2823 (2011).

[110] I. Nievinkski, M. Souza, P. Goldfeld, D. Augusto, J. Rodrigues, and L. Carvalho, *Parallel implementation of a two-level algebraic ILU(k)-based domain decomposition preconditioner,* TEMA (Sao Carlos) **19**, 59 (2018).

[111] R. A. Nicolaides, *Deflation of conjugate gradients with applications to boundary value problems,* SIAM Journal on Numerical Analysis **24**, 355 (1987).

[112] R. Tielen, M. Möller, D. Göddeke, and C. Vuik, *p-multigrid methods and their comparison to h-multigrid methods within isogeometric analysis,* Computer Methods in Applied Mechanics and Engineering **372** (2020).

[113] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *Parallel time integration with multigrid,* SIAM Journal on Scientific Computing **36**, C635 (2014).

[114] M. Ries, U. Trottenberg, and G. Winter, *A note on MGR methods,* Linear Algebra and its Applications **49**, 1 (1983).

[115] J.-L. Lions, *Résolution d'edp par un schéma en temps pararéel a"parareal"in time discretization of PDE's,* CRASM **332**, 661 (2001).

[116] U. Langer, S. E. Moore, and M. Neumüller, *Space–time isogeometric analysis of parabolic evolution problems,* Computer Methods in Applied Mechanics and Engineering **306**, 342 (2016).

[117] K. Takizawa, T. E. Tezduyar, Y. Otoguro, T. Terahara, T. Kuraishi, and H. Hattori, *Turbocharger flow computations with the space–time isogeometric analysis (ST-IGA),* Computers & Fluids **142**, 15 (2017).

[118] C. Hofer, U. Langer, M. Neumüller, and I. Toulopoulos, *Time-multipatch discontinuous Galerkin space-time isogeometric analysis of parabolic evolution problems,* ETNA - Electronic Transactions on Numerical Analysis **49**, 126 (2018).

[119] C. Hofer, U. Langer, M. Neumüller, and R. Schneckenleitner, *Parallel and robust preconditioning for space-time isogeometric analysis of parabolic evolution problems,* SIAM Journal on Scientific Computing **41**, A1793 (2019).

[120] G. Bal, *On the convergence and the stability of the parareal algorithm to solve partial differential equations,* in *Lecture Notes in Computational Science and Engineering* (Springer-Verlag) pp. 425–432.

[121] M. J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method,* SIAM Journal on Scientific Computing **29**, 556 (2007).

[122] M. J. Gander and E. Hairer, *Nonlinear convergence analysis for the parareal algorithm,* in *Domain Decomposition Methods in Science and Engineering XVII*, edited by U. Langer, M. Discacciati, D. E. Keyes, O. B. Widlund, and W. Zulehner (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008) pp. 45–56.

[123] V. A. Dobrev, T. Kolev, N. A. Petersson, and J. B. Schroder, *Two-level convergence theory for multigrid reduction in time (MGRIT),* SIAM Journal on Scientific Computing **39**, S501 (2017).

[124] B. S. Southworth, *Necessary conditions and tight two-level convergence bounds for parareal and multigrid reduction in time,* SIAM Journal on Matrix Analysis and Applications **40**, 564 (2019).

[125] S. Günther, N. R. Gauger, and J. B. Schroder, *A non-intrusive parallel-in-time approach for simultaneous optimization with unsteady PDEs,* Optimization Methods and Software **34**, 1306 (2018).

[126] S. Günther, N. R. Gauger, and J. B. Schroder, *A non-intrusive parallel-in-time adjoint solver with the XBraid library,* Computing and Visualization in Science **19**, 85 (2018).

[127] M. Lecouvez, R. D. Falgout, C. S. Woodward, and P. Top, *A parallel multigrid reduction in time method for power systems,* in *2016 IEEE Power and Energy Society General Meeting (PESGM)* (2016) pp. 1–5.

[128] J. B. Schroder, R. D. Falgout, C. S. Woodward, P. Top, and M. Lecouvez, *Parallel-in-time solution of power systems with scheduled events,* in *2018 IEEE Power Energy Society General Meeting (PESGM)* (2018) pp. 1–5.

[129] F. Danieli and S. MacLachlan, *Multigrid reduction in time for non-linear hyperbolic equations,* (2021), arXiv:2104.09404 [math.NA] .

[130] Y. Maday and G. Turinici, *The parareal in time iterative solver: a further direction to parallel implementation,* in *Lecture Notes in Computational Science and Engineering* (Springer-Verlag) pp. 441–448.

[131] Y. Maday and G. Turinici, *A parareal in time procedure for the control of partial differential equations,* Comptes Rendus Mathematique **335**, 387 (2002).

[132] G. A. Staff and E. M. Rønquist, *Stability of the parareal algorithm,* in *Lecture Notes in Computational Science and Engineering* (Springer-Verlag) pp. 449–456.

[133] C. Farhat and M. Chandesris, *Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications,* International Journal for Numerical Methods in Engineering **58**, 1397 (2003).

[134] I. Garrido, B. Lee, G. Fladmark, and M. Espedal, *Convergent iterative schemes for time parallelization,* Mathematics of computation **75**, 1403 (2006).

[135] A. de la Riva, C. Rodrigo, and F. J. Gaspar, *A two level method for isogeometric discretizations,* (2020), arXiv:2009.01499 [math.NA] .

[136] R. Tielen, M. Möller, and C. Vuik, *Efficient multigrid based solvers for B-spline MPM,* in *Proceedings of the 2nd International Conference on the MPM for Modelling Soil-Water-Structure Interaction* (2019).

[137] G. C. Diwan and M. S. Mohamed, *Iterative solution of Helmholtz problem with high-order isogeometric analysis and finite element method at mid-range frequencies,* Computer Methods in Applied Mechanics and Engineering **363**, 112855 (2020).

[138] V. Dwarka, R. Tielen, M. Möller, and C. Vuik, *Towards accuracy and scalability: Combining isogeometric analysis with deflation to obtain scalable convergence for the Helmholtz equation,* Computer Methods in Applied Mechanics and Engineering **377**, 113694 (2021).

APPENDIX

APPENDIX A. DIRECT OR INDIRECT PROJECTION

To investigate the effect of a direct projection to $p = 1$, we consider Poisson's equation on the quarter annulus. The number of multigrid cycles needed to achieve convergence with a direct projection and indirect projection has been determined for different values of h and p. Table 1 shows the number of iterations needed to achieve convergence with a direct and indirect projection, respectively. For most configurations, the number of iterations is very similar. Only for higher values of p, the indirect project leads to a diverging method when Gauss-Seidel is applied as a smoother. With a direct projection, all configurations lead to a converging multigrid method. It should be noted that these conclusions also hold for the other considered test problems in Chapter 4.

	$p=2$		$p=3$		$p=4$		$p=5$	
	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	61	3	163	3	473
$h=2^{-9}$	5	32	3	61	3	163	3	452

(a) p-multigrid with a direct projection

$h=2^{-6}$	4	30	3	62	3	–	3	–
$h=2^{-7}$	4	29	3	61	3	–	3	–
$h=2^{-8}$	5	30	3	61	3	–	3	–
$h=2^{-9}$	5	32	3	63	3	–	3	–

(b) p-multigrid with an indirect projection

Table 1: Number of multigrid cycles needed to achieve convergence with p-multigrid for Poisson's equation on the quarter annulus with a direct and an indirect projection.

APPENDIX B. CONSISTENT VS. LUMPED PROJECTION

In Chapter 4, the prolongation and restriction operator to transfer residuals and corrections from level p to 1 and vice versa have been defined. Note that, the mass matrix in Equation (4.8) and (4.10) can be lumped to reduce computational costs. To investigate the effect of lumping the mass matrix within the L_2 projection, the first benchmark is considered.

Table 2 shows the number of multigrid cycles needed to achieve convergence using the lumped or consistent mass matrix in Equation (4.8) and (4.10) for Poisson's equation on the quarter annulus. When ILUT is adopted as a smoother, the number of multigrid cycles needed to reach convergence is identical for all configurations. For Gauss-Seidel, the use of the consistent mass matrix leads to a slightly lower number of iterations. It should be noted that these conclusions also hold for the other considered test problems in Chapter 4. Considering the decrease of computational costs, however, the lumped mass matrix is adopted throughout this dissertation in the prolongation and restriction operator.

	$p=2$		$p=3$		$p=4$		$p=5$	
	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	61	3	163	3	473
$h=2^{-9}$	5	32	3	63	3	163	3	452

(a) Lumped mass matrix $\mathbf{M}_k^L$

$h=2^{-6}$	4	29	3	57	3	171	3	475
$h=2^{-7}$	4	29	3	52	3	174	3	524
$h=2^{-8}$	5	29	3	54	3	165	3	446
$h=2^{-9}$	5	31	3	52	3	164	3	441

(b) Consistent mass matrix $\mathbf{M}_k$

Table 2: Number of multigrid cycles to reach convergence with p-multigrid adopting a lumped or consistent mass matrix in the prolongation and restriction operator for Poisson's equation on the quarter annulus.

APPENDIX C. CPU TIMES p-MULTIGRID

Table 3 shows the CPU timings with p-multigrid as a stand-alone solver for Poisson's equation on the quarter annulus when adopting ILUT and Gauss-Seidel (GS) as a smoother. For each configuration, the assembly, factorization and solver costs are shown separately. Note that, a small difference between the assembly costs can be observed when adopting both smoothers, which are caused by random fluctuations. As Gauss-Seidel does not require any setup costs, only the factorization costs for the ILUT smoother are presented.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	0.65	0.66	1.20	1.19	2.21	2.14	3.63	3.91
$h=2^{-7}$	2.58	2.63	4.64	4.52	8.58	8.53	15.06	14.78
$h=2^{-8}$	10.22	10.77	18.93	19.41	34.77	34.07	57.95	60.81
$h=2^{-9}$	41.52	41.86	74.74	76.70	135.79	131.66	243.64	248.11

(a) Assembly costs in seconds

$h=2^{-6}$	0.10	–	0.27	–	0.55	–	0.93	–
$h=2^{-7}$	0.50	–	1.43	–	2.85	–	4.76	–
$h=2^{-8}$	2.13	–	6.70	–	13.98	–	25.32	–
$h=2^{-9}$	8.66	–	29.78	–	75.89	–	134.57	–

(b) Factorization costs in seconds

$h=2^{-6}$	0.02	0.11	0.02	0.29	0.03	1.03	0.03	3.74
$h=2^{-7}$	0.07	0.35	0.07	1.00	0.09	3.54	0.11	13.24
$h=2^{-8}$	0.32	1.46	0.26	3.82	0.36	13.62	0.49	50.39
$h=2^{-9}$	1.30	6.48	1.07	15.80	1.46	56.15	1.88	195.28

(c) Solver costs in seconds

$h=2^{-6}$	0.77	0.77	1.49	1.48	2.79	3.17	4.59	7.65
$h=2^{-7}$	3.15	2.98	6.14	5.52	11.52	12.07	19.93	28.02
$h=2^{-8}$	12.67	12.23	25.89	23.23	49.11	47.69	75.97	111.20
$h=2^{-9}$	51.48	48.34	105.59	92.50	213.14	187.81	380.09	443.39

(d) Total costs in seconds

Table 3: CPU timings for Poisson's equation on the quarter annulus using p-multigrid.

APPENDIX D. CPU TIMES h-MULTIGRID

Table 4 shows the CPU timings with h-multigrid as a stand-alone solver for Poisson's equation on the quarter annulus when adopting ILUT and Gauss-Seidel (GS) as a smoother. For each configuration, the assembly, factorization and solver costs are shown separately. Note that, a small difference between the assembly costs can be observed when adopting both smoothers, which are caused by random fluctuations. As Gauss-Seidel does not require any setup costs, only the factorization costs for the ILUT smoother are presented.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	GS	ILUT	GS	ILUT	GS	ILUT	GS
$h=2^{-6}$	0.47	0.50	1.11	1.08	2.31	2.16	4.32	4.14
$h=2^{-7}$	2.03	1.80	4.21	4.00	9.35	9.16	15.82	17.28
$h=2^{-8}$	7.68	7.17	17.33	17.64	36.17	33.53	64.25	68.57
$h=2^{-9}$	31.77	30.14	67.10	64.69	143.86	145.38	272.68	265.49

(a) Assembly costs in seconds

$h=2^{-6}$	0.13	–	0.35	–	0.71	–	1.22	–
$h=2^{-7}$	0.70	–	1.84	–	3.85	–	6.12	–
$h=2^{-8}$	2.93	–	8.98	–	18.74	–	32.94	–
$h=2^{-9}$	12.52	–	40.53	–	94.89	–	178.58	–

(b) Setup costs smoother in seconds

$h=2^{-6}$	0.01	0.07	0.02	0.23	0.02	0.98	0.04	4.24
$h=2^{-7}$	0.05	0.18	0.06	0.66	0.09	3.12	0.13	13.26
$h=2^{-8}$	0.21	0.70	0.21	2.49	0.34	10.85	0.49	45.14
$h=2^{-9}$	0.86	3.01	0.87	9.93	1.35	43.12	1.94	168.74

(c) Solver costs in seconds

$h=2^{-6}$	0.61	0.57	1.48	1.31	3.05	3.14	5.59	8.39
$h=2^{-7}$	2.78	1.98	6.11	4.66	13.29	12.28	22.07	30.54
$h=2^{-8}$	10.82	7.87	26.52	20.13	55.25	44.38	97.68	113.71
$h=2^{-9}$	45.15	33.15	108.50	74.62	240.10	188.50	453.20	434.22

(d) Total costs in seconds

Table 4: CPU timings for Poisson's equation on the quarter annulus using h-multigrid with different smoothers.

APPENDIX E. CPU TIMES WITH AN ALTERNATIVE SMOOTHER

Table 5 shows the CPU timings with p-multigrid adopting the smoother from [82] (SCMS) and the ILUT smoother to solve Equation (4.18). For each configuration, the assembly, factorization and solver costs are shown separately. Note that, a small difference between the assembly costs can be observed when adopting both smoothers, which are caused by random fluctuations. The setup times of the ILUT smoother are significantly higher compared to the setup costs of the SCMS smoother. On the other hand, solving the resulting linear system of equations is significantly faster when adopting the ILUT smoother.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS
$h=2^{-6}$	0.42	0.41	0.76	0.81	1.46	1.42	2.70	2.84
$h=2^{-7}$	1.62	1.62	3.04	2.94	6.26	5.91	11.74	11.22
$h=2^{-8}$	6.50	6.65	12.47	12.96	24.19	24.64	46.63	43.60
$h=2^{-9}$	26.91	25.86	47.85	49.10	93.50	94.84	184.05	178.78

(a) Assembly costs in seconds

$h=2^{-6}$	0.06	0.01	0.19	0.01	0.41	0.01	0.78	0.02
$h=2^{-7}$	0.30	0.02	0.96	0.03	2.20	0.04	4.64	0.05
$h=2^{-8}$	1.25	0.09	4.16	0.10	10.64	0.17	22.53	0.20
$h=2^{-9}$	5.26	0.34	17.44	0.37	44.30	0.66	120.64	0.77

(b) Setup costs smoother in seconds

$h=2^{-6}$	0.02	0.19	0.02	0.26	0.03	0.31	0.05	0.44
$h=2^{-7}$	0.06	0.62	0.09	0.77	0.09	1.08	0.16	1.46
$h=2^{-8}$	0.23	2.08	0.26	2.76	0.48	4.03	0.54	5.51
$h=2^{-9}$	0.80	9.05	0.99	11.54	1.82	16.72	2.02	22.31

(c) Solver costs in seconds

$h=2^{-6}$	0.50	0.61	0.97	1.08	1.90	1.74	3.53	3.30
$h=2^{-7}$	1.98	2.26	4.09	3.74	8.55	7.03	16.54	12.73
$h=2^{-8}$	7.98	8.82	16.89	15.82	35.31	28.84	69.70	49.31
$h=2^{-9}$	32.97	35.25	66.28	61.01	139.62	112.22	306.71	201.86

(d) Total costs in seconds

Table 5: CPU times (in seconds) for convergence with p-multigrid to solve Equation (4.18).

APPENDIX F. CPU TIMES WITH AN ALTERNATIVE SMOOTHER

Table 6 shows the CPU timings with h-multigrid adopting the smoother from [82] (SCMS) and the ILUT smoother to solve Equation (4.18). For each configuration, the assembly, factorization and solver costs are shown separately. Note that, a small difference between the assembly costs can be observed when adopting both smoothers, which are caused by random fluctuations. The setup times of the ILUT smoother are significantly higher compared to the setup costs of the SCMS smoother. On the other hand, solving the resulting linear system of equations is significantly faster when adopting the ILUT smoother.

	$p=2$		$p=3$		$p=4$		$p=5$	
	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS	ILUT	SCMS
$h=2^{-6}$	0.30	0.30	0.73	0.71	1.60	1.62	3.24	3.19
$h=2^{-7}$	1.13	1.19	2.90	2.86	6.42	6.47	12.78	11.73
$h=2^{-8}$	4.56	4.88	11.63	11.59	26.06	25.86	48.84	49.67
$h=2^{-9}$	19.08	19.44	46.63	46.19	95.92	104.10	202.52	202.64

(a) Assembly costs in seconds

$h=2^{-6}$	0.08	0.01	0.25	0.01	0.63	0.02	1.02	0.02
$h=2^{-7}$	0.39	0.02	1.25	0.03	2.80	0.06	5.79	0.08
$h=2^{-8}$	1.66	0.09	5.71	0.12	14.23	0.22	27.66	0.27
$h=2^{-9}$	7.27	0.33	24.74	0.43	59.12	0.88	156.73	1.04

(b) Setup costs smoother in seconds

$h=2^{-6}$	0.03	0.07	0.06	0.11	0.09	0.16	0.14	0.25
$h=2^{-7}$	0.08	0.22	0.11	0.34	0.23	0.58	0.37	0.80
$h=2^{-8}$	0.25	0.96	0.34	1.45	0.91	2.22	1.11	3.03
$h=2^{-9}$	0.88	4.34	1.17	5.72	3.19	9.11	3.85	12.13

(c) Solver costs in seconds

$h=2^{-6}$	0.41	0.38	1.04	0.83	2.32	1.80	4.40	3.46
$h=2^{-7}$	1.60	1.43	4.26	3.23	9.45	7.11	18.94	12.61
$h=2^{-8}$	6.47	5.93	17.68	13.16	41.20	28.30	77.61	52.97
$h=2^{-9}$	27.23	24.11	72.54	52.34	158.23	114.09	363.10	215.81

(d) Total costs in seconds

Table 6: CPU times (in seconds) for convergence with h-multigrid to solve Equation (4.18).

ACKNOWLEDGEMENTS

This thesis could not have been written without the help of many around me. First of all, I would like to address my gratitude to my promotor and supervisor **Matthias Möller**. The choice of doing my master work under your supervision turned out to be the start of a very pleasant collaboration we had during the past five years. Over the years, I learned a lot from you with respect to scientific writing, programming and numerical analysis in general for which I am very thankful. There are many characteristics one can look for in the ideal supervisor and, for me, you had so many of these. Apart from working together, I really enjoyed the many conferences we visited together, for example those in San Diego, Austin and Munich.

I would also like to thank my promotor **Kees Vuik** for his guidance during the PhD project. Your experience and ability to ask the right questions has helped me a lot to finish my PhD successfully. Furthermore, despite your busy schedule, you always remained approachable for me and read every page of text I wrote during the entire PhD. I am very aware of the luxury position we as PhD students have with such a promotor and head of the department. Finally, you created an atmosphere which is pleasant to work in, supporting team building activities as the Krylov Tigers and SIAM Student Chapter.

Thank you, my committee members, for reading this thesis and providing me with useful comments and suggestions. Special thanks to **Dominik Göddeke**, who made my stay in Stuttgart possible, which helped me to develop myself not only as a researcher, but as a person as well. The personal growth during those six months in Germany formed a strong foundation and led to a kick-start of my PhD project in 2017. Furthermore, I want to thank **Kees Oosterlee** for the fruitful discussions we had on multigrid methods at the start of my PhD and the pointers he provided to interesting multigrid literature.

I had the pleasure to be surrounded with people that made life in the office pleasant. Thank you, my academic brother and neighbor **Jochen**, whom I could never convince to learn a proper programming language. Even though I was often puzzled after talking with you about research, it was nice to share experiences and ideas. The workshop we visited together in Stanford, including cycling the Golden Gate bridge in San Francisco, is definitely one of the highlights of my PhD. **Marieke**, for the many chats we had on more philosophical and less mathematical stuff. I still try to follow your advice and evaluate the success of my PhD on the amount of happiness it creates instead of the number of papers published. **Vandana**, although you support the wrong football team, it was a pleasure to share the office with you. In particular, I am very proud how we worked together on a paper about solvers for the Helmholtz equation which was published in no time! Also thanks to **Mousa**, who was with me in the SIAM Student Chapter board in 2018 and helped with organizing the workshop on 'Multigrid and Multilevel methods'. Finally, thank you **Prajakta**, who tried to be earlier than me in the office every day over and over again, but hardly succeeded. I have great respect for the project you started during the Covid-19 pandemic ('Masks for India') to help people in India.

It was a luxury that there was a second office where I felt at home and became a 20% office mate. I played nice games of badminton with you, **Alice**, and had so often fries at the snackbar afterwards! Furthermore, your own cooking skills can not be underestimated, both in quantity as quality. Thank you for bringing me to China, it was a trip I will never forget! Furthermore, I played football twice a week with **Mo**, who never stopped running during the game (in contrast to me). It was a pleasure to have the fondue nights at your place with the others of the 'Coolest office'. Being part of the lunchgroup with you, **Xiao Shan**, was a pleasure every week again. Many thanks to **Merel** as well, who did so much as president of the SIAM SC in 2018 and accompanied me to China as well. In particular, I want to thank **Gaby** for all the nice moments we shared during the final years of your PhD. It was a great pleasure and honour for me to be your paranimf! Although being part of another office, I wanna mention **Anne** here as well, as being my occasional biking partner who taught me how to play 'klaverjassen', which I mostly enjoy now (although I still prefer 'rikken').

Most of the days I was wandering around lunch time on the second and third floor to gather everyone for lunch. Thanks **Kristof, Dennis, Giordano, Jiao, Thomas, Alexander, Merel** and **Deepesh** for joining me from time to time! Also thanks to **Martin, Fred, Neil** and **Peter** for the chats at the coffee machine and the cake corner.

Special thanks to **Hugo** for the many coffee breaks we had to talk about Isogeometric Analysis and politics, of which I enjoyed the latter subject the most. Furthermore, I am happy that you took over my educational tasks with respect to Multi-Media Math Education (MUMIE). In that respect, I wanna thank **Petrus Tan** and **Sabine Greiser** from Integral Learning for all their help during the last years to keep the online education with MUMIE up and running for all students.

The BaNaNa talks that I organized would not have been there without the founding father **Manuel**. It was a pleasure to take over organizing these talks and I hope many will follow in the future. I started the PhD Forum together with **Amey, Andrea, Bart, Gerrit** and **Tom**. It was a pleasure to organize this non-standard seminar specifically designed for PhD students! Also thanks to **Deborah** for arranging all the small but important things during my PhD and **Kees Lemmens** for the technical support provided during my PhD.

During the second year of my PhD, I had the opportunity to help supervising a master student, **Pascal**, whose work even led to a journal publication. You did a very good job! In the last year of my PhD, my second master student, started his project on block ILUT smoothers for Isogeometric Analysis. Keep up the good work **Mark** and I am looking forward to reading your thesis! Furthermore, thank you, **Lisa**, for working successfully together on MPM for quite some years and, of course, for supervising me when I was still a master student.

Playing football on Monday for the Krylov Tigers was a highlight every week over and over again. Thanks, **Baljaa, Nirmal, Dave, Keshav, Shuaiqiang** and many more for forming this fantastic team! Then, there was the Tuesday football group, including dinner and Champions League afterwards. I want to thank the core members, **Reinaldo, Michele, Swej, Mogre** and **Wissam** for staying afterwards for drinks and analyzing the game over and over again. Finally, there was the Delft Summer Football group, for which I played many matches during the last year of my PhD. Thanks **Robbert-Jan, Ibrahim** and **Chris-**

tian for co-organizing those games! Special thanks to my football and tennis partner **Slawek**, who also prepared so many nice dinners together with **Dominika**. Furthermore, I have to thank him for upgrading my home office with all the necessary equipment to write this dissertation.
When starting my Bachelors in Delft, I met my friends **Sander, Pieter** and **Hugo**. Even after finishing our masters, we still kept in touch, having dinner, go cycling or watching the movies together. People often say that COVID-19 brings people further apart, but for us it was actually the opposite. The weekly boardgames during the lockdown have been a pleasure, thank you guys!
Finally, I would like to thank my parents, who always supported me to perform the best I could. Furthermore, they made sure, together with my brothers, that I had a stable and loving foundation to fall back on if needed. I believe that you can influence many aspects in your life, but in which family you grow up is a matter of pure luck. Thank you so much that I can call myself so lucky!

Curriculum Vitæ

Roel Petrus Wilhelmus Maria TIELEN

24-10-1992 Born in Roosendaal, The Netherlands.

Education

2004–2010 Secondary School
Norbertuscollege, Roosendaal

2010–2013 Bachelor in Applied Mathematics
Delft University of Technology

2013–2016 Master in Applied Mathematics
Delft University of Technology

2013–2016 Master in Science Education and Communication
Delft University of Technology

Career history

2016–2017 Research assistant
University of Stuttgart

2017–2021 PhD researcher
Delft University of Technology

Awards

2017 Poster award Woudschoten conference

2018 Poster award IgA 2018

2019 SIAM Award of Recognition

Roel Tielen was born on October 24 in the year 1992 in Roosendaal (the Netherlands). His academic career started in 2010 at Delft University of Technology (TU Delft), where he started his Bachelor's in Applied Mathematics. In 2016, he graduated from his masters in both Applied Mathematics and Science Education and Communication (SEC).
After finishing his masters, he worked as a research assistant at the University of Stuttgart under auspices of Prof. Dominik Göddeke on multigrid methods for Isogeometric Analysis. In 2017, he returned to TU Delft to start his PhD at the Delft Institute of Applied Mathematics (DIAM) under the supervision of Prof. Cornelis Vuik and Dr. Matthias Möller.
The initial goal of the PhD project was to adopt concepts from Isogeometric Analysis and high-order time integration techniques to improve the Material Point Method, which was already part of his masters. However, during the PhD the focus of the research remained on multigrid methods (in particular p-multigrid methods) for Isogeometric Analysis.
Most of his work has been performed in the open-source C++ library G+Smo, where he contributed as a developer. Furthermore, he has been responsible for Multi-Media Math Education (MUMIE), an online learning tool used in the courses on Numerical Methods for Differential Equations at TU Delft. During his PhD project, he supervised, together with Matthias Möller, his first master student, Pascal de Koster, on the use of Powell-Sabin splines within MPM.
Over the years, he visited a variety of conferences to present his work, including the international conferences on Isogeometric Analysis (in 2016, 2018 and 2019), Isogeometric Analysis and Applications (in 2018 and 2020) and the Material Point Method (in 2017 and 2019). Furthermore, he published in (peer-reviewed) proceedings and international journals, both as a first and second author on p-multigrid methods and MPM. During two conferences, Woudschoten 2017 and IgA 2018, his work was awarded with the third place in the poster competition.

List of Publications

Under Review

1. **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 [math.NA]

Scientific Articles in Journals

4. V. Dwarka, **R. Tielen**, M. Möller and C. Vuik, *Towards accuracy and scalability: Combining Isogeometric Analysis with deflation to obtain scalable convergence for the Helmholtz equation*, Computer Methods in Applied Mechanics and Engineering **377** 113694 (2021)

3. P. de Koster, **R. Tielen**, E.D. Wobbes and M. Möller, *Extension of B-spline Material Point Method for unstructured triangular grids using Powell–Sabin splines*, Journal of Computational Particle Mechanics **8** (2021)

2. E.D. Wobbes, **R. Tielen**, M. Möller and C. Vuik, *Comparison and unification of material-point and optimal transportation meshfree methods*, Journal of Computational Particle Mechanics **8** (2021)

1. **R. Tielen**, M. Möller, D. Göddeke and C. Vuik, *p-multigrid methods and their comparison to h-multigrid methods within Isogeometric Analysis*, Computer Methods in Applied Mechanics and Engineering **372** (2020)

Scientific Articles in Proceedings (peer-reviewed)

7. **R. Tielen**, M. Möller and C. Vuik, *Multigrid Reduced in Time for Isogeometric Analysis*, in The Proceedings of the Young Investigators Conference (2021)

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

5. **R. Tielen**, M. Möller and C. Vuik, *Efficient p-Multigrid based solvers for multipatch geometries in Isogeometric Analysis*, Lecture Notes in Computational Science and Engineering, Springer, **133** (2020)

4. **R. Tielen**, M. Möller and C. Vuik, *A direct projection to low-order level for p-multigrid methods in Isogeometric Analysis*, in The Proceedings of the European Numerical Mathematics and Advanced Applications Conference (2019).

3. **R. Tielen**, M. Möller and C. Vuik, *Efficient multigrid based solvers for B-spline MPM*, in The Proceedings of the 2nd International Conference on the MPM for Modelling Soil-Water-Structure Interaction (2019)

2. E.D. Wobbes, **R. Tielen**, M. Möller, C. Vuik and V. Galavi, *Comparison between Material Point Method and meshfree schemes derived from optimal transportation theory*, in The Proceedings of the 2nd International Conference on the MPM for Modelling Soil-Water-Structure Interaction (2019)

1. **R. Tielen**, E.D. Wobbes, M. Möller and L. Beuth, *A high order material point method*, in The Proceedings of the 1st International Conference on the MPM for Modelling Soil-Water-Structure Interaction (2017)

SCIENTIFIC ARTICLES IN PROCEEDINGS

1. **R. Tielen**, M. Möller and C. Vuik, *Efficient multigrid based solvers for isogeometric analysis*, in The Proceedings of the 6th European Conference on Computational Mechanics and 7th European Conference on Computational Fluid Dynamics (2018)

BOOK CHAPTERS

2. W.T. Solowski, M. Berzins, W.M. Coombs, J.E. Guilkey, M. Möller, Q.A. Tran, T. Adibaskoro, S. Seyedan, **R. Tielen**, and K. Soga, *Chapter 3 in Advances in Applied Mechanics: Material Point Method: overview and challenges ahead*, Academic Press, US

1. E.D. Wobbes, **R. Tielen**, M. Möller, C. Vuik and V. Galavi, *Chapter 4 in The Material Point Method for Geotechnical Engineering: A Practical Guide*, CRP Press, US

List of Presentations

Talks at conferences

An overview of all conferences visited can be found below. At all conferences, a talk has been given, apart from the Spring Meetings.

2021

- 20th Copper Mountain Conference On Multigrid Methods 2021, March 29- April 2, 2019 (virtual)
- Spring Meeting, Delft, the Netherlands, May 28, 2021 (virtual)
- VI Eccomas Young Investigators Conference 2021, July 7-9, 2021 (virtual)
- 16th US National Congress on Computational Mechanics, July 25-29, 2021 (virtual)

2020

- International Conference on Isogeometric Analysis (VIgA) , August 11-12, 2020 (virtual)
- Sparse Days , November 23-24, 2020 (virtual)
- G+Smo developer days, December 9-11, 2020 (virtual)

2019

- 2nd International Conference on the Material Point Method for Modelling Large Deformation and Soil-Water-Structure Interaction (MPM 2019), organized by Aruna 3D MPM Research Community, Cambridge, UK, January 8-10, 2019
- 19th Copper Mountain Conference On Multigrid Methods 2019, Copper Mountain, Colorado, March 24-28, 2019
- Spring Meeting, Antwerp, Belgium, May 17, 2019
- The Mathematics of Finite Elements and Applications (MAFELAP) 2019, London, UK, June 18-21, 2019
- International Conference on Isogeometric Analysis (IgA) 2019, Munich, Germany, September 18-20, 2019

- European Numerical Mathematics and Advanced Applications Conference (ENUMATH) 2019, Egmond aan Zee, The Netherlands, September 30- October 4, 2019

2018

- 3rd Conference on Isogeometric Analysis and Applications (IGAA 2018), Delft, The Netherlands, April 23-27, 2018
- G+Smo developer days 2018, Öckerö, Sweden, May 7-9, 2018
- SIAM Student Chapter workshop on Multigrid and Multilevel Methods, Delft, The Netherlands, May 30, 2018
- Spring Meeting, Eindhoven, The Netherlands, June 1, 2018.
- 6th European Conference on Computational Mechanics (ECCM VI), 7th European Conference on Computational Fluid Dynamics (ECFD VII), Glasgow, UK, June 11-15, 2018.
- USACM Conference on Isogeometric Analysis: Integrating Design and Analysis , Austin, Texas, October 10-12, 2018.

2017

- 1st International Conference on the Material Point Method for Modelling Large Deformation and Soil-Water-Structure Interaction (MPM 2017), organized by Aruna 3D MPM Research Community, Delft, January 10-13, 2017

2016

- USACM Conference on Isogeometric Analysis and Meshfree Methods, La Jolla, California, October 10-12, 2016

POSTER PRESENTATIONS

- 42th Woudschoten Conference, Zeist, The Netherlands, October 4-6, 2017.
- Advanced School on Immersed Methods, Eindhoven, The Netherlands, November 6-9, 2017
- 43th Woudschoten Conference, Zeist, The Netherlands, October 3-5, 2018
- USACM Conference on Isogeometric Analysis: Integrating Design and Analysis, Austin, Texas, October 10-12, 2018.
- 44th Woudschoten Conference, Zeist, The Netherlands, October 9-11, 2019