Efficient p-multigrid solvers for Isogeometric Analysis

Mark Looije
Supervised by Roel Tielen and Kees Vuik

April 22, 2021

Contents

1 Introduction

Isogeometric Analysis (IgA) is easiest explained and motivated as an extension to the Finite Element Method (FEM). While FEM is currently most prevalent for solving problems over irregular domains, this method does have its inefficiencies. The main one being the translation between the geometry created by CAD (Computer Aided Design) and an analysis-suitable geometry. This process is computationally expensive; it is estimated that for complex engineering designs this translation is responsible for 80% of overall analysis time. All while only being an approximation to the 'exact' geometry.

This is the original motivation for IgA; the use of B-Spline basis functions allows for a highly accurate representation of complex geometries as well as establishing a link between the design and the engineering tools. Another positive for IgA is the higher continuity of basis functions, which is a nice property that is more and more appreciated.

Challenges in IgA include assembly of the system matrix and right hand side vector, which isn't as straightforward as it may be in for example FEM. The main challenge however, lies in the fact that many common solvers for linear systems don't perform well for our IgA discretization.
The proposed method for solving the linear system is a p-multigrid method. In most multigrid methods the coarsening is in the mesh width h, here coarsening is applied in the spline degree p. This comes with its own intergrid operators and challenges.
Traditional smoothers such as Gauss Seidel do not perform for higher spline degree p. Hence we need to consider alternative smoothers. For this smoothers are suggested based on incomplete LU factorizations.

The core of this literature review will start with a description of IgA and how to go from a boundary value problem to a linear system of equations (chapter 2). After this strategies are proposed on how to efficiently solve these systems (chapter 3). Finally in chapter 4 there are the results of several numerical experiments, that match the behaviour one would expect from the literature.

Sources: [1], [2].

2 Isogeometric Analysis

2.1 Variational Formulation and Geometry Function

As mentioned before, IgA is used to find an approximation to boundary value problems. For instance, the homogeneous Poisson equation could be considered.

$$\begin{cases} -\Delta u &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega. \end{cases}$$

Here Ω is an open domain in $\mathbb{R}^d$, $\partial\Omega$ the boundary of Ω, $f \in L^2(\Omega)$.

2.1.1 Variational Form

As is the case with FEM, the solution stategy is based on a textbfvariational formulation (also known as weak formulation). To find the variational formulation to this boundary value problem, we consider the function space V. Here V is defined as

$$V := \{v \in H^1(\Omega),\ v = 0 \text{ on } \partial\Omega\},$$

where $H^1(\Omega)$ the first order Sobolev space. This space consists of all functions $v \in L_2(\Omega)$ that have weak and square-integrable first derivatives as well as being 0 on the boundary. Please note that the solution u to the boundary value problem must lie in this set V.

The variational form is obtained by multiplying the PDE with a test function $v \in V$ and integrating over Ω. Then integration by parts and Gauss' divergence theorem are applied. The boundary integral vanishes as v is zero on the boundary.

$$\begin{aligned} -\int_\Omega \Delta u v \ \mathrm{d}\Omega &= \int_\Omega f v \ \mathrm{d}\Omega, \\ -\int_\Omega \operatorname{div}(\nabla u v) \ \mathrm{d}\Omega + \int_\Omega \nabla u \nabla v \ \mathrm{d}\Omega &= \int_\Omega f v \ \mathrm{d}\Omega, \\ -\int_{\partial\Omega} \nabla u v \cdot n \ \mathrm{d}\Gamma + \int_\Omega \nabla u \nabla v \ \mathrm{d}\Omega &= \int_\Omega f v \ \mathrm{d}\Omega, \\ \int_\Omega \nabla u \nabla v \ \mathrm{d}\Omega &= \int_\Omega f v \ \mathrm{d}\Omega, \end{aligned}$$

which will be abbreviated as

$$a(u, v) = \langle f, v \rangle .$$

For the boundary value problem to hold, $a(u, v) = \langle f, v \rangle$ must hold for every $v \in V$. Hence solving the variational form for every such v will give the desired solution to the boundary value problem.

To discretize the variational form, V is replaced by a finite dimensional subspace $V_h \subseteq V$. Let $\phi_1, ..., \phi_n$ be a basis of V_h, then the numerical approximation $u_h \in V_h$ is constructed as the linear combination of these basis functions:

$$u_h = \sum_{i=1}^{n} u_i \phi_i.$$

Inserting u_h into the variational form and testing with $v = \phi_i$ for $i = 1, ..., n$ we find

$$a(u_h, \phi_i) = \sum_{i=1}^{n} u_i \ a(\phi_i, \phi_j) \ \text{ for } j = 1, ..., n$$

And therefore we obtain the linear system

$A\mathbf{u} = \mathbf{f}$, where

$A_{i,j} = a(\phi_i, \phi_j),\ f_i = \langle f, \phi_i \rangle \quad i, j = 1, ..., n.$

Determining the coefficients u_i through this linear system will give the approximation to the variational form and therefore the boundary value problem.

2.1.2 Geometry function

Now, suppose the physical domain is parametrized. Suppose that there is some **geometry function** $\mathbf{F}$ that is an invertible mapping from parameter domain Ω_0 to physical domain Ω (see figure 1).

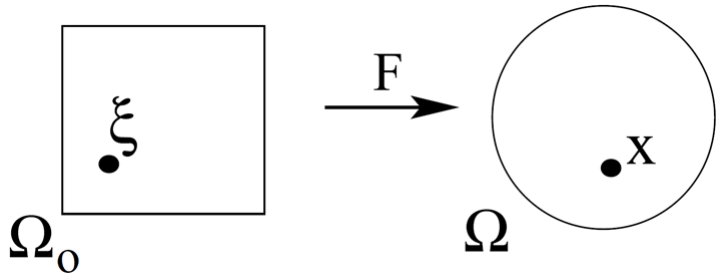


Figure 1: Parametrization of Ω. The parametric domain is denoted by Ω_0. From: [3]

The following is a well known integration rule

$$\int_\Omega w(\boldsymbol{x})d\boldsymbol{x} = \int_{\Omega_0} w(\boldsymbol{F}(\boldsymbol{\xi}))|\det \boldsymbol{DF}(\boldsymbol{\xi})|d\boldsymbol{\xi},$$

with $\boldsymbol{DF}(\boldsymbol{\xi}) = (\frac{\partial F_i}{\partial \xi_j})_{i,j=1,...,d}$ the Jacobian matrix.

Applying this to the integrals in the variational form we find

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v d\boldsymbol{x} = \int_{\Omega_0} (\nabla u \boldsymbol{DF}(\boldsymbol{\xi})^{-1}) \cdot (\nabla v \boldsymbol{DF}(\boldsymbol{\xi})^{-1})|\det \boldsymbol{DF}(\boldsymbol{\xi})|d\boldsymbol{\xi},$$

$$\langle f, v \rangle = \int_\Omega f v d\boldsymbol{x} = \int_{\Omega_0} (fv)(\boldsymbol{F}(\boldsymbol{\xi}))|\det \boldsymbol{DF}(\boldsymbol{\xi})|d\boldsymbol{\xi}.$$

This is slightly different to what is done in FEM. There the geometry transformation is applied later, on integrals over elements, not the entire domain. As the grid in FEM is (often) build from triangles, this transformation is a linear transformation. The basis functions are often tent-functions, but also other choices for basis functions can be made.

In IgA B-splines (or more generally NURBS) are used to define $\mathbf{F}$. For the basis functions we will use B-spline basis functions, which will be defined in section 2.2.

Sources: [4],[3].

2.2 B-splines

An example of a B-spline curve can be seen in figure 2. B-spline curves, surfaces and solids are excellent for describing complicated shapes. Also, the B-spline basis functions are used as the basis functions in our IgA discretization.

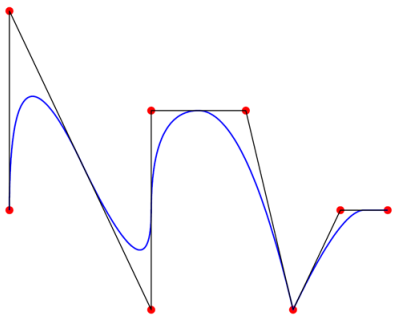


Figure 2: An example of a B-spline, in this case a piecewise quadratic curve in $\mathbb{R}^2$. From: [5]

2.2.1 B-spline basis functions

B-spline basis functions are piecewise polynomials and defined using a **knot vector**. A knot vector in one dimension is a set of coordinates in the parametric space, written $\Xi = \{\xi_1, \xi_2, ..., \xi_{n+p+1}\}$. Here $\xi_i \in \mathbb{R}$ is the i-th knot, i the knot index, p the polynomial order, and n the number of basis functions which compromise the B-spline.
For instance, a typical knot vector for piecewise quadratic ($p = 2$) polynomials would be $\{0, 0, 0, \frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8}, 1, 1, 1\}$. This vector is open (as it repeats the endpoints p times) and uniform (the distances between the knots is uniform).

B-spline basis functions are defined recursively

$$\phi_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1}, \\ 0 & \text{else,} \end{cases}$$

$$\phi_{i,p}(\xi) = \begin{cases} \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i}\phi_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}}\phi_{i+1,p-1}(\xi) & \text{if well defined,} \\ 0 & \text{else.} \end{cases}$$

This recursion is known as the Cox-de Boor formula. Due to this recursive definition the derivatives of B-spline basis functions are efficiently represented in terms of the lower order bases

$$\frac{d}{d\xi}\phi_{i,p}(\xi) = \begin{cases} \frac{p}{\xi_{i+p} - \xi_i}\phi_{i,p-1}(\xi) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}}\phi_{i+1,p-1}(\xi) & \text{if well defined,} \\ 0 & \text{else.} \end{cases}$$

Similar expressions exist for the higher order derivatives, but we don't deem it necessary discussing them here.

Examples of some lower order basis functions can be seen in figure 3. The knot vector used is $\{0, \frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8}, 1\}$ for $p = 0$ and expended with additional zeros and ones for increasing p.

Some important properties of the B-spline basis functions include:

- They add to one, that is, $\forall\xi \ \sum_{i=1}^{n} \phi_{i,p}(\xi) = 1$. This is known as the partition of unity property.
- They are non-negative, that is $\phi_{i,p}(\xi) \geq 0 \ \forall\xi$.
- The support of each $\phi_{i,p}$ is compact and contained in the interval $[\xi_i, \xi_{i+p+1}]$.

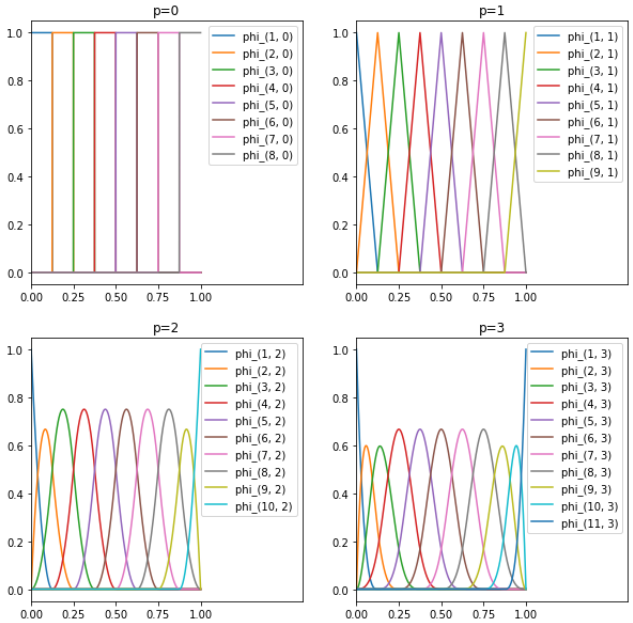


Figure 3: Basis functions of order 0,1,2,3 for an open uniform knot vector

- They are C^{p-1}(that is, they have continuous derivatives up to order $p-1$), if there are no repeated knots. They are C^{p-k-1} if a knot is repeated k times.

These properties immediatly give for some usefull consequences. For example, the support structure gives for sparse matrices. It can also be seen that the amount of non-zero entries increases for higher polynomial order p. The mass matrix M (more on this in later chapters) will be easy to lump due to the partition of unity property, etc.

2.2.2 B-spline curves, surfaces and solids

B-spline curves in $\mathbb{R}^d$ are constructed by taking linear combinations of B-spline basis functions. The coefficients are the control points. Given n basis functions, $\phi_{i,p}$, $i = 1, 2, ..., n$ and corresponding control points $B_i \in \mathbb{R}^d$, $i = 1, 2, ..., n$ a piecewise-polynomial B-spline curve is given by

$$C(\xi) = \sum_{i=1}^{n} \phi_{i,p}(\xi) B_i$$

In general the control points are not interpolated by B-spline curves. Their continuity is inherited by the continuity of the B-spline basis functions, though now the continuity is also decreased for repeated control points. Both this interpolation of control points (marked in red) and the decrease in continuity can be seen in figure 2.

B-spline surfaces and B-spline solids are defined in a similar manner. For the **B-spline surfaces** consider a control net $B_{i,j}$, $i = 1, ..., n$, $j = 1, ..., m$ and knot vectors $\Xi = \{\xi_1, ..., \xi_{n+p+1}\}$ and $H = \{\eta_1, ..., \eta_{m+q+1}\}$. Then the tensor product B-spline surface is defined by

$$S(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \phi_{i,p}(\xi) \psi_{j,q}(\eta) B_{i,j}$$

The formulation for a B-spline solid contains a control net $B_{i,j,k}$ with 3 indices, 3 knot vectors and 3 sets of basis functions.

2.2.3 Refinement strategies

In the multigrid solver it will be necessary to describe B-splines using different sets of basis functions and control points. Several strategies are known for refining the B-spline basis through expanding the knot vector Ξ and adjusting the control points. These strategies are called h-refinement, p-refinement and k-refinement.
h- and p-refinement also exist in FEM and are known as knot insertion and order elevation respectively. k-refinement is exclusive to IgA. They are best explained through a simple example:

h-refinement: $\{0, 0, \frac{1}{2}, 1, 1\}$ to $\{0, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 1\}$; additional distinct knots are inserted.
p-refinement: $\{0, 0, \frac{1}{2}, 1, 1\}$ to $\{0, 0, 0, \frac{1}{2}, \frac{1}{2}, 1, 1, 1\}$; the multiplicity of every knot is increased.
k-refinement: $\{0, 0, \frac{1}{2}, 1, 1\}$ to $\{0, 0, 0, \frac{1}{2}, 1, 1, 1\}$; the multiplicity of the endpoints is increased.
The impact of these refinements on the basis functions can be seen in figure 4.

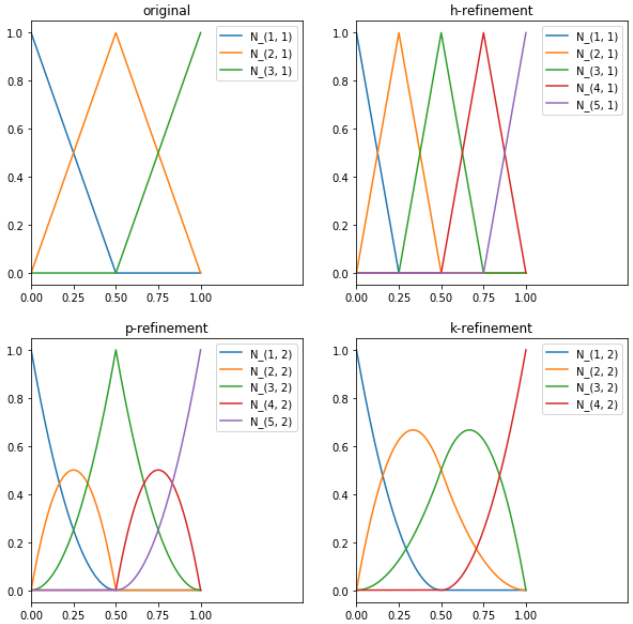


Figure 4: Different refinement techniques

Important observations to make: The number of basis functions almost doubles for h- and p-refinement (though this is better seen in bigger examples), while for k-refinement the number of basis functions only increases by one. Both p- and k-refinement allow for order elevation of the basis functions, though another major benefit of k-refinement opposed to p-refinement being a better continuity for the internal knots.
Throughout this literature study, we adopt k-refinement unless stated otherwise. In chapter 3 it is further elaborated how changing the knot vector and basis functions impacts the coefficients.

Sources: [1], [5].

2.3 Matrix Assembly

For the matrix assembly a method is shown that is very similar to the method used in FEM. The idea is to decompose the integral over the entire parameter domain Ω_0 into rectangular or cuboid atomic contributions and apply standard cubature rules such as Gauss formulae on each rectangle resp. cuboid afterwards.

Using system matrix A and right hand side vector $\mathbf{f}$ from section 2.1, assuming $\mathbf{F} = \mathbf{id}$ so that the expressions remain readable, we receive

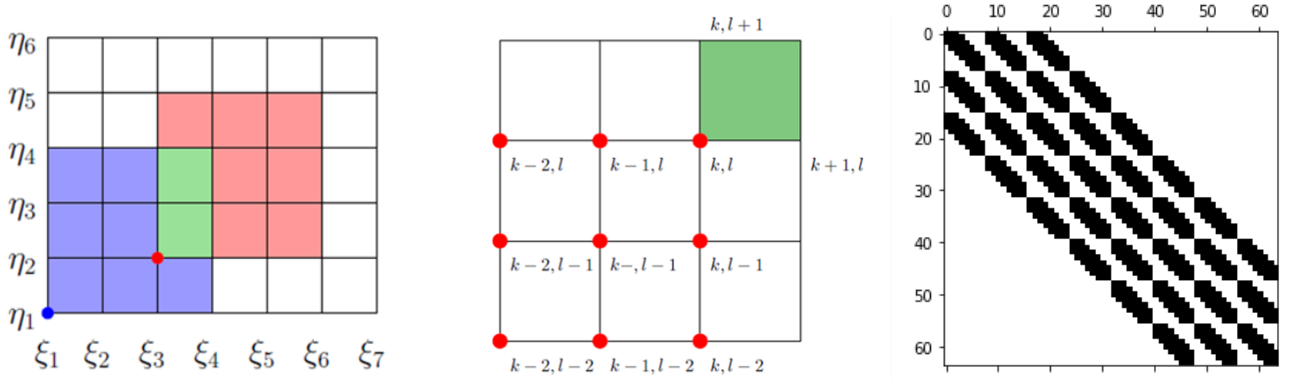
$$A_{i,j} = \int_\Omega \nabla\phi_i \nabla\phi_j \, \mathrm{d}\Omega = \sum_{k=1}^{N_{\mathrm{el}}} \int_{\Omega_k} \nabla\phi_i \nabla\phi_j \, \mathrm{d}\Omega,$$

$$f_i = \int_\Omega f\phi_i \, \mathrm{d}\Omega = \sum_{k=1}^{N_{\mathrm{el}}} \int_{\Omega_k} f\phi_i \, \mathrm{d}\Omega.$$

2.3.1 Support of basis functions

In section 2.2.1 the support structure of the basis functions was already mentioned. They are zero over most elements, making the integrals over those elements zero by default, thus reducing the amount of integrals that have to be calculated.

The support of these basis functions can be visualized in several ways. The first being: given a basis function, in what elements is this basis function unequal to zero (see figure 5a). The blue area is the support of basis function $\phi_{1,1}$, the red area the support of $\phi_{3,2}$, the green area is where both basis functions are supported. $p = 2$ in this image.
Another way to look at it: given an element, what basis functions are supported on this element (see figure 5b). Again for $p = 2$, the red dots show what basis functions are relevant for integrating over the green area.
Finally, in figure 5c the nonzero pattern of the matrix A is shown for $p = 2$ and $N_{\mathrm{el}} = 8$ in both directions. As $p = 2$ we see two non-zero entries to the left and two non-zero entries to the right of the diagonal. As well as two blocks to the left and on the right respectively.



(a) bsplinesupport1 (b) bsplinesupport2 (c) sparsitypattern

Figure 5: bsplinesupport. (a) and (b) from: [1]

2.3.2 Integral approximation

The selected approximation method is Gauss quadrature.

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

An n-point Gauss quadrature rule is exact for polynomials of degree $2n-1$, by suitable choice of quadrature points x_i and weights w_i. These points and weights for the standard interval $[-1,1]$ can be found in figure 6.

Number of points, n	Points, x_i		Weights, w_i	
1	0		2	
2	$\pm\frac{1}{\sqrt{3}}$	±0.57735...	1	
3	0		$\frac{8}{9}$	0.888889...
	$\pm\sqrt{\frac{3}{5}}$	±0.774597...	$\frac{5}{9}$	0.555556...
4	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$	±0.339981...	$\frac{18+\sqrt{30}}{36}$	0.652145...
	$\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	±0.861136...	$\frac{18-\sqrt{30}}{36}$	0.347855...
5	0		$\frac{128}{225}$	0.568889...
	$\pm\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}$	±0.538469...	$\frac{322+13\sqrt{70}}{900}$	0.478629...
	$\pm\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$	±0.90618...	$\frac{322-13\sqrt{70}}{900}$	0.236927...

Figure 6: Gauss quadrature points and weights. From: [6]

Considering for example the expressions for $A_{i,j}$ earlier this section, degree of basis functions $p=2$. Then both $\nabla\phi_i$ and $\nabla\phi_j$ would have have degree 1, hence the integrand having polynomial order 2. 2 quadrature points would be sufficient to exactly calculate the integral.

Generally, the geometry function $\mathbf{F}$ isn't simply the identity, the integrand isn't a polynomial. Still for these functions, Gauss quadrature works quite well.

Change of interval

Figure 6 gives the points and weights for an integral over $[-1,1]$, however this isn't the domain we are interested in. This is easily accounted for using a simple transformation. An integral over $[\xi_i,\xi_{i+1}]$ is approximated

$$\int_{\xi_i}^{\xi_{i+1}} f(\xi)d\xi = \frac{\xi_{i+1}-\xi_i}{2}\int_{-1}^{1} f\left(\frac{\xi_{i+1}-\xi_i}{2}x+\frac{\xi_i+\xi_{i+1}}{2}\right)dx$$

$$\approx \frac{\xi_{i+1}-\xi_i}{2}\sum_{k=1}^{N_{\text{el}}} w_i f\left(\frac{\xi_{i+1}-\xi_i}{2}x_i+\frac{\xi_{i+1}-\xi_i}{2}\right).$$

The quadrature rule is easily expanded to squares or cubes. For example, for $p=3$ and integrating over $[-1,1]^2$, the quadrature points are $\{-\sqrt{\frac{3}{5}},0,\sqrt{\frac{3}{5}}\}\times\{-\sqrt{\frac{3}{5}},0,\sqrt{\frac{3}{5}}\}$. The weights are $(\frac{5}{9},\frac{8}{9},\frac{5}{9})\otimes(\frac{5}{9},\frac{8}{9},\frac{5}{9})$. For a rectangle $[\xi_i,\xi_{i+1}]\times[\eta_j,\eta_{j+1}]$ we again will have to use a simple transformation.

Sources: [1], [7], [8], [6].

2.4 Multipatch

So far we have pretended all domains can be mapped onto either the unit square (2D) or the unit cube (3D). But this is of course not the case, in almost all practical circumstances it will be necessary to describe domains with multiple patches.
Most common reason is the situation where the domain simply differs topologically from a square or cube. For example the shape in figure 7, it is clear we can't find invertible $\mathbf{F}$ that maps this domain to the unit square. Instead all coloured are mapped seperately to their own unit squares.
Other reasons to use multiple patches include when different materials or different physical models

Figure 7: Example of a geometry where multiple patches are required. From: [9]

are used in different parts of the domain. Finally it may be computationally efficient to assemble different subdomains in parallel, which is more convenient when using multiple patches.

In multipatch, Ω is divided into a collection of non-overlapping subdomains Ω^k such that $\bar{\Omega} = \bigcup_{k=1}^{K} \bar{\Omega}^k$. We call Ω a multipatch geometry consisting of k patches. For each Ω^k a geometry function $\mathbf{F}^k$ is then defined to parametrize each subdomain individually

$$\mathbf{F}^k : \Omega_0 \to \Omega^k \qquad \mathbf{F}^k(\boldsymbol{\xi}) = \mathbf{x} \in \Omega^k, \ \forall \boldsymbol{\xi} \in \Omega_0.$$

For illustration, consider the multipatch geometry consisting of 4 patches as shown in figure 8. In the same figure you can also see the resulting block structure of the system matrix A.
The first 4 diagonal blocks A_{ii} are associated with the interior degrees of freedom on Ω_i. $A_{\Gamma\Gamma}$ 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.
The resulting block structure is called a block arrowhead matrix. This particular block structure is of course only achieved for particular numberings of the degrees of freedom. The blocks $A_{11}, ..., A_{44}, A_{\Gamma\Gamma}$ will look similar to the matrix A in figure 5c.

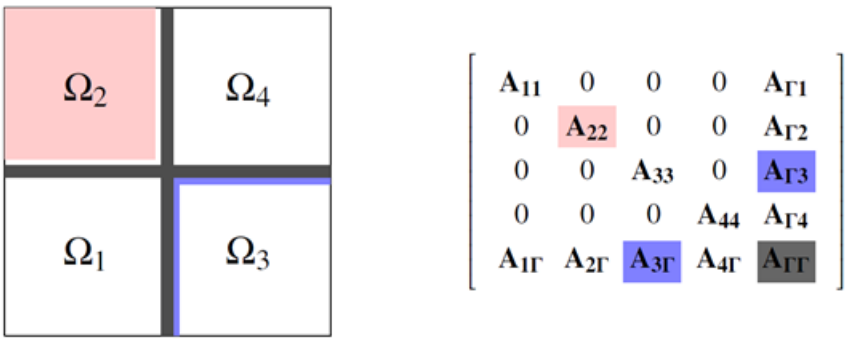


Figure 8: A multipatch geometry, consisting of 4 patches and the resulting block structure of the system matrix. From: [9]

Sources: [1], [9].

3 Multigrid Methods

3.1 Introduction Multigrid

In chapter 2 we described how a boundary value problem leads to a linear system of equations, using IgA. This chapter focusses on solving the resulting linear system. In particular, we focus on multigrid methods.
Multigrid methods aim to solve linear systems by combinining a basic iterative method and a correction which is based on a connected and easier problem. Multigrid is shown to be very efficient, especially for large systems.

The goal is to determine $\mathbf{u}$ in $A\mathbf{u} = \mathbf{f}$, but the dimensions of this problem are too big to do so effectively with just a basic iterative method.
Therefore consider the connected system $\tilde{A}\tilde{\mathbf{u}} = \tilde{\mathbf{f}}$. Here $\tilde{A}$, $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{f}}$ are connected to the original matrix and vectors via a prolongation operator $\mathcal{I}_P$ ($\mathbf{u} = \mathcal{I}_P\tilde{\mathbf{u}}$) and restriction operator $\mathcal{I}_R$ ($\tilde{\mathbf{u}} = \mathcal{I}_R\mathbf{u}$).
$\tilde{A}$ can be obtained by either rediscretizing the bilinear form or by applying so-called Galerkin coarsening: $\tilde{A} = \mathcal{I}_R A \mathcal{I}_P$.

In the remainder of this section various multigrid algorithms are shown, starting with the most basic application of multigrid. As more and more is added these alterations will be explained and motivated.
$\mathbf{v}$ will be used to denote the approximation to $\mathbf{u}$.
Multigrid algorithm 1 shows the use of two different grids and can be used to determine a reasonable initial guess $\mathbf{v}$. It is however of no use in trying to iteratively improving our solution.

ALGORITHM 1
Goal: Determine $\mathbf{u}$ in $A\mathbf{u} = \mathbf{f}$.
1. Determine $\tilde{A}$ and $\tilde{\mathbf{f}}$, using $\tilde{A} = \mathcal{I}_R A \mathcal{I}_P$ and $\mathbf{f} = \mathcal{I}_P\tilde{\mathbf{f}}$.
2. Solve for $\tilde{\mathbf{v}}$, using $\tilde{A}\tilde{\mathbf{v}} = \tilde{\mathbf{f}}$.
3. Determine $\mathbf{v}$, by prolongating $\mathbf{v} = \mathcal{I}_P\tilde{\mathbf{v}}$.

To further improve on an initial guess we use the residual equation. That is, instead of working on $\tilde{A}\tilde{\mathbf{u}} = \tilde{\mathbf{f}}$ we work on $\tilde{A}\tilde{\mathbf{e}} = \tilde{\mathbf{r}}$. We see (approximately) how big the error is and thus how much our guess should be corrected. This leads to multigrid algorithm 2. The sequence can be repeated iteratively to further improve your guess $\mathbf{v}$.

ALGORITHM 2 (Coarse grid correction)
Goal: Determine $\mathbf{u}$ in $A\mathbf{u} = \mathbf{f}$.
1. Compute the residual $\mathbf{r} = \mathbf{f} - A\mathbf{v}$.
2. Restrict the residual $\tilde{\mathbf{r}} = \mathcal{I}_R\mathbf{r}$.
3. Solve $\tilde{A}\tilde{\mathbf{e}} = \tilde{\mathbf{r}}$.
4. Prolongate the error $\mathbf{e} = \mathcal{I}_P\tilde{\mathbf{e}}$.
5. Update the guess $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{e}$.

It is known from literature that coarse grid correction (algorithm 2) is effective in reducing low frequency components of the error. On the other hand basic iterative methods such as Gauss Seidel or (damped) Jacobi reduce the high frequency components. Hence, they are often used together. A typical multigrid method contains both smoothing steps and a coarse grid correction. Algorithm 3 is an example of this. The number of pre- and postsmoothing steps is denoted by ν_1 and ν_2 respectively.

ALGORITHM 3 (Two-grid cycle)
Goal: Determine $\mathbf{u}$ in $A\mathbf{u} = \mathbf{f}$.
1. Relax ν_1 times on $A\mathbf{u} = \mathbf{f}$ with initial guess $\mathbf{v}$.
2. Compute the residual $\mathbf{r} = \mathbf{f} - A\mathbf{v}$.
3. Restrict the residual $\tilde{\mathbf{r}} = \mathcal{I}_R\mathbf{r}$.

4. Solve $\tilde{A}\tilde{\mathbf{e}} = \tilde{\mathbf{r}}$.
5. Prolongate the error $\mathbf{e} = \mathcal{I}_P\tilde{\mathbf{e}}$.
6. Update the guess $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{e}$.
7. Relax ν_2 times on $A\mathbf{u} = \mathbf{f}$ with initial guess $\mathbf{v}$.

The most effective way of solving $\tilde{A}\tilde{\mathbf{e}} = \tilde{\mathbf{r}}$ may be to restrict to an even easier grid. That is, we want to apply the multigrid approach recursively.
This recursive use of multigrid leads to a V-cycle and is described in multigrid program 4. Figure 9a shows the V-cycle scheme in a more visual way.

ALGORITHM 4 (V-cycle)
Goal: Determine $\mathbf{u}$ in $A\mathbf{u} = \mathbf{f}$.
Relax ν_1 times on $A\mathbf{u} = \mathbf{f}$ with initial guess $\mathbf{v}$.
1. Compute $\tilde{\mathbf{f}} = \mathcal{I}_R\mathbf{r}$.
2. Relax ν_1 times on $\tilde{A}\tilde{\mathbf{u}} = \tilde{\mathbf{f}}$ with initial guess $\tilde{\mathbf{v}} = \mathbf{0}$.
3. Compute $\tilde{\tilde{\mathbf{f}}} = \mathcal{I}_R\tilde{\mathbf{r}}$.

⋮

4. Solve $A^*\mathbf{u}^* = \mathbf{f}^*$.

⋮

5. Update $\tilde{\mathbf{v}} \leftarrow \tilde{\mathbf{v}} + \mathcal{I}_P\tilde{\tilde{\mathbf{v}}}$.
6. Relax ν_2 times on $\tilde{A}\tilde{\mathbf{u}} = \tilde{\mathbf{f}}$ with initial guess $\tilde{\mathbf{v}}$.
7. Update $\mathbf{v} \leftarrow \mathbf{v} + \mathcal{I}_P\tilde{\mathbf{v}}$.
8. Relax ν_2 times on $A\mathbf{u} = \mathbf{f}$ using initial guess $\mathbf{v}$.

Next to V-cycles there are several other ways of going through the hierarchy, the main ones being W-cycles, F-cycles and FMG (full multigrid) and shown in figure 9. Trade-off is speed of doing a single iteration versus the expected number of iterations necessary.

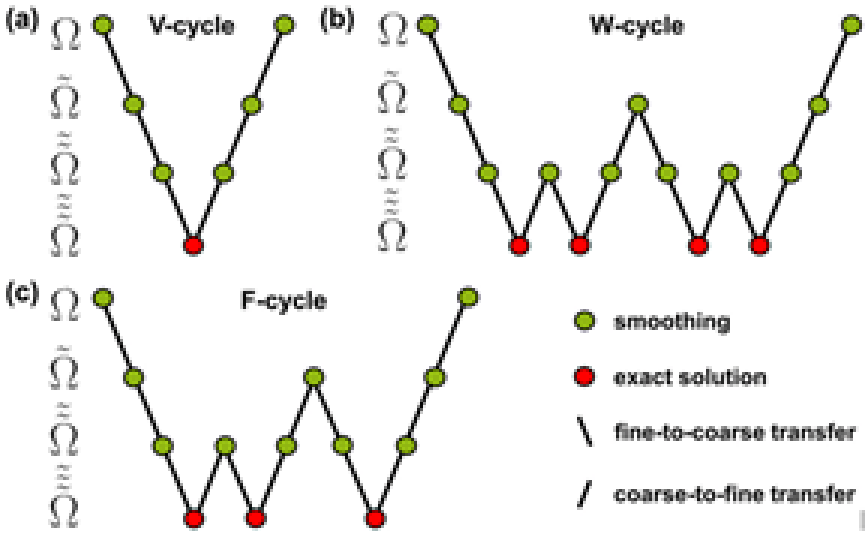


Figure 9: Visual description of a V,W and F-cycle scheme using 4 levels.

Sources: [10], [11].

3.2 h-multigrid

The most common choice for the operator $\tilde{A}$ is the operator corresponding to a coarser discretization. $\tilde{A}$ is the operator resulting from a discretization with a mesh width $2h$ instead of h. Hence why we call it h-multigrid. As a result of the increased mesh width, the matrices and vectors on the coarser level are significantly smaller.
These are its main advantages: it is well known, well researched and we get to work on significantly smaller systems on the coarser levels.

For notation we use A^h, A^{2h}, $\mathbf{v}^h$, $\mathbf{v}^{2h}$, etc. for matrices and vectors on the different grids.
The prolongation operator is denoted $\mathcal{I}_{2h}^h$. This is to be read from te bottom to the top; it transforms vectors over a grid with mesh width $2h$ to vectors over a grid with mesh width h. The restriction operator is denoted $\mathcal{I}_h^{2h}$.
We will show how these operators look like for both 1- and 2-dimensional problems.

3.2.1 Prolongation operator

To prolongate information from the coarse grid to the fine grid within h-multigrid methods, **linear interpolation** is typically adopted. For a 1-dimensional problem this means $\mathcal{I}_{2h}^h \mathbf{v}^{2h} = \mathbf{v}^h$, where

$$\begin{aligned} v_{2j}^h &= v_j^{2h}, \\ v_{2j+1}^h &= \frac{1}{2}(v_j^{2h} + v_{j+1}^{2h}), \qquad 0 \leq j \leq \frac{1}{2}n - 1. \end{aligned}$$

In matrix-vector form a small example would look like

$$\mathcal{I}_{2h}^h \mathbf{v}^{2h} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{pmatrix} = \mathbf{v}^h.$$

When expanding to a 2-dimensional problem, we see $\mathcal{I}_{2h}^h \mathbf{v}^{2h} = \mathbf{v}^h$, where

$$\begin{aligned} v_{2i,2j}^h &= v_{ij}^{2h}, \\ v_{2i+1,2j}^h &= \frac{1}{2}(v_{ij}^{2h} + v_{i+1,j}^{2h}), \\ v_{2i,2j+1}^h &= \frac{1}{2}(v_{ij}^{2h} + v_{i,j+1}^{2h}), \\ v_{2i+1,2j+1}^h &= \frac{1}{4}(v_{ij}^{2h} + v_{i+1,j}^{2h} + v_{i,j+1}^{2h} + v_{i+1,j+1}^{2h}), \qquad 0 \leq i,j \leq \frac{1}{2}n - 1. \end{aligned}$$

3.2.2 Restriction operator

For restriction it may be tempting to simply keep the information on those grid points which are also on the coarser grid, discard the information on those nodes only on the fine grid. This is known as injection: $\mathbf{v}_j^{2h} = \mathbf{v}_{2j}^h$.
But most common and the better choice is **full weighting**, where a weighted average is taken of fine grid nodes. For a 1-dimensional problem this means $\mathcal{I}_h^{2h} \mathbf{v}^h = \mathbf{v}^{2h}$, where

$$v_j^{2h} = \frac{1}{4}(v_{2j-1}^h + 2v_{2j}^h + v_{2j+}^h), \qquad 1 \leq j \leq \frac{1}{2}n - 1.$$

Or in case of a small example in matrix-vector form

$$\mathcal{I}_h^{2h}\mathbf{v}^h = \begin{pmatrix} 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_7 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \mathbf{v}^2 h.$$

For a 2-dimensional problem we can write the operator $\mathcal{I}_h^{2h}$ in stencil notation as

$$\mathcal{I}_h^{2h} = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}_h^{2h}.$$

With this choice of prolongation and restriction operators the variational condition $\mathcal{I}_{2h}^h = c(\mathcal{I}_h^{2h})^T$ is satisfied.

Sources: [10], [11].

3.3 p-multigrid

In p-multigrid the coarser level is based on a lower order discretization. That is, the hierarchy is based on a different order p of the basis functions. A vector $\mathbf{v}^p$ contains the coefficients of the order p basis functions to some B-spline curve, surface or solid. $\mathbf{v}^{p-1}$ contains the coefficients to the same curve, surface or solid, but now using the order $p-1$ basis functions.

3.3.1 L_2-projection

This transformation is a bit more complicated then the transformations we have seen for h-multigrid. Here the prolongation and restriction operators are based on a L_2-projection. To demonstrate the change of bases, suppose we have 2 bases $\mathcal{B}_1 = \{\phi_1, ..., \phi_n\}$ and $\mathcal{B}_2 = \{\psi_1, ..., \psi_m\}$. Also suppose we have a function $\sum_i a_i\phi_i$ lying in the span of basis $\mathcal{B}_1$, which we want to rewrite using the basisfunctions ψ_j.
This means we have to find coefficients b_j such that $\frac{1}{2}||\sum_j b_j\psi_j - \sum_i a_i\phi_i||^2_{L^2}$ is minimized.

$$\frac{1}{2}||\sum_j b_j\psi_j - \sum_i a_i\phi_i||^2_{L^2} = \frac{1}{2}\int(\sum_j b_j\psi_j - \sum_i a_i\phi_i)^2 \ \mathrm{d}\Omega.$$

For this to be minimized, we need $\frac{d}{db_k} = 0$ for all k.

$$\int(\sum_j b_j\psi_j - \sum_i a_i\phi_i)\psi_k \ \mathrm{d}\Omega = 0 \qquad \forall k,$$

$$\int(\sum_j b_j\psi_j)\psi_k \ \mathrm{d}\Omega = \int(\sum_i a_i\phi_i)\psi_k \ \mathrm{d}\Omega \qquad \forall k,$$

$$\sum_j b_j \int \psi_j\psi_k \ \mathrm{d}\Omega = \sum_i a_i \int \phi_i\psi_j \ \mathrm{d}\Omega \qquad \forall k,$$

$$\left(\int \psi_j\psi_k \ \mathrm{d}\Omega\right)_{j,k} \mathbf{b} = \left(\int \phi_i\psi_k \ \mathrm{d}\Omega\right)_{i,k} \mathbf{a},$$

$$M\mathbf{b} = P\mathbf{a},$$

$$\mathbf{b} = M^{-1}P\mathbf{a}.$$

Note that M is an $n \times n$-matrix, whereas P is an $n \times m$-matrix.

Now to apply this on the different order B-spline basis functions. It has been shown that it is effective to directly project to the level $p = 1$ [12]. Therefore only the operators between level p and level 1 need to be specified.
The **prolongation operator** is given by

$$\mathcal{I}_1^p = (M_p)^{-1}P_1^p,$$

the **restriction operator** is given by

$$\mathcal{I}_p^1 = (M_1)^{-1}P_p^1.$$

Here the mass matrices M and transfer matrices P are defined as

$$(M_p)_{i,j} = \int_\Omega \phi_{i,p}\phi_{j,p} \ \mathrm{d}\Omega, \qquad (P_1^p)_{i,j} = \int_\Omega \phi_{i,p}\phi_{j,1} \ \mathrm{d}\Omega,$$

$$(M_1)_{i,j} = \int_\Omega \phi_{i,1}\phi_{j,1} \ \mathrm{d}\Omega, \qquad (P_p^1)_{i,j} = \int_\Omega \phi_{i,1}\phi_{j,p} \ \mathrm{d}\Omega.$$

To avoid having to invert the mass matrices M, its lumped variant $M^L_{i,i} = \sum_j M_{i,j}$ is used. Numerical

experiments show this barely effects the convergence behaviour of the p-multigrid method. By the properties (in particular the partition of unity property, see section 2.2) of B-spline basis functions the lumped matrix is easily calculated.

$$M_{i,i}^L = \sum_j M_{i,j} = \sum_j \int_\Omega \phi_{i,p}\phi_{j,p} \, \mathrm{d}\Omega = \int_\Omega \sum_j \phi_{i,p}\phi_{j,p} \, \mathrm{d}\Omega = \int_\Omega \phi_{i,p} \, \mathrm{d}\Omega.$$

3.3.2 Motivation for p-multigrid

On the level $p = 1$ the system will not be significantly smaller as it was when we used h-multigrid. However, the system will be a lot more sparse and will also look a lot more like other better researched systems. This because the level $p = 1$ equals FEM, which is widely known and widely researched.

As our system on the coarse grid is very similar to a system resulting from a FEM discretization, we can use methods which are known to perform well for FEM. This means we can use something simple like h-multigrid and Gauss Seidel.
That is, after we have gone from level $p = p$ to level $p = 1$, we can go and do h-multigrid on the level $p = 1$. This scheme is visualized in figure 10.

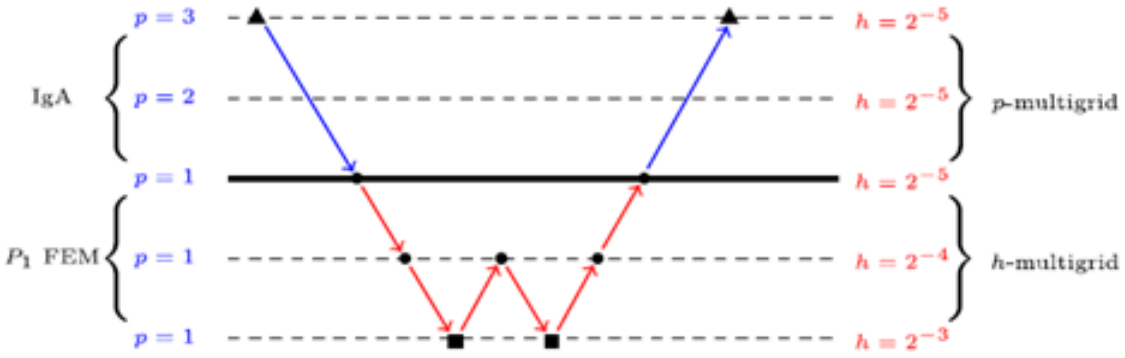


Figure 10: Scheme for combined use of p- and h-multigrid. From: [13]

Sources: [12], [2], [13], [4].

3.4 Smoothers

Within Multigrid methods, a basic iterative method is typically used as a smoother. However, in IgA the performance of classical smoothers such as (damped) Jacobi or Gauss Seidel decreases significantly for higher values of p [14]. Therefore other methods are suggested, such as a smoother using an Incomplete LU factorization.

3.4.1 Incomplete LU factorization

The goal of an incomplete LU factorization is to find sparse lower- and upper triangular matrices L, U such that $A \approx LU$. This is significantly cheaper then full LU factorization, both for computation and storage. The resulting ILU smoother still requires significant setup time, however you should need only one or few smoothing steps.

Two ILU factorizations are suggested: ILU(0) (which is actually part of a wider class of methods ILU(k)) and ILUT. Both are based on the ikj version of Gaussian elimination. ILU(0) only allows for nonzero entries on L, U on those positions A contains nonzero entries. $NZ(A)$ denotes those points where the matrix A contains nonzero entries.

ALGORITHM 1: ILU(0)
Goal: Determine sparse lower and upper triangular L, U such that $LU \approx A$.
1. Start with $L = I$, $U =$ copy of A.
2. for $i = 2, ..., n$:
3. for $k = 1, ..., i-1$ and if $(i,k) \in NZ(A)$:
4. Compute pivot $l_{ik} = u_{ik}/u_{kk}$, set $u_{ik} = 0$.
5. for $j = k+1, ..., n$ and if $(i,j) \in NZ(A)$:
6. Compute $u_{ij} = u_{ij} - l_{ik}u_{kj}$.
7. end for
8. end for
9. end for

The accuracy of the ILU(0) incomplete factorization may be insufficient to yield an adequate rate of convergence. Allowing for more fill in, that is more nonzero entries, may give for better approximations to L and U. The methods that allow for this fill in can be more efficient as well as more reliable. ILUT is similar to ILU(0), except the condition $(i,j) \in NZ(A)$ is replaced by another dropping rule (explained below the main algorithm).

ALGORITHM 2: ILUT
Goal: Determine sparse lower and upper triangular L, U such that $LU \approx A$.
1. Start with L=I, U=copy of A.
2. for $i = 1, ..., n$:
3. $w := a_{i\star}$
4. for $k = 1, ..., i-1$ and if $w_k \neq 0$:
5. $w_k = w_k/a_{kk}$
6. Apply a dropping rule to w_k.
7. if $w_k \neq 0$:
8. Update $w := w - w_k u_{k\star}$.
9. end if
10. end for
11. Apply a dropping rule to row w.
12. $l_{ij} = w_j$ for $j = 1, ..., i-1$.
13. $u_{ij} = w_j$ for $j = i, ..., n$.

14. Set $w = 0$.
15. end for

In the factorization ILUT(m, τ) the following rules are used:
1. In line 6, an element w_k is dropped (i.e. replaced by zero) if it is less then the relative tolerance τ_i, obtained by multiplying τ by the original norm of the i-th row.
2. In line 11, drop again any element in the row with a magnitude below τ_i. Then, only keep the m largest elements in both the L part and the U part of the row, as well as the diagonal element, which is always kept.

Once the factorization is obtained, a single smoothing step is applied as follows:
Calculate error $\mathbf{e} = U^{-1}L^{-1}(\mathbf{f} - A\mathbf{u})$
Update guess $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{e}$.

3.4.2 Alternative smoothers

In this literature review we decided too focus on the ILU type smoothers, however other methods have also been suggested. Recent research includes the use of multiplicative Schwarz smoothers [15], subspace corrected mass smoothers [16], hybrid smoothers [17] and symbol based multigrid smoothers [18].
These aren't further elaborated upon in this literature review.

Sources: [14], [19], [20], [13].
Extra sources: [15], [16], [17], [18].

4 Numerical Results

4.1 1-dimensional problem

In this section we will consider the homogeneous Poisson equation.
For the 1-dimensional case this will reduce to

$$\begin{cases} -u_{xx} = f, \\ u(0) = u(1) = 0. \end{cases}$$

To make everything as easy is possible for the right hand side $f(x) = \pi^2 \sin(\pi x)$ is chosen. This makes $u(x) = \sin(\pi x)$ the exact solution to this problem. Also the geometry function in the IgA discretization is simply the identity. The initial gues will be the zero vector: $u^0 \equiv 0$.

The primary thing we are interested in for our multigrid program is to see how many steps it takes to converge. For this we use the stopping criterium

$$\frac{||\mathbf{r}^k||}{||\mathbf{r}^0||} < 10^{-8}.$$

where $\mathbf{r}^k$ and $\mathbf{r}^0$ are the residuals after k and 0 steps respectively.

4.1.1 h-multigrid + GS

For various combinations of p and N_{dof} we performed this test for a h-multigrid program with Gauss Seidel as a smoother, the results are shown in table 1.
The theory suggests that a multigrid code should be h-independent, that is convergence should be independent of the mesh width. This is reflected in table 1, where for constant p and varying N_{dof} the number of steps stays roughly the same.
The theory also suggests that for $p = 1$, i.e. the finite element case, Gauss Seidel is known to work very well. However for increasing p the performance of traditional smoothers like Gauss Seidel drops rapidly. We do see this in table 1 also, but only from $p = 5$ onwards. In practice one will not go for such high order basis functions anyway. So not entirely the results one would have expected from the literature.

	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
$N_{\text{dof}} = 17$	9	7	6	10	18	49
$N_{\text{dof}} = 33$	10	8	8	10	19	49
$N_{\text{dof}} = 65$	10	8	8	10	20	52
$N_{\text{dof}} = 129$	11	8	8	10	20	54
$N_{\text{dof}} = 257$	11	8	8	11	20	55

Table 1: h-multigrid V-Cycle: steps till convergence (GS smoother, $\nu_1 = \nu_2 = 1$)

4.1.2 p-multigrid + GS

Next a similar test is performed but this time for p-multigrid.
Here a two level method is adopted. We project directly from $p = p$ to $p = 1$, then on the $p = 1$ level a direct solver is used. On the coarse grid the Gauss Seidel smoother is used.
As expected from the literature we see indepence of h (or independence of N_{el}) in table 2, but dependence of p.

4.1.3 p-multigrid + ILUT

In our final test again the two level method is adopted, projecting directly from $p = p$ to $p = 1$ and using a direct solver on the $p = 1$ level. But now, instead of Gauss Seidel an ILUT smoother is used

	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
$N_{\text{el}}=16$	-	2	2	3	6	17
$N_{\text{el}}=32$	-	2	2	3	6	15
$N_{\text{el}}=64$	-	2	2	3	5	13
$N_{\text{el}}=128$	-	2	2	3	5	11
$N_{\text{el}}=256$	-	1	2	2	5	10

Table 2: p-multigrid two level method: steps till convergence (GS smoother, $\nu_1=\nu_2=1$)

(parameters $m=1$,$\tau=10^{\imath 2}$), as this smoother should be independent of p.
We can not see this in table 3 though. No matter the value of p or N_{el}, it always converges within a single iteration. Not a lot of information to be gained from this.
This behaviour is explained by the structure of the system matrix A for the 1-dimensional problem. All information is located very close to the diagonal and because of this the incomplete LU factorization and 'full' LU factorization are equal. So even the pre- and postsmoothing steps are already direct methods, which of course converge in one iteration.

	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
$N_{\text{el}}=16$	-	1	1	1	1	1
$N_{\text{el}}=32$	-	1	1	1	1	1
$N_{\text{el}}=64$	-	1	1	1	1	1
$N_{\text{el}}=128$	-	1	1	1	1	1
$N_{\text{el}}=256$	-	1	1	1	1	1

Table 3: p-multigrid two level method: steps till convergence (ILUT smoother)

4.1.4 Discussion

The goal of this section was to see if I could replicate the different aspects of IgA and multigrid, then hopefully see the patterns I was hoping to see. I would say the first part has been achieved. The second part not quite, the 1-dimensional problem was just to simplified for this.
This is most evident in table 3, caused by the sparsity structure of the matrix A. This will not happen in a 2- or 3-dimensional problem.
Luckily all routines used in this section are easily adjusted for a 2-dimensional Poisson equation. This should be a better test to show IgA and multigrid at work. See section ... (does not exist).

List of symbols

In order of appearance (roughly).

u the unknown in the PDE
f the RHS in the PDE

V function space
V_h finite dimensional subspace
ϕ_1,...,ϕ_n basis for V_h

Ω physical domain
Ω_0 parameter domain
$\mathbf{x}$ point in Ω
$\boldsymbol{\xi}$ point in Ω_0
$\mathbf{F}$ function linking physical and parameter domain
Ω^k, $\mathbf{F}^k$ multipatch variations of Ω and $\mathbf{F}$

A the discretization matrix in the matrix equation
$\mathbf{u}$ the unknown in the matrix equation
$\mathbf{f}$ the RHS in the matrix equation
Can be specified further as $A_{h,p}$, $\mathbf{u}_{h,p}$, $\mathbf{f}_{h,p}$

Ξ, knot vector (if we need to consider multiple knot vectors also H and L)
ξ_i, the i-th knot of Ξ (η_j, ζ_k also as j-th and k-th knot for H and L respectively)

$\phi_{i,p}(\xi)$ the i-th order p basis function to knot vector Ξ
h distance between knots
p order of basis functions
N_{el} number of elements
N_{dof} degrees of freedom, i.e. the amount of basis functions

x_q the quadrature points
w_q the weights of said quadrature points

$\mathbf{v}$ approximation to $\mathbf{u}$
$\mathbf{e}$ general error
$\mathbf{r}$ general residual
Can be specified further as $\mathbf{v}_{h,p}$, $\mathbf{e}_{h,p}$, $\mathbf{r}_{h,p}$, etc.

$\mathcal{I}_h^{2h}$ example of intergrid transfer
$\mathcal{I}_{2h}^{h}$ example of intergrid transfer

M mass matrix, $M_{i,j} = \int_\Omega \phi_i(x)\phi_j(x)\ \mathrm{d}\Omega$
M^L lumped mass matrix

References

[1] J.A. Cottrell, T.J.R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, 2009. ISBN: 9780470749098.

[2] R Tielen et al. "Efficient p-multigrid methods for isogeometric analysis". In: *arXiv preprint arXiv:1901.01685* (2019).

[3] B. Simeon and A. Vuong. *Isogeometric Analysis Primer*.

[4] J. van Kan, A. Segal, and F. Vermolen. *Numerical Methods in Scientific Computing*. Jan. 2005.

[5] T. Hughes, J.A. Cottrell, and Y. Bazilevs. "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement". In: *Computer methods in applied mechanics and engineering* 194 (Oct. 2005), pp. 4135–4195. DOI: 10.1016/j.cma.2004.10.008.

[6] *Gaussian quadrature*. Mar. 2021. URL: https://en.wikipedia.org/wiki/Gaussian_quadrature.

[7] M. Möller. *Assembly strategies in isogeometric analysis*. 2015.

[8] R.G. McClarren. *Computational Nuclear Engineering and Radiological Science Using Python*. 2018. ISBN: 978-0-12-812253-2.

[9] R. Tielen, M. Möller, and Vuik C. "A block ILUT smoother for multipatch geometries in Isogeometric Analysis". In: (2021).

[10] W. Briggs, V. Henson, and S. McCormick. *A Multigrid Tutorial, 2nd Edition*. Jan. 2000. ISBN: 978-0-89871-462-3.

[11] C. Vuik and D.J.P. Lahaye. *Course WI4201, Scientific Computing*. 2017.

[12] 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, Egmond aan Zee, the Netherlands*. 2019.

[13] R. Tielen et al. "p -multigrid methods and their comparison to h -multigrid methods within Isogeometric Analysis". In: *Computer Methods in Applied Mechanics and Engineering* 372 (Dec. 2020), p. 113347. DOI: 10.1016/j.cma.2020.113347.

[14] N. Collier et al. "The Cost of Continuity: Performance of Iterative Solvers on Isogeometric Finite Elements". In: *SIAM Journal on Scientific Computing* 35 (June 2012). DOI: 10.1137/120881038.

[15] A. Riva, C. Rodrigo, and F. Gaspar. "A Robust Multigrid Solver for Isogeometric Analysis Based on Multiplicative Schwarz Smoothers". In: *SIAM Journal on Scientific Computing* 41 (Jan. 2019), S321–S345. DOI: 10.1137/18M1194407.

[16] C. Hofreither and S. Takacs. "Robust Multigrid for Isogeometric Analysis Based on Stable Splittings of Spline Spaces". In: *SIAM Journal on Numerical Analysis* 55 (Jan. 2017), pp. 2004–2024. DOI: 10.1137/16M1085425.

[17] J. Sogn and S. Takacs. "Robust multigrid solvers for the biharmonic problem in isogeometric analysis". In: *Computers and Mathematics with Applications* 77 (Feb. 2018). DOI: 10.1016/j.camwa.2018.09.017.

[18] M. Donatelli et al. "Symbol-Based Multigrid Methods for Galerkin B-Spline Isogeometric Analysis". In: *SIAM Journal on Numerical Analysis* 55(2017) (Jan. 2017), pp. 31–. DOI: 10.1137/140988590.

[19] Y. Saad. "ILUT: A dual threshold incomplete LU factorization". In: *Numerical Linear Algebra with Applications* 1 (July 1994), pp. 387–402. DOI: 10.1002/nla.1680010405.

[20] Y. Saad. *Iterative Methods for Sparse Linear Systems*. 2000.