Error and Stability Analysis for B-spline Finite Element Methods

Hongrui Wang

Department of Mathematics and Statistics

University of Strathclyde

Glasgow, UK

October 2015

This thesis is submitted to the University of Strathclyde for the degree of Doctor of Philosophy in the Faculty of Science.

Acknowledgements

I would like to express my great appreciation to my supervisors Prof. Mark Ainsworth and Prof. Oleg Davydov for their supervisions in the collaborated research where the thesis is based on and all of their supports during the study. My deep gratitude also goes to my supervisor Prof. Des Higham for his guidances in the writing of the thesis and all other advices and helps. I am also deeply grateful to Prof. Xuerong Mao for his helps and supports during my study.

I would like to show my appreciation to the financial support provided by Strathclyde University Scholarship.

Many thanks to my friends Wei Liu, Jiafeng Pan, Lei Zhang, Wenwen Huo, and Change Xu for making the study more enjoyable.

Finally, I would like to say thank you to my father Changyong Wang and mother Wenxian Li for their supports and encouragements through my life. My special thanks also go to Yungting Tzou for her understanding and companion during my study.

Abstract

The thesis studies the approximation properties of splines with maximum smoothness. We are interested in the behaviour of the approximation as the degree of the spline increases (so does its smoothness). By studying B-spline interpolation, we obtain error estimates measured in the semi-norm that are explicit in terms of mesh size, degree and smoothness. This new result also gives a higher approximation order than existing estimations. With the results, we investigate the B-spline finite element approximation with k-refinement, which is a strategy of improving the accuracy by increasing the degree and smoothness. The problem is studied in the setting of heat equations and wave equations. We give B-spline FEM schemes for the problems, and obtain error estimates. Moreover, by proving a Markov-type inequality for splines, where an exact constant is derived, we deduce how the stability of the scheme behaves with the k-refinements. We also improve the efficiency of the schemes for problems with periodic boundary conditions by applying the fast Fourier transform.

The thesis also focuses on developing algorithms for efficiently evaluating the element system matrices in finite element methods with Berstein-Bézier splines as shape functions, where the splines are of arbitrary order and defined on quadrilaterals and hexahedrons. The algorithms achieve the optimal complexity by making use of the sum factorial procedure. We test the algorithms in C++ implementation, and the numerical results illustrate that the optimal cost and expected accuracy are achieved.

Contents

Nomenclature

$.*$	component-wise multiplication of vectors
$[\cdot]_N$	modulo of N
$\boldsymbol{F}$	Fourier matrix
$\mathbb{N}_0, \mathbb{Z}, \mathbb{R}, \mathbb{T}$	natural number including 0, integers, real number and unit circle, respectively
$\mathcal{B}_{h,n}$	operator of B-spline interpolation
$\mathcal{H}^n(\mathbb{T}^d)$, $\mathcal{H}^n(\Omega)$	special Sobolev spaces defined in Section 1.3.4.2
$\mathcal{I}, \mathcal{J}$	index sets
$\mathcal{Q}_{0,n}, \mathcal{Q}_{1,n}$	error of multi-dimensional interpolation defined in (2.4.6)
$\mathrm{Cir}(a_0, \ldots, a_{N-1})$	A circulant matrix whose first row is $(a_0, \ldots, a_{N-1})$
$\mu_{\boldsymbol{i}}^{\boldsymbol{n}}(f, [0,1]^d)$	Bernstein polynomial moments
Ω, $\partial\Omega$	given domain and the boundary of a problem or space
$\otimes$	Kronecker product
$\phi_{\boldsymbol{i},n}(\boldsymbol{x})$, $X^{N,n}([0,1]^d)$	multi-variate tensor product B-splines and its spanned space
$\phi_{i,n}(x)$, $X^{N,n}([0,1])$	B-spline and B-spline space on finite interval, respectively (See Section 1.5.3)

$\psi_{\boldsymbol{i}}^{n,H}$, $\mathcal{P}_K^n$ Bernstein-Bézier spline shape functions on a quadrilateral or hexahedron K and the space spanned by the functions

$\sigma_N(\boldsymbol{i})$ bijective mapping defined in (1.3.4)

$\xi_i, \xi_{\boldsymbol{i}}$ points for interpolation

$B_i^n(x)$,$B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{x})$ Bernstein polynomial of degree n and tensor product Bernstein polynomial of degree n.

$b_n(x)$, $X^n(\mathbb{R})$ cardinal B-spline of degree n and cardinal B-spline space, respectively (see Section 1.5.1)

$b_{\boldsymbol{i},n}(\boldsymbol{x})$, $X^{N,n}(\mathbb{T}^d)$ multi-variate tensor product periodic B-splines and its spanned space

$b_{k,n}(x)$, $X^{N,n}(\mathbb{T})$ periodic B-spline and periodic B-spline space (see Section 1.5.2.)

$D^{\vec{\alpha}}$ derivative in weak sense

D_t^+,D_t^- forward difference and backward difference, respectively

E_0, E_1 operator of relative error defined in (2.7.1)

$L_2([0,T];V)$, $C^k([0,T];V)$ special spaces for the PDEs defined in Section 1.3.4.3

R_h orthogonal projection defined in (3.1.26) and (3.2.8)

$S_{h,n}$ operator of periodic B-spline interpolation

$W_p^k(\Omega)$, $H^k(\Omega)$ standard Sobolev space on Ω with $H^k(\Omega)= W_2^k(\Omega)$

The material from Chapter 3 and 4 was presented at MAFELAP conference in London, June 2013.

Chapter 1

Introduction

1.1 Background and Motivation

The finite element method (FEM) is one of the most important numerical techniques for approximating solutions of differential equations with a broad application in industry and science [71]. The method has the advantage of handling problems in complex geometries and dealing with various constrains. The technique typically involves dividing the domain of the problem into regular shaped sub-domains and then approximating the true solution on each of the sub-domains with appropriate simple functions, which are called shape functions [13]. The approximation is obtained by formulating governing equations locally on each sub-domain from a variational formulation of the problem. Then a set of global equations is constructed by assembling the local equations on each sub-domain under certain constraints of the problem, say boundary conditions. Finally, the numerical solution is achieved by solving the equations. In Section 1.4, we give a more detailed introduction to the Galerkin FEM method.

There has been a wide range of shape functions used for approximation in FEM, for instance, Lagrange polynomials and Bernstein polynomials [95]. Another option is B-splines. A spline is a piecewise polynomial function that possesses certain smoothness at connecting points. The connection points and the union of the sub-intervals are referred to as nodes and meshes, respectively. B-splines are the splines with minimal support for a prescribed smoothness requirement on the nodes. When employed as a basis for spline space in numerical approximation, since B-splines are compactly supported, they possess non-global behaviour, that is, changing a coefficient only affects a local part of the function's shape. B-splines also have the properties of forming a partition of unity and point-wise non-negativity. Moreover, Cox and de Boor's recursive B-spline definition [21, 24] enables a variety of efficient manipulations on splines. For instance, the de Boor algorithm [25, 67] provides an efficient way of value evaluation of spline curves. B-splines play an important role in approximation and geometric modelling. They are applied in data fitting, computer-aided design (CAD), automated manufacturing, and computer graphics.

B-splines have been employed as shape functions in FEM because they can generate smooth approximations, which improves the accuracy for certain problems. For instance, cubic B-splines have been used in the analysis of problems such as vibration on shells, beams and plates [61, 57], and on elastic rods [50]. However, their recent popularity for FEM arises from a technique called isogeometric analysis (IGA) proposed in [45, 19], which lays a foundation for interfacing CAD systems with FEM. The technique selects shape function employed in FEM to be the same types of function utilized to express a geometry in CAD, which are usually B-spline based functions, for example, the non-uniform rational spline (NURBS) and T-splines [19, 67]. Thus, the geometry from CAD is represented exactly in FEM, which greatly simplifies the procedure for discretization and improves the accuracy. In addition, by employing a basis with high degree and continuity, IGA also shows high accuracy and robustness for certain problems such as structural vibrations, wave propagations [18, 72], phase field problem [59, 33], and turbulent flows [4, 9].

The flexibility of B-splines enables efficient use of classical h-refinement and p-refinement [17, 82, 7], the two common strategies to improve the approximation in FEM. h-refinement represents the same geometry with a finer mesh while maintaining the same degree and global smoothness. p-refinement increases the degree and maintains the same mesh and global smoothness. Besides these two approaches, another option of refinement available in B-spline FEM is increasing both the degree and smoothness of the spline with the mesh size fixed, which is referred to as k-refinement in [45]. k-refinement has been shown to outperform the classical p-refinement and h-refinement by extensive spectrum and dissipation analysis on problems of structural vibrations, acoustic and wave propagations [18, 72, 20, 46]. In particular, [30] studies the Kolmogorov n-widths of some types of spline spaces and shows that they are optimal subspaces for approximating certain Sobolev spaces.

Although the approximation property for h-refinement is comprehensively studied [81, 8, 85], the approximation theory for k-refinement is still incomplete. The existing studies include [86] which gives error estimates for the splines of maximum smoothness, but the estimates are only sharp for functions of low smoothness. [22] provides an estimate that is explicit regarding degree and smoothness of a spline. However, an assumption has to be satisfied that the smoothness of the spline is sufficiently low compared with the degree. Particularly, as pointed out in the paper, the interesting case of the spline of maximum smoothness is still open.

This thesis aims to study the problems in this case, that is, the approximation property for the splines of maximum smoothness when the degree and smoothness are increasing. Notice that in this case, smoothness is fixed to be 1 degree lower than the degree, therefore when the degree grows, so does the smoothness. In the following context, we only mention the growth of the degree. In particular, we focus on B-spline FEM schemes using the Galerkin method in the setting of heat equations and wave equations. We develop error estimates using the classic error analysis treatments for the Galerkin methods in [87, 56, 39]. This analysis requires knowledge about how well the shape functions can approximate the exact solution. In our case, we need estimates of the best approximation measured in semi-norms.

Particularly, for demonstrating the behaviour of the approximation as the degree is raised, the estimates are supposed to be dependant on the degree. For this reason, we first study a B-spline interpolation problem to give the estimates of the approximation. Besides the error analysis, a behaviour of the stability when the degree increases is also investigated. By studying a Markov-type inequality for the periodic B-spline, we derive a stability condition for the scheme in the k-refinements. Efficiency is also an important factor when studying FEM. We consider the efficiency of the schemes for the problems with periodic boundary conditions. By utilizing the fast Fourier transform, we derive efficient schemes for these problems. The thesis only considers the standard Galerkin method of the B-spline FEM and the reader may refer to the literature for other approaches such as collocation methods [73, 77], adaptive methods using hierarchical B-splines [90, 49, 12, 76], and mesh-less methods [43, 42].

We also consider algorithms for optimal assembly of element matrices for Berstein-Bézier spline finite element methods. More introduction is given in Chapter 5.

1.2 Thesis Layout

The thesis is arranged as follows. In the following parts of this chapter, we give some preliminaries and overview the Galerkin method, the definitions and properties of B-splines.

In Chapter 2, we study the approximation properties regarding B-spline interpolation of periodic functions, and then generalize the result to the non-periodic case. We also give some experiments to justify our theoretical conclusions.

In Chapter 3, we study the B-spline FEM approximation for heat equations, where two kinds of boundary conditions are considered, periodic and Dirichlet. We first review the heat equations of interest and then derive a B-spline scheme for the problems. We also analyse the stability of the scheme and the approximation error. Finally, we conduct some experiments justifying the result regarding the analysis.

In Chapter 4, we consider the approximation for wave equations. Analogously to the study of heat equations, we investigate the stability and error estimation, and at last compare the analysis with the result of experiments.

In Chapter 5, we give an algorithm for efficiently evaluating the load vector, mass matrix and stiffness matrix when using Berstein-Bézier spline function as a basis for the finite element method. We first introduce the Berstein-Bézier spline on a quadrilateral and hexahedron respectively. Then we develop the algorithms for evaluating Bernstein polynomial moments on $[0,1]^d$, which is used to develop algorithms for evaluating the load vector, mass matrix and stiffness matrix. Finally, we check the experimental results based on our C++ implementation.

Appendix A.1 contains samples of the Matlab codes of schemes for the periodic heat equation in one-dimensional. Appendix A.2 contains samples of the C++ codes for evaluating Bernstein-Bézier element matrices on a hexahedron.

1.3 Preliminary

1.3.1 Fast Fourier Transform

The fast Fourier transform (FFT) is an algorithm for efficiently evaluating the discrete Fourier transform of a vector $\vec{v} \in \mathbb{C}^N$, which is also viewed as the product of Fourier matrix $\boldsymbol{F} \in \mathbb{C}^{N \times N}$ and $\vec{v}$. Define a $N \times N$ Fourier matrix $\boldsymbol{F}$ as

$$\boldsymbol{F}_{i,j} = \frac{1}{\sqrt{N}} e^{-ij2\pi\mathrm{i}/N}. \tag{1.3.1}$$

The product $\boldsymbol{F}\vec{v}$ is formed by the FFT in $\mathcal{O}(N \log N)$ operations. For more background about various FFT algorithms, the reader may see, for example, [60, p. 206],[70].

1.3.2 Circulant Matrix

For $a_0, \dots, a_{N-1} \in \mathbb{C}$, a $N \times N$ circulant matrix (see [37, p. 31]) is characterized as a square matrix $\boldsymbol{A}$ with (i, j)th entry

$$\boldsymbol{A}_{i,j} = a_{(j-i) \bmod N},$$

with the form

$$\boldsymbol{A} = \begin{bmatrix} a_0 & a_1 & \dots & a_{N-2} & a_{N-1} \\ a_{N-1} & a_0 & a_1 & & a_{N-2} \\ \vdots & a_{N-1} & a_0 & \ddots & \vdots \\ a_2 & & \ddots & \ddots & a_1 \\ a_1 & a_2 & \dots & a_{N-1} & a_0 \end{bmatrix}.$$

From the definition, it's obvious that the whole matrix is able to be constructed with only the first row of the matrix. We denote such a circulant matrix by $\text{Cir}(a_0, \dots, a_{N-1})$ or $\text{Cir}(a_0, \dots, a_{N-1})$. The following are some standard properties of circulant matrices which are used in our analysis (see [93] for more details).

Property 1. *(i) The sum or product of two circulant matrices is still a circulant matrix, and the inverse of a circulant matrix is still circulant.*

(ii) The eigenvalues of a circulant matrices are given by

$$\lambda_j = a_0 + a_1 w_j + a_2 w_j^2 + \dots + a_{N-1} w_j^{N-1}, \; j = 1, \dots, N-1,$$

where $w_j = \exp(2\pi \mathrm{i} j/n), \mathrm{i} = \sqrt{-1}$. The corresponding eigenvectors are given by

$$v_j = (1, w_j, w_j^2, \dots, w_j^{N-1}).$$

A circulant matrix is diagonalizable with the Fourier matrix [60, p. 206],

$$\boldsymbol{A} = \boldsymbol{F}^{-1} \boldsymbol{\Lambda}_A \boldsymbol{F}. \tag{1.3.2}$$

Matrices $\boldsymbol{\Lambda}_A$ are diagonal matrices, whose diagonal entries are corresponding eigenvalues of $\boldsymbol{A}$. It follows from the Property 1(ii) in Page 7 that the ith diagonal entry of $\boldsymbol{\Lambda}_A$ is given by ith element of the vector $\boldsymbol{F}\vec{a}^T$, where $\vec{a} = (a_0, \ldots, a_{N-1})$ is the first row of $\boldsymbol{A}$.

When calculating the product of a circulant matrix $\boldsymbol{A}$ and a vector $\vec{v}$, a straightforward approach would cost $\mathcal{O}(N^2)$ operations. However the FFT can be applied to make the calculation more efficiently by making use of (1.3.2). For a vector $\vec{a} = (a_1, \ldots, a_d)$ and $\vec{b} = (b_1, \ldots, b_d)$, define the component-wise product to be

$$\vec{a}. * \vec{b} = (a_1 b_1, \ldots, a_d b_d). \tag{1.3.3}$$

The task of computing $\boldsymbol{A}\vec{v}$ is summarized in the following steps:

1. $\vec{f} = \boldsymbol{F}\vec{v}$
2. $\vec{g} = \boldsymbol{F}\vec{a}^T$
3. $\vec{z}^T = \vec{f}. * \vec{g}$
4. $\vec{y} = \boldsymbol{F}^{-1}\vec{z}$.

Since the FFT can be used to perform the steps 1,2 and 4 in $\mathcal{O}(N \log N)$ operations, the overall cost of forming the product $\boldsymbol{A}\vec{v}$ is reduced to $\mathcal{O}(N \log N)$ operations. When it comes to evaluating the product of a $N \times N$ circulant matrix and an arbitrary $N \times N$ matrix, by viewing the matrix product as N matrix-vector products, FFT is still applicable, which reduce the computation down to $\mathcal{O}(N^2 \log N)$.

1.3.3 Kronecker Product

Let matrices $\boldsymbol{A} = (a_{ij}) \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} = (b_{ij}) \in \mathbb{R}^{r \times s}$. The Kronecker product of $\boldsymbol{A}$ and $\boldsymbol{B}$ is defined to be

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \cdots & a_{mn}\boldsymbol{B} \end{bmatrix}.$$

The Kronecker product has the following properties (see [36] and [83]):

Property 2. *(i) For $\boldsymbol{j} = (j_1, \ldots, j_d)$, define the bijective mapping $\sigma_N : \mathbb{N}_0^d \to \mathbb{N}_0$ as*

$$\sigma_N(\boldsymbol{j}) = j_d N^{d-1} + j_{d-1} N^{d-2} + \cdots + j_1. \tag{1.3.4}$$

Assume $\boldsymbol{A}_k \in \mathbb{R}^{N\times N}$, $i = \sigma_N(i_1, \ldots, i_d)$ *and* $j = \sigma_N(j_1, \ldots, j_d)$, *where* σ_N *is defined in* (1.3.4). *Then*

$$\boldsymbol{A} = \boldsymbol{A}_d \otimes \boldsymbol{A}_{d-1} \otimes \cdots \otimes \boldsymbol{A}_1$$

is equivalent to

$$\boldsymbol{A}_{i,j} = \boldsymbol{A}_{d;i_d,j_d}\boldsymbol{A}_{d-1;i_{d-1},j_{d-1}} \ldots \boldsymbol{A}_{1;i_1,j_1}.$$

(ii) $(\boldsymbol{A} \otimes \boldsymbol{B}) \otimes \boldsymbol{C} = \boldsymbol{A} \otimes (\boldsymbol{B} \otimes \boldsymbol{C})$.

(iii) If $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_d$ *are invertible, then* $\boldsymbol{B} = \boldsymbol{B}_d \otimes \boldsymbol{B}_{d-1} \otimes \cdots \otimes \boldsymbol{B}_1$ *are invertible, and* $\boldsymbol{B}^{-1} = \boldsymbol{B}_d^{-1} \otimes \boldsymbol{B}_{d-1}^{-1} \otimes \cdots \otimes \boldsymbol{B}_1^{-1}$.

(iv) If $\boldsymbol{A}_k$ *is a* $m_k \times n_k$ *matrix and* $\boldsymbol{B}_k$ *is a* $n_k \times r_k$ *matrix, then* $\boldsymbol{A}\boldsymbol{B} = (\boldsymbol{A}_d \otimes \boldsymbol{A}_{d-1} \otimes \cdots \otimes \boldsymbol{A}_1)(\boldsymbol{B}_d \otimes \boldsymbol{B}_{d-1} \otimes \cdots \otimes \boldsymbol{B}_1) = (\boldsymbol{A}_d\boldsymbol{B}_d) \otimes (\boldsymbol{A}_{d-1}\boldsymbol{B}_{d-1}) \otimes \cdots \otimes (\boldsymbol{A}_1\boldsymbol{B}_1)$.

(v) If $\boldsymbol{A}$ *and* $\boldsymbol{B}$ *are* $m\times m$ *and* $n \times n$ *matrices, the Kronecker sum of two matrices* $\boldsymbol{A}$ *and* $\boldsymbol{B}$ *is defined to be* $\boldsymbol{A} \oplus \boldsymbol{B} = \boldsymbol{A} \otimes \boldsymbol{I}_m + \boldsymbol{I}_n \otimes \boldsymbol{B}$, *where* $\boldsymbol{I}_m$ *is the identity matrix. If* $\boldsymbol{A}$ *has eigenvalue* $\lambda_{a,i}$ $(i = 1, 2, \ldots, m)$ *and* $\boldsymbol{B}$ *has eigenvalue* $\lambda_{b,j}$, $j = 1, 2, \ldots, n)$ *then* $\boldsymbol{A} \oplus \boldsymbol{B}$ *has* mn *eigenvalues, which are all the possible sums of* $\lambda_{a,i}$ *and* $\lambda_{b,j}$.

1.3.4 Function Spaces

We first give the definition of weak derivative. Some spaces considered in the thesis are introduced here. For more detail, the reader may refer to [64, 13].

1.3.4.1 Sobolev Spaces

Given a domain $\Omega \in \mathbb{R}^d$, the set of locally integrable functions is denoted by

$$L^1_{loc}(\Omega) := \{f : f \in L^1(K)\ \forall \text{ compact } K \subset \text{interior } \Omega\}.$$

For $\vec{\alpha} = (\alpha_1, \ldots, \alpha_d)$, the weak derivative of $f \in L^1_{loc}(\Omega)$, $D^{\vec{\alpha}} f$ exists provided there exists a function $g \in L^1_{loc}(\Omega)$ such that

$$\int_\Omega D^{\vec{\alpha}} f(x)\phi(x)dx = (-1)^{|\vec{\alpha}|} \int_\Omega f(x)\phi^{(\vec{\alpha})}(x)dx \quad \forall \phi \in C_c^\infty(\Omega),$$

where $C_c^\infty(\Omega)$ is the set of C^∞ functions with compact support in Ω and $\phi^{(\vec{\alpha})}$ is the partial derivative,

$$\frac{\partial^{\alpha_1}}{\partial x_1^{\alpha_1}} \cdots \frac{\partial^{\alpha_d}}{\partial x_d^{\alpha_d}} \phi.$$

Let $f \in L^1_{loc}(\Omega)$. If the weak derivative $D^{\vec{\alpha}}$ exists for all $|\vec{\alpha}| \leq k$, the Sobolev norm is defined to be

$$\|f\|_{W_p^k} := (\sum_{|\vec{\alpha}| \leq k} \|D^{\vec{\alpha}} f\|^p_{L^p(\Omega)})^{1/p}.$$

The Sobolev space is defined to be

$$W_p^k(\Omega) := \{f \in L^1_{loc}(\Omega) : \|f\|_{W_p^k} < \infty\}.$$

We denote by $H^k(\Omega)$ the Sobolev space $W_2^k(\Omega)$. Let $|\cdot|_k$ be the semi-norm, that is

$$|f|_k := (\sum_{|\vec{\alpha}| = k} \|D^{\vec{\alpha}} f\|^p_{L^p(\Omega)})^{1/p}.$$

For a vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$, define the norm $\|\vec{\alpha}\|_\infty = \max_{i=1,\ldots,d}\{\alpha_i\}$. By writing $\vec{\alpha} \leq n$, we mean $\|\vec{\alpha}\|_\infty \leq n$. Define $\mathcal{H}^n(\Omega)$ be the space consisting of all functions $f \in L_2(\Omega)$ with norm $\|D^{\vec{\alpha}} f\|_{L_2(\Omega)} < \infty$ for all $\alpha \leq n$. We also define the space $\mathcal{H}_0^n(\Omega)$ to be the completion of $C_c^\infty(\Omega)$ on $\mathcal{H}^n(\Omega)$.

1.3.4.2 Periodic Sobolev Spaces

Let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ be a circle (see [27, p.1]), which is understand to be $\mathbb{R}$ with identification of points with modulo 1. Let $L_2(\mathbb{T}^d)$ be Lebesgue space of functions defined on $\mathbb{T}^d$. Note that the integral on $\mathbb{T}^d$ is the integral on a period, that is,

$$\int_{\mathbb{T}^d} f(\boldsymbol{x})d\boldsymbol{x} = \int_{[0,1]^d} f(\boldsymbol{x})d\boldsymbol{x}.$$

We denote by $\|\cdot\|$ the norm, and the bilinear form in the space $L_2(\mathbb{T}^d)$ by

$$(u,v) := \int_{\mathbb{T}^d} u(\boldsymbol{x})v(\boldsymbol{x})d\boldsymbol{x} \text{ and } a(u,v) := \int_{\mathbb{T}^d} \nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})d\boldsymbol{x}.$$

Let $\mathcal{H}^n(\mathbb{T}^d)$ be the space of all functions $f \in L_2(\mathbb{T}^d)$ with norm $\|D^{\vec{\alpha}}f\| < \infty$ for all $\alpha \leq n$. Note that $\mathcal{H}^n(\mathbb{T}) = H^n(\mathbb{T})$.

1.3.4.3 Space $L_2([0,T];V)$ and $C^k([0,T];V)$

Given a Banach space V with norm $\|\cdot\|_V$, denote by $L_2([0,T];V)$ the space of functions $f : [0,T] \to V$ with norm

$$\|f\|_{L_2([0,T];V)} = \left(\int_0^T \|f(t)\|_V^2 \mathrm{d}t\right)^{1/2}.$$

The dual space of $L_2([0,T];V)$ is $L_2([0,T];V^*)$, where V^* is the dual space of V.

We also define $C^k([0,T];V)$, $k \in \mathbb{N}_0$ consisting of all continuous functions $u : [0,T] \to V$ that have continuous derivatives up to order k on $[0,T]$.

1.3.5 The Constant C

Throughout the thesis, C denotes a general constant that is independent of mesh size h, the degree of the spline p, and is not necessarily the same at different occasions.

1.4 Overview of the Finite Element Method

In this section, we give a brief introduction to the Galerkin finite element method. For more detailed discussion, the reader my consult the literature, for example, [96, 63, 13].

1.4.1 Ritz-Galerkin Method

We first review the Ritz-Galerkin method for approximating a variational problem. Let V be a Hilbert space with inner product $(\cdot,\cdot)$ and norm $\|\cdot\|$, $a(\cdot,\cdot)$ be a bilinear form, and L be a functional in V'. Consider the variational form: find $u \in V$ such that

$$a(u, v) = \langle L, v\rangle \text{ for any } v \in V. \tag{1.4.1}$$

We say $a(\cdot, \cdot)$ is continuous if there exist an constant α such that $|a(u, v)| < \alpha\|u\|\|v\|$ for all $u, v \in V$. We say $a(\cdot, \cdot)$ is coercive if there exist an constant $\beta > 0$ such that $|a(u, u)| > \beta\|u\|^2$ for all $u \in V$. The Lax Milgram theorem (see [29]) states that if $a(\cdot, \cdot)$ is continuous and coercive , then problem (1.4.1) has a unique solution for the equation.

Let $V_h = \{\phi_i\}_{i\in\mathcal{J}} \subset V$ be a finite dimensional space, where $\mathcal{J}$ is an index set. An approximation from V_h to the solution of the weak formulation is obtained by finding $u_h \in V_h$ such that

$$a(u_h, v) = \langle L, v\rangle \qquad \text{for any } v \in V_h. \tag{1.4.2}$$

Letting $u_h = \sum\limits_{j\in\mathcal{J}} \alpha_j\phi_j$ and choosing $v = \phi_i$ for each $i \in \mathcal{J}$ gives

$$\sum_{j\in J} \alpha_j a(\phi_j, \phi_i) = \langle L, \phi_i\rangle .$$

This leads to a series of equations which may be written as

$$\boldsymbol{A}\vec{\alpha} = \vec{l}, \tag{1.4.3}$$

with the entries $\boldsymbol{A}_{i,j} = a(\phi_i, \phi_j)$, $\vec{l_j} = \langle L, \phi_j\rangle$ and $\vec{\alpha} = (\alpha_j)$. Since the finite dimensional space V_h is a Hilbert space, according to the Lax Milgram theorem, the equation has a unique solution. u_h is an approximation in the space V_h to the solution $u \in V$.

1.4.2 Discretization

Let Ω be the domain for functions belonging to the space V in equation (1.4.1) and $\Sigma_\Omega = \{K_i\}_{i=1}^N$ be a subdivision, which is a set of sub-domains such that the union of all K_i is Ω and the intersection of the interior of any two different K_i is empty.

In finite element analysis, the space V_h in (1.4.2) is constructed by discretizing Ω, and assembling finite elements. We define the finite element here following Cialet's definition in [13]. $(K, \mathcal{P}, \mathcal{N})$ is called a finite element if

(i) $K \in \mathbb{R}^n$ is a bounded closed set with non-empty interior and piecewise smooth boundary,

(ii) $\mathcal{P} = \{\phi_1^K, \ldots, \phi_k^K\}$ is a finite-dimensional space of functions on K, and

(iii) $\mathcal{N} = \{N_1, N_2, \ldots, N_k\}$ is a basis for the dual space of $\mathcal{P}'$. The set K is called an element domain, $\{\phi_1^K, \ldots, \phi_k^K\}$ are call shape functions, and $\{N_1, \ldots, N_k\}$ are called nodal variables. Thus, for each sub-domain K_i of Ω, we have a corresponding finite element. Then the space V_h is obtained by assembling the shape functions on each element domain K_i. Some constraints such as functions in space V_h satisfying certain prescribed smoothness or boundary conditions may also be imposed.

1.4.3 Implementation

In practice, a common approach to produce the shape functions efficiently on an element domain is the transformation of a reference finite element using an isoparametric mapping (see [13]). The linear system (1.4.3) is constructed by deriving the element matrices on each of the domains K_i first, and then assembling them along with the smoothness requirements and boundary conditions. For more detail, the reader may see [96].

1.5 Overview of B-splines

We only consider the uniform B-spline with maximum smoothness, where the maximum smoothness means that the spline with degree n is globally $n-1$ times continuously differentiable, which will be simply called a B-spline in the following treatment, since that is the only spline considered in the thesis. We shall first review the cardinal splines [79] which are the B-splines on the uniform mesh on

$\mathbb{R}$, since they have nice definitions in convolution form, which are frequently employed in our analysis. Based on the cardinal splines, we define periodic B-splines and derive some similar properties. At last, we introduce the general recurrence definition of B-spline.

1.5.1 Cardinal B-spline

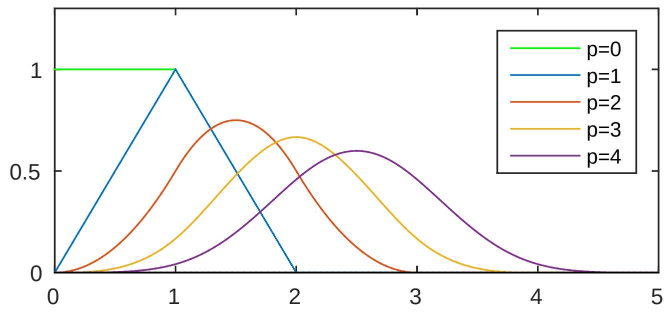


Figure 1.1: Cardinal B-splines of degree p from 0 to 4

A cardinal B-spline with degree n is defined recurrently in convolution as

$$b_n(x) = (b_{n-1} * b_0)(x),$$

with b_0 to be

$$b_0(x) = \begin{cases} 1 & x \in (0,1] \\ 0 & x \notin (0,1] \end{cases} .$$

We call the space spanned by all the cardinal splines on $\mathbb{R}$ a cardinal spline space, denoted by $X^n(\mathbb{R}) = \text{span}\{b_n(x-i), i \in \mathbb{Z}\}$. Figure 1.1 gives a example of cardinal B-splines of different degrees. Here are their basic properties, which can also be found for instance in [42], [81], [79, p . 11] and [16].

Property 3.

(i) $b_n(x)$ *is non-negative for any* n *and* x *and its support is* $[0, n+1]$.

(ii) *$b_n(x)$ is symmetric in the sense that*

$$b_n(x) = b_n(n+1-x),$$

and strictly monotonically increasing on $[0,(n+1)/2]$ and decreasing on $[(n+1)/2, n+1]$.

(iii) *$b_n(x)$ is infinitely differentiable in a subinterval and $b_n(x)$ is $n-1$ times continuously differentiable at a node within its support.*

(iv) *A cardinal spline has the convolution form*

$$\int_{-\infty}^{+\infty} b_m(t-x)b_n(t)dt = b_{m+n+1}(n+1-x).$$

(v) *For $f \in H^{n+1}([0,n+1])$ and $k = 0,\ldots,n+1$, we have*

$$\int_{-\infty}^{+\infty} b_{k-1}(x)f^{(k)}(x)dx = \left(D^{+}\right)^{k} f(0),$$

where $D^{+}f(x) = f(x+1) - f(x)$.

(vi) *The derivative of a basis function can be written as*

$$b_n(x)' = b_{n-1}(x) - b_{n-1}(x-1).$$

(vii) *$b_n(x)$ is a weighted combination of $b_{n-1}(x)$ and $b_{n-1}(x-1)$*

$$b_n(x) = \frac{x}{n} b_{n-1}(x) + \frac{n+1-x}{n} b_{n-1}(x-1).$$

(viii) *The Fourier transform for b_n is $\widetilde{b_n} = \frac{1}{\sqrt{2\pi}}\left(sinc(x/2)\right)^{n+1}$, where $sinc(x) = \sin(x)/x$.*

1.5.2 Periodic B-spline

In the thesis, periodic B-splines refer to the B-splines under periodic boundary conditions. Since periodic B-splines with arbitrary period can be derived by scaling

1-period B-splines, without loss of generality, we only discuss this case and simply call them periodic B-splines. Assume the interval [0,1] is divided by N subintervals, and denote the mesh size by $h = 1/N$. For any $k \in \mathbb{Z}$, a periodic B-spline of degree n is defined as

$$b_{k;n,N}(x) = \sum_{i \in \mathbb{Z}} b_n(x/h - k + i/h),$$

where $b_n(x)$ is a cardinal spline as defined in Section 1.5.1. Throughout the thesis, the number of subintervals is always assumed to be N; for the sake of simplicity, we use a notation $b_{k,n}(x)$ instead of $b_{k;n,N}(x)$. It follows immediately from this definition that $b_{k,n}(x)$ is 1-periodic and can be seen as defined on the unit circle $\mathbb{T}$. We define the periodic B-spline space by $X^{N,n}(\mathbb{T})$=span$\{b_{k,n}\}_{k\in\mathbb{Z}}$. Since $b_{k,n}(x) = b_{k+N,n}(x)$, the space is of dimension N and has a basis $\{b_{i,n}\}_{i\in\mathcal{I}}$, where the index set $\mathcal{I} = \{0, \dots, N-1\}$. From the definition of the cardinal B-spline, we also have that

$$b_{i,n}(x) = \int_{x-h}^{x} b_{i,n-1}(t)dt. \tag{1.5.1}$$

The smoothness and periodicity of $b_{k,n}(x)$ implies $X^{N,n}(\mathbb{T})$ is a subspace of $H^n(\mathbb{T})$. Figure 1.2 gives an example of a B-spline basis with degree 2.

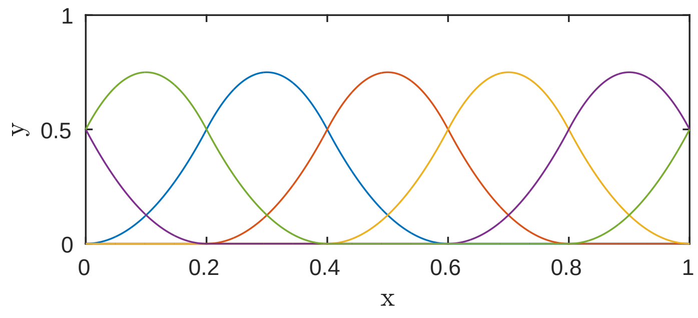


Figure 1.2: Basis of periodic B-splines with degree 2 and 5 sub-intervals.

The periodic B-spline $b_{k,n}(x)$ possesses some similar properties as the cardinal spline $b_n(x)$ as follows.

Property 4. *(i) For $x \in \mathbb{R}$, $b_{k,n}(x)$ is non-negative for any n and x and its support is $[kh + i, (k + n + 1)h + i], i \in \mathbb{Z}$.*

(ii) B-spline is periodic in the sense that, for $k \in \mathbb{Z}$,

$$b_{k,n}(x) = b_{k+N,n}(x).$$

(iii) B-spline is symmetric in the sense that

$$b_{k,n}(x) = b_{-(n+1+k),n}(-x).$$

(iv) For any $i, j \in \mathbb{Z}$,

$$\int_0^1 b_{i,m}(x+t) b_{j,n}(t) dt = h b_{i-j-(n+1),m+n+1}(x).$$

(v) For $f \in H^{n+1}(\mathbb{T})$, *we have*

$$\int_0^1 b_{i,k-1}(x) f^{(k)}(x) dx = 1/h^{k-1} (D_h^+)^k f(ih),$$

where $D_h^+ f(x) = f(x+h) - f(x)$.

(vi) The derivative of $b_{k,n}$ *is a linear combination of* $b_{k,n-1}$ *and* $b_{k+1,n-1}$

$$b'_{k,n}(x) = \frac{1}{h} b_{k,n-1}(x) - \frac{1}{h} b_{k+1,n-1}(x).$$

Proofs. Proof of (i) follows directly from Property 3(i) in Page 14.

Proof of (ii):

The definition of $b_{k,n}(x)$ gives

$$\begin{aligned} b_{k,n}(x) &= \sum_{i \in \mathbb{Z}} b_n(x/h - k + i/h) \\ &= \sum_{i \in \mathbb{Z}} b_n(x/h - k + (i-1)/h) \\ &= \sum_{i \in \mathbb{Z}} b_n(x/h - k - N + i/h) \\ &= \sum_{i \in \mathbb{Z}} b_{k+N,n}(x). \end{aligned}$$

Proof of (iii):

From the definition of $b_{k,n}(x)$ and Property 3(ii) in Page 14,

$$\begin{aligned} b_{k,n}(x) &= \sum_{i\in\mathbb{Z}} b_n(x/h-k+i/h) \\ &= \sum_{i\in\mathbb{Z}} b_n(n+1-(x/h-k+i/h)) \\ &= \sum_{i\in\mathbb{Z}} b_n(-x/h+k+n+1-i/h) \\ &= b_{-(n+1+k),n}(-x). \end{aligned}$$

Proof of (iv):

Since $b_{k,n}$ is periodic, its definition and (iii) imply

$$\begin{aligned} \textstyle\int_0^1 b_{i,m}(x+t)b_{j,n}(t)dt &= \textstyle\int_0^1 b_{i-j,m}(x+t)b_{0,n}(t)dt \\ &= \textstyle\int_0^1 b_{-(m+1+i-j),m}(-x-t)b_{0,n}(t)dt. \end{aligned}$$

On the interval $[0,1]$, $b_{0,n}(t)$ is identical to the cardinal spline $b_n(t/h)$. Then the equality is expressed as

$$\int_0^1 b_{-(m+1+i-j),m}(-x-t)b_{0,n}(t)dt = \int_0^1 \sum_{k\in\mathbb{Z}} b_m(-(x+t)/h+m+1+i-j+k/h)b_n(t/h)dt.$$

Changing the variable by $t = h\tau$, since $b_n(x)$ vanishes outside the interval $[0,1/h]$, we have

$$\begin{aligned} &\textstyle\int_0^1 \sum_{k\in\mathbb{Z}} b_m(-(x+t)/h+m+1+i-j+k/h)b_n(t/h)dt \\ &= h\textstyle\int_0^{1/h} \sum_{k\in\mathbb{Z}} b_m(-x/h-\tau+m+1+i-j+k/h)b_n(\tau)d\tau \\ &= h\sum_{k\in\mathbb{Z}} \textstyle\int_{-\infty}^{+\infty} b_m(-x/h+m+1+i-j+k/h-\tau)b_n(\tau)d\tau. \end{aligned}$$

Moreover, Property 3(iv) and Property 3(iii) give

$$\begin{aligned} &h\sum_{k\in\mathbb{Z}} \textstyle\int_{-\infty}^{+\infty} b_m(-x/h+m+1+i-j+k/h-\tau)b_n(\tau)d\tau \\ &= h\sum_{k\in\mathbb{Z}} b_{m+n+1}(-x/h+m+1+i-j+k/h) \\ &= hb_{-(m+1+i-j),m+n+1}(-x) \\ &= hb_{-(n+1)+i-j,m+n+1}(-x), \end{aligned}$$

which completes the proof.

Proof of (v):

Since both $b_{i,k-1}(x)$ and f are of period 1, we have

$$\begin{aligned}\int_0^1 b_{i,k-1}(x)f^{(k)}(x)dx &= \int_0^1 b_{0,k-1}(x-ih)f^{(k)}(x)dx \\ &= \int_0^1 b_{0,k-1}(x)f^{(k)}(x+ih)dx.\end{aligned}$$

The fact that $b_{0,k-1}(x)$ vanishes outside the interval $[0,1]$ shows

$$\int_0^1 b_{0,k-1}(x)f^{(k)}(x+ih)dx = \int_0^1 b_{k-1}(x/h)f^{(k)}(x+ih)dx.$$

Let $g(x) = f(hx+ih)$ and change the variable with $z = x/h$. It follows that

$$\int_0^1 b_{k-1}(x/h)f^{(k)}(x+ih)dx = 1/h^{k-1}\int_0^{1/h} b_{k-1}(z)g^{(k)}(z)dz.$$

Property 3(v) implies that

$$\int_0^k b_{k-1}(x)g^{(k)}(x)dx = (D^+)^k g(0).$$

Thereupon we have

$$\begin{aligned}\int_0^1 b_{i,k-1}(x)f^{(k)}(x)dx &= 1/h^{k-1}(D^+)^k g(0) \\ &= 1/h^{k-1}(D_h^+)^k f(ih).\end{aligned}$$

Proof of (vi):

It follows from Property 3(vi) that

$$\begin{aligned}b'_{k,n}(x) &= \sum_{i\in\mathbb{Z}} 1/hb'_n(x/h-k+i/h) \\ &= 1/h\sum_{i\in\mathbb{Z}}\big(b'_{n-1}(x/h-k+i/h) - b'_{n-1}(x/h-k+i/h-1)\big) \\ &= 1/hb'_{k,n-1}(x) - 1/hb'_{k+1,n-1}(x).\end{aligned}$$

□

1.5.3 B-spline on Finite Interval

We would like to review de Boor's recursive definition of B-splines. This is a more general definition which is able to define the cardinal and periodic B-splines as discussed before. The definition is widely used in implementation since it facilitates efficient computation of B-splines.

To begin with, we define a knots vector, which determines the span and smoothness of the B-splines. A knots vector is a vector $\Xi = (\zeta_1, \ldots, \zeta_{N+2n+1})$, where $\zeta_1 \leq \ldots \leq \zeta_{N+2n+1}$. Note that there may be repeated values in the vector. We say a knot ζ_i has multiplicity k if there are k elements in the vector with the same value as ζ_i. For instance, it has multiplicity 1, when all the other knots in Ξ are distinct to ζ_i. The first and last knots are called end knots which are the ends of the finite interval and the other knots are called interior knots. A knot vector is called open if both the end knots both have multiplicity $n+1$.

B-splines with degree n on the knots vector Ξ are given by the following recurrence relation:

$$\phi_{i,n}(x) = \frac{x - \zeta_i}{\zeta_{i+n} - \zeta_i}\phi_{i,n-1}(x) + \frac{\zeta_{i+n+1} - x}{\zeta_{i+n+1} - \zeta_{i+1}}\phi_{i+1,n-1}(x), \tag{1.5.2}$$

and for the case when $n = 0$,

$$\phi_{i,0}(x) = \begin{cases} 1, & x \in [\zeta_i, \zeta_{i+1}) \\ 0, & \text{otherwise} \end{cases}.$$

Note that in the recurrence relation (1.5.2), if $\zeta_{i+n} - \zeta_i = 0$, then the term $\frac{x-\zeta_i}{\zeta_{i+n}-\zeta_i} = 0$, and similarly if $\zeta_{i+n+1} - \zeta_{i+1} = 0$, then $\frac{\zeta_{i+n+1}-x}{\zeta_{i+n+1}-\zeta_{i+1}} = 0$. If a knot ζ_i has multiplicity k, a basis is C^{n-k} continuous at the knot. In particular, if a knot has multiplicity n, the basis is C^0 at the knot. Moreover, in this case, the basis is interpolatory at the knot, that is, the value of the B-spline at the knot is the same as the value of the coefficient corresponding to this knot. Making use of this fact, de Boor's method (see [69, 67] for more details) provides an efficient way to evaluate a B-spline at any point.

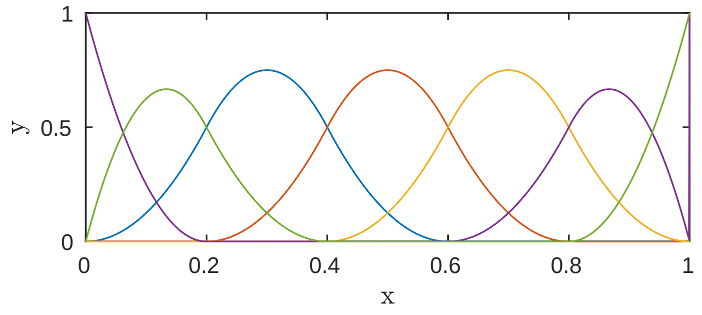


Figure 1.3: B-spline of degree 2 on the interval [0,1]

In this work, we only consider the B-splines with open end knots, and equispaced and non-repeated interior knots, that is, B-spline with $\Xi = (\underbrace{0,\ldots,0}_{n+1}, 1/N, 2/N, \ldots, (N-1)/N, \underbrace{1,\ldots,1}_{n+1})$. In this case, the interval $[0,1]$ is divided into N sub-intervals, and there are $N+2n+1$ elements in the knots vector and $N+n$ B-spline basis. We define the corresponding B-spline space by $X^{N,n}([0,1]) = \text{span}\{\phi_{0,n}, \ldots . \phi_{N+n-1,n}\}$. $X^{N,n}([0,1])$ consists of equally spaced piecewise polynomials with C^{n-1} smoothness. It follows that $X^{N,n}([0,1]) \subset H^n([0,1])$. Figure 1.3 gives an example of a basis with degree 2 on the interval [0,1].

To study the problems with Dirichlet boundary condition in the following chapters, we also need a subspace $X_0^{N,n}([0,1])$, which is composed of all the functions in $X^{N,n}([0,1])$ vanishing on the boundary of the domain. Obtained by imposing two extra constraints to $X^{N,n}([0,1])$, this space $X_0^{N,n}([0,1])$ has the dimension $N+n-2$. Also because the functions in $\{\phi_{i,n}\}_{i=1}^{N+n-2}$ vanish on the boundary and are linearly independent, they form a basis for $X_0^{N,n}([0,1])$. For simplicity, by denoting $\psi_{i,n} = \phi_{i+1,n}$, the basis is given by $\{\psi_{i,n}\}_{i=0}^{N+n-3}$.

1.5.4 Multivariate Tensor Product B-spline

Let $\mathcal{I} = \{0,\ldots,N-1\}$, $\mathcal{J} = \{0,\ldots,N+n-1\}$ and $\mathcal{J}_0 = \{0,\ldots,N+n-3\}$. We define a multi-variate B-spline on $\mathbb{R}^d$ as a tensor product of univariate B-splines in the sense that for variable $\boldsymbol{x} = (x_1,\ldots,x_d) \in \mathbb{R}^d$, and index $\boldsymbol{i} = (i_1,\ldots,i_d) \in \mathcal{I}^d$,

$\boldsymbol{j} = (j_1, \ldots, j_d) \in \mathcal{J}^d$, and $\boldsymbol{k} = (k_1, \ldots, k_d) \in \mathcal{J}_0^d$,

$$\begin{aligned} b_{\boldsymbol{i},n}(\boldsymbol{x}) &= b_{i_1,n}(x_1) b_{i_2,n}(x_2) \ldots b_{i_d,n}(x_d), \\ \phi_{\boldsymbol{j},n}(\boldsymbol{x}) &= \phi_{j_1;n}(x_1) \phi_{j_2,n}(x_2) \ldots \phi_{j_d,n}(x_d), \\ \text{and } \psi_{\boldsymbol{k},n}(\boldsymbol{x}) &= \psi_{k_1,n}(x_1) \psi_{k_2,n}(x_2) \ldots \psi_{k_d,n}(x_d). \end{aligned}$$

The tensor-product B-spline space is defined as

$$X^{N,n}(\mathbb{T}^d) = \text{span}\{b_{\boldsymbol{i},n}(\boldsymbol{x})\}_{\boldsymbol{i} \in \mathcal{I}^d}, \qquad X^{N,n}([0,1]^d) = \text{span}\{\phi_{\boldsymbol{i},n}(\boldsymbol{x})\}_{\boldsymbol{i} \in \mathcal{J}^d},$$
$$\text{and } X_0^{N,n}([0,1]^d) = \text{span}\{\psi_{\boldsymbol{i},n}(\boldsymbol{x})\}_{\boldsymbol{i} \in \mathcal{J}_0^d}.$$

The definition of $b_{\boldsymbol{i},n}(\boldsymbol{x})$ implies that it is periodic in the sense that $b_{\boldsymbol{i},n}(\boldsymbol{x}) = b_{\boldsymbol{i},n}(\boldsymbol{x} + \boldsymbol{e}^k)$, where $\boldsymbol{e}^k = (\underbrace{0, \ldots, 0, 1}_{k}, 0, \ldots, 0)$. The smoothness and periodicity of functions in $X^{N,n}(\mathbb{T}^d)$ implies that the space is a subspace of $\mathcal{H}^n(\mathbb{T}^d)$. We also have that $X^{N,n}([0,1]^d) \subset \mathcal{H}^n([0,1]^d)$ and $X_0^{N,n}([0,1]^d) \subset \mathcal{H}_0^n([0,1]^d)$.

Sometimes it's more convenient to identify a B-spline by using a scalar index instead of vector, as we do in the following chapters. Recall the bijective operator σ_N defined in (1.3.4). Letting $i = \sigma_N(\boldsymbol{i})$, $j = \sigma_{N+n}(\boldsymbol{j})$ and $k = \sigma_{N+n-2}(\boldsymbol{k})$, we define

$$b_{i,n}(\boldsymbol{x}) = b_{\boldsymbol{i},n}(\boldsymbol{x}), \quad \phi_{j,n}(\boldsymbol{x}) = \phi_{\boldsymbol{j},n}(\boldsymbol{x}), \text{ and } \psi_{k,n}(\boldsymbol{x}) = \psi_{\boldsymbol{k},n}(\boldsymbol{x}), \tag{1.5.3}$$

respectively.

Chapter 2

B-spline Interpolation

2.1 Background

Interpolation is the process of constructing a function that fits the data points from the underlying function. It is widely applied in an approximation of functions, numerical differentiation and numerical integrations.

A commonly used family of functions for interpolation is polynomials, which are widely studied in literature such as [68, 23]. When high degree polynomials are used with the data of equispaced interpolation points, the approximation may suffer from Runge's phenomenon, which is a problem of oscillation near the ends of the intervals. One option to avoid the problem is to use splines in stead of polynomials for the interpolation. Cardinal spline interpolation is studied by Schoenberg in [79]. Regarding its approximation property, the author obtains a Peano type remainder formula for the interpolation [80]. In [34], Goodman and Lee generalized the remainder to a class of symmetric cardinal interpolation problems on $\mathbb{R}$ and gave a L_∞ error estimation. The estimation depends on the degree of the spline, which means it is capable of showing the behaviour of the error as the degree increases.

We are interested in estimation with the same feature but measured in semi-norms on a finite interval. At first, we study interpolation with periodic B-splines and give an error estimate in the L_2 norm by using the Peano remainder and estimation in [34]. After showing a best approximation property, this result together with the Kolmogorov inequality in (2.3.10) leads to an estimate in semi-norm. Following from this, we derive an estimate for interpolation of tensor-product splines. Finally, we extend these results to interpolation of functions which are not necessarily periodic by introducing a new basis for the B-spline space.

2.2 Overview of Cardinal B-spline Interpolation

We recall some results about cardinal spline interpolation in [34]. For definition and properties of cardinal B-spline, the reader may see Section 1.5.1.

A function f is called of power growth if there exist a constant γ such that $f^{(n)}(x) = \mathcal{O}(|x|^\gamma)$ as $x \to \infty$. Assume the function $f \in C^n(\mathbb{R})$, whose nth derivative $f^{(n)}$ is absolutely continuous on $(j, j+1)$, $j \in \mathbb{Z}$, is of power growth.

Letting

$$z_n = \begin{cases} 0 & \text{when } n \text{ is odd} \\ 1/2 & \text{when } n \text{ is even} \end{cases},$$

there exists a unique cardinal spline $f_h \in X^n(\mathbb{R})$, such that $f_h(i + z_n) = f(i + z_n)$, for each $i \in \mathbb{Z}$. The interpolation is given by

$$f_h = \sum_{i=-\infty}^{+\infty} f(i + z_n) l_n(x - i),$$

where $l_n(x)$ is the unique Lagrange function in the cardinal B-spline space such that $l_n(i + z_n) = \delta_{i,0}$. We denote by S_n the operator such that $S_n(f; x) = f_h(x)$, or $S_n f = f_h$.

Theorem 2.1 in [34] gives a Peano type remainder formula for the interpolation,

$$f(x) - S_n(f; x) = \int_{-\infty}^{+\infty} K_n(x, t) f^{(n+1)}(t) \mathrm{d}t, \tag{2.2.1}$$

where $K_n(x, t)$ is the Peano kernel defined as

$$K_n(x, t) = \frac{1}{n!}((x - t)_+^n - S_n((\cdot - t)_+^n; x)),$$

with

$$(x-t)_+^n = \begin{cases} (x-t)^n & x \geq t \\ 0 & x < t \end{cases}.$$

The kernel $K_n(x,t)$ is bounded by Lemma 4.1 in [34] with

$$\left\| \int_{-\infty}^{+\infty} |K_n(\cdot,t)| dt \right\|_\infty \leq C/\pi^{n+1}, \tag{2.2.2}$$

where the constant C is independent of n and $\|\cdot\|_\infty$ is the essential supremum norm. The remainder formula implies an error estimate given in Theorem 1 in [34],

$$\|f - S_n f\|_\infty \leq C \|f^{(n+1)}\|_\infty,$$

where the constant C is independent of n and f.

2.3 Univariate Periodic B-spline Interpolation

We first investigate the problem for univariate functions, and then extend to multivariate functions.

2.3.1 Method of Interpolation

Let $\mathcal{I} = \{0, \dots, N-1\}$. We choose the points for interpolation to be $\{\xi_j\}_{j \in \mathcal{I}}$ with

$$\xi_j = \begin{cases} jh & \text{when } n \text{ is odd} \\ (j+0.5)h & \text{when } n \text{ is even} \end{cases}. \tag{2.3.1}$$

A B-spline interpolant of a function $f \in H^{n+1}(\mathbb{T})$ is obtained by finding a spline $f_h \in X^{N,n}(\mathbb{T})$ such that for each $j \in \mathcal{I}$,

$$f_h(\xi_j) = f(\xi_j). \tag{2.3.2}$$

The interpolation problem has a unique solution as a special case of cardinal spline interpolation on $\mathbb{R}$, which is proved in Corollary 4.3 of [35].

Express the interpolant in a B-spline basis as

$$S_{h,n}(f;x) = \sum_{j\in\mathcal{I}} \beta_j b_{j,n}(x).$$

To find the unknown coefficients β_j, we impose constraints on the points in (2.3.2) for each $j \in \mathcal{I}$. This gives a series of N equations, which may be rewritten as a linear system,

$$\boldsymbol{G}\vec{\beta} = \vec{f},$$

with entries for matrix $\boldsymbol{G} \in \mathbb{R}^{N\times N}$,

$$\boldsymbol{G}_{i,j} = b_{j,n}(\xi_i), \tag{2.3.3}$$

and for vector $\vec{f} \in \mathbb{R}^N$,

$$\vec{f_i} = f(\xi_i).$$

Let $\boldsymbol{T} = \boldsymbol{G}^{-1}$. It follows that

$$\vec{\beta} = \boldsymbol{T}\vec{f}.$$

Therefore the interpolant f_h is given by

$$f_h(x) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \boldsymbol{T}_{i,j} f(\xi_j) b_{i,n}(x),$$

where denote by $S_{h,n} : \mathcal{H}^{n+1}(\mathbb{T}) \to X^{N,n}(\mathbb{T})$ the interpolation operator s.t. $S_{h,n}(f;x) = f_h(x)$ or $S_{h,n}f = f_h$.

Property 4(ii) in Page 16 together with (2.3.3) implies that

$$\begin{aligned}\boldsymbol{G}_{i,j} &= b_{j-i,n}(\xi_0)\\ &= \boldsymbol{G}_{0,[j-i]_N},\end{aligned}$$

with $[k]_N = k \mod N$. Therefore $\boldsymbol{G}$ is the circulant matrix

$$\boldsymbol{G} = \mathrm{Cir}(b_{0,n}(\xi_0), b_{1,n}(\xi_0), \ldots, b_{N-1,n}(\xi_0)). \tag{2.3.4}$$

As introduced in Section 1.3.2, the circulant matrix $\boldsymbol{G}$ has a diagonalized form

$$\boldsymbol{G} = \boldsymbol{F}^{-1}\boldsymbol{\Lambda}_G\boldsymbol{F}. \tag{2.3.5}$$

$\mathbf{\Lambda}_G$ is a diagonal matrix, whose diagonal entries are eigenvalues of $\boldsymbol{G}$, and the diagonal of $\mathbf{\Lambda}_G$ can be computed by applying the FFT to the first of row of $\boldsymbol{G}$ as shown in section 1.3.2. The coefficients of the interpolant $\vec{\beta}$ are then

$$\vec{\beta} = \boldsymbol{F}^{-1}\mathbf{\Lambda}_G^{-1}\boldsymbol{F}\vec{f}.$$

Recall that $.*$ stands for component-wise product of two vectors as in (1.3.3). The interpolation is summarized in the following steps:

Algorithm 2.1. *1. Use the FFT to obtain $\hat{f} = \boldsymbol{F}\vec{f}$ and $\vec{v}_G = \boldsymbol{F}\vec{a}_G^T$. $\vec{a}_G$ is the first row of $\boldsymbol{G}$ computed using expression* (2.3.3) *and de Boor's algorithm.*

2. Calculate $\tilde{f} = \vec{v}_G. * \hat{f}$.

3. Apply the FFT again to get $\vec{\beta} = \boldsymbol{F}^{-1}\tilde{f}$.

2.3.2 Error of Interpolation

We use a similar approach as in [34] by first deriving the remainder formula and then estimating the error by analysing the formula. The formula is given as follows.

Lemma 2.2. *If $f \in H^{n+1}(\mathbb{T})$, then*

$$f(x) - S_{h,n}(f;x) = h^n \int_0^1 K_{h,n}(x,t) f^{(n+1)}(t)dt, \tag{2.3.6}$$

with

$$K_{h,n}(x,t) = \sum_{j\in\mathbb{Z}} K_n(x/h, (t-j)/h).$$

Proof. For any $f \in H^{n+1}(\mathbb{T})$, define $\tilde{f}(x) = f(x/h)$. Then $\tilde{f} \in C^n(\mathbb{R})$, whose nth derivative $\tilde{f}^{(n)}$ is absolutely continuous on $(j, j+1)$, $j \in \mathbb{Z}$, and $\tilde{f}(x) = \tilde{f}(x+N)$ on $\mathbb{R}$, which implies that it has power growth. Let $y = x/h$ and $\tau = t/h$. The equality (2.2.1) and periodicity of $\tilde{f}^{(n)}$ gives

$$\begin{aligned} f(x) - S_{h,n}(f;x) &= \tilde{f}(y) - S_n(\tilde{f};y) \\ &= \int_{-\infty}^{+\infty} K_n(y,\tau)\tilde{f}^{(n+1)}(\tau)\mathrm{d}\tau \\ &= \int_0^N \sum_{i\in\mathbb{Z}} K_n(y,\tau+Ni)\tilde{f}^{(n+1)}(\tau)\mathrm{d}\tau \\ &= h^n \int_0^1 \sum_{i\in\mathbb{Z}} K_n(x/h,(t+i)/h)f^{(n+1)}(t)\mathrm{d}t. \end{aligned}$$

□

To estimate the remainder in Lemma 2.2 , one would intuitively apply the Cauchy-Schwarz inequality to the right hand side of equality (2.3.6), leaving

$$f(x) - S_{h,n}(f;x) \le h^n \left(\int_0^1 K_{h,n}^2(x,t)\mathrm{d}t\right)^{1/2} \|f^{(n+1)}\|.$$

Taking the L_2 norm on both sides of the inequality, it follows that

$$\|f - S_{h,n}f\| \le h^n \left(\int_0^1\int_0^1 K_{h,n}^2(x,t)\mathrm{d}t\mathrm{d}x\right)^{1/2} \|f^{(n+1)}\|. \tag{2.3.7}$$

Applying the first mean value theorem gives

$$\begin{aligned} \int_0^1\int_0^1 |K_{h,n}^2(x,t)|\mathrm{d}t\mathrm{d}x &= \int_0^1 K_{h,n}(\xi,t)\int_0^1 |K_{h,n}(x,t)|\mathrm{d}x\mathrm{d}t \\ &= \int_0^1 |K_{h,n}(\xi,t)|\mathrm{d}t\int_0^1 |K_{h,n}(x,\eta)|\mathrm{d}x, \end{aligned} \tag{2.3.8}$$

where ξ and η are real numbers in $[0,1]$.

Now to estimate the right hand side terms in this equality, we first put forward a relation between them given in the following lemma, which can save us the trouble of estimating both of them.

Lemma 2.3. *The kernel $K_{h,n}(x,t)$ defined in Lemma 2.2 has the property*

$$K_{h,n}(x,t) = \begin{cases} K_{h,n}(t,x) & n \text{ is odd} \\ -K_{h,n}(t-0.5,x-0.5) & n \text{ is even} \end{cases}.$$

Proof. From the definition of the kernel $K_{h,n}(x,t)$ in (2.3.6), the result of the lemma follows naturally if the equality

$$K_n(x,t) = \begin{cases} K_n(t,x) & n \text{ is odd} \\ -K_n(t-0.5, x-0.5) & n \text{ is even} \end{cases} \tag{2.3.9}$$

is shown to be true. This is the object of the rest of the proof.

Equality (2.7) in [34] implies that

$$S_n((x-t)_+^n; x) = \begin{cases} S_n((x-\cdot)_+^n; t) & n \text{ is odd} \\ S_n((x-\cdot-0.5)_+^n; t-0.5) & n \text{ is even} \end{cases}.$$

This leads to

$$K_n(x,t) = \begin{cases} \frac{1}{n!}(x-t)_+^n - \frac{1}{n!}S_n((x-\cdot)_+^n; t) & n \text{ is odd} \\ \frac{1}{n!}(x-t)_+^n - \frac{1}{n!}S_n(x-(\cdot+0.5))_+^n; t-0.5)) & n \text{ is even} \end{cases}.$$

From the fact

$$(x-t)^n = (x-t)_+^n - (-1)^{n+1}(t-x)_+^n,$$

and $S_n((x-\cdot)^n; t) = (x-t)^n$, it follows that

$$K_n(x,t) = \begin{cases} \frac{1}{n!}(t-x)_+^n - \frac{1}{n!}S_n((\cdot-x)_+^n; t) & n \text{ is odd} \\ -\frac{1}{n!}(t-0.5-(x-0.5))_+^n - \frac{1}{n!}S_n((\cdot-(x-0.5))_+^n; t-0.5) & n \text{ is even} \end{cases}.$$

This shows equality (2.3.9) is true, and therefore completes the proof of the lemma □

An error bound of the interpolation is obtained as stated in the following theorem.

Theorem 2.4. *If $f \in H^{n+1}(\mathbb{T})$, then*

$$\|f - S_{h,n}f\| \le Ch^{n+1}/\pi^{n+1}\|f^{(n+1)}\|.$$

Proof. Together with (2.3.7), we estimate the kernel terms in (2.3.8) to give the result. To estimate the term $\int_0^1 |K_{h,n}(\xi,t)|\mathrm{d}t$, the triangle inequality and the error bound in (2.2.2) yield

$$\begin{aligned}\int_0^1 |K_{h,n}(\xi,t)|\mathrm{d}t &\le \sum_{i\in\mathbb{Z}} \int_0^1 |K_n(\xi/h,(t+i)/h)|\mathrm{d}t \\ &= h\int_{-\infty}^{+\infty} |K_n(\xi/h,t)|\mathrm{d}t \\ &\le h\|\int_{-\infty}^{+\infty} |K_n(\cdot,t)|\mathrm{d}t\|_\infty \\ &\le Ch/\pi^{n+1}.\end{aligned}$$

As for the term $\int_0^1 |K_{h,n}(x,\eta)|\mathrm{d}x$, using Lemma 2.3 and a similar approach to the previous analysis it follows

$$\int_0^1 |K_{h,n}(x,\eta)|\mathrm{d}x \le Ch/\pi^{n+1}.$$

Hence,

$$\left(\int_0^1\int_0^1 K_{h,n}^2(x,t)\mathrm{d}t\mathrm{d}x\right)^{1/2} \le Ch/\pi^{n+1}.$$

Substituting this estimate in (2.3.7) completes the proof. □

Making use of this result and the Kolmogorov inequality, we now study the error $\|(f-S_{h,n}f)^{(k)}\|$. The Kolmogorov inequality (see Proposition 3.3.7 [53]) states that if $g \in H^n(\mathbb{T})$ then

$$\|g^{(k)}\| \le \|g\|^{1-k/n}\|g^{(n)}\|^{k/n}. \tag{2.3.10}$$

Let $n = 2r+1$, $r \in \mathbb{N}_0$, the term $f - S_{h,n}f \in H^n(\mathbb{T})$, and $f \in H^{r+1}(\mathbb{T})$. Applying the Kolmogorov inequality we have

$$\|(f-S_{h,n}f)^{(k)}\| \le \|f-S_{h,n}f\|^{1-k/(r+1)}\|(f-S_{h,n}f)^{(r+1)}\|^{k/(r+1)}. \tag{2.3.11}$$

We used the Kolmogorov inequality corresponding to functions in $H^{r+1}(\mathbb{T})$ rather than $H^n(\mathbb{T})$, because we are able to give an error bound for the term $\|(f-S_{h,n}f)^{(r+1)}\|$, where the term is shown to be smaller than $\|f^{(r+1)} - S_{h,n}f^{(r+1)}\|$ based on the following lemma and then estimated using Theorem 2.4.

Denote by $I^n_{m,h}$ the operator such that $I^n_{m,h}(f;\cdot) = g \in X^{N,n}(\mathbb{T})$ satisfying

$$(g, v) = (f, v) \text{ for any } v \in X^{N,m}(\mathbb{T}). \tag{2.3.12}$$

Lemma 2.5. *Let $n = 2r + 1$, $r \in \mathbb{N}_0$. We have*

$$(S_{h,n}f)^{(k)} = I^{n-k}_{k-1,h} f^{(k)} \text{ for } k = 0, \dots, n.$$

Proof. We start with expressing the right hand side of the equality by B-splines. Let $I^{n-k}_{k-1,h} f = \sum\limits_{j=0}^{N-1} c_j b_{j,n-k}$ and choose v in (2.3.12) to be $b_{i,k-1}$, for each $i \in \mathcal{I}$. The process leaves a series of equations that are able to be written in the form of the linear system

$$\boldsymbol{M}^{k-1,n-k}\vec{c} = \vec{F},$$

with $\boldsymbol{M}^{k-1,n-k}_{i,j} = (b_{i,k-1}, b_{j,n-k})$, $\vec{c} = (c_0, c_1, \dots, c_{N-1})^T$ and $\vec{F}_i = (b_{i,k-1}, f^{(k)})$. Let $\vec{b}_n = (b_{0,n}, b_{1,n}, \dots, b_{N-1,n})$. Then we have that

$$I^{n-k}_{k-1,h} f = \vec{b}_{n-k}(\boldsymbol{M}^{k-1,n-k})^{-1}\vec{F}. \tag{2.3.13}$$

Using Property 4(iv) in Page 16, we have

$$\begin{aligned} \boldsymbol{M}^{k-1,n-k}_{i,j} &= \int_0^1 b_{i,k-1}(x) b_{j,n-k}(x) dx \\ &= h b_{j-i+k,n}(0). \end{aligned}$$

This fact and the periodic property of B-splines shows that

$$\boldsymbol{M}^{k-1,n-k} = h\mathrm{Cir}\{(b_{k,n}(0), b_{1+k,n}(0), \dots, b_{N-1+k,n}(0))\}.$$

As for the left hand side, assuming $S_{h,n}f = \sum\limits_{i=0}^{N-1} d_i b_{i,n}$, the coefficients d_i are obtained by solving the linear system

$$\boldsymbol{G}\vec{d} = \vec{f},$$

with $\vec{f}_i = f(ih)$, $\vec{d}_i = d_i$ and

$$\begin{aligned} \boldsymbol{G}_{i,j} &= b_{j,n}(ih) \\ &= b_{j-i,n}(0). \end{aligned}$$

This implies

$$\boldsymbol{G} = \text{Cir}(b_{0,n}(0), b_{1,n}(0), \ldots, b_{N-1,n}(0)).$$

Note also that $S_{h,n}f = \vec{b}_n \boldsymbol{G}^{-1}\vec{f}$.

If we compare $\boldsymbol{M}^{k-1,n-k}$ and $\boldsymbol{G}$, and consider the periodicity of B-splines, it's apparent that $\boldsymbol{G}$ is obtained by shifting the matrix $1/h\boldsymbol{M}^{k-1,n-k}$ down by k rows. To express the relation using matrix products, let $\boldsymbol{D} = \text{Cir}\{(0, \ldots, 0, 1)\}$, then

$$\boldsymbol{G} = 1/h\boldsymbol{D}^k\boldsymbol{M}^{k-1,n-k}.$$

Inverting the matrices on both side of the equation gives,

$$\boldsymbol{G}^{-1}\boldsymbol{D}^k = h(\boldsymbol{M}^{k-1,n-k})^{-1}. \tag{2.3.14}$$

To relate $\vec{F}$ and $\vec{f}$, we first define the matrix $\boldsymbol{D}_F = \text{Cir}(-1, 1, 0, \ldots, 0)$ whose product with a vector gives a forward difference. Property 4(v) in Page 16 implies

$$\begin{aligned} \vec{F}_i &= \textstyle\int_0^1 b_{i,k-1}(x) f^{(k)}(x)dx \\ &= 1/h^{k-1}(D_h^+)^k f(ih). \end{aligned}$$

It follows that

$$\vec{F} = 1/h^{k-1}\boldsymbol{D}_F^k\vec{f}. \tag{2.3.15}$$

Let $\boldsymbol{D}_B := \text{Cir}(1, 0, \ldots, 0, -1)$. A simple matrix calculation gives $\boldsymbol{D}\boldsymbol{D}_F = \boldsymbol{D}_B$. Since the multiplication of circulant matrices is commutative [93], taking the kth derivative of $S_{h,n}(f;x)$ and using Property 4(vi) in Page 16, we have

$$\begin{aligned} (S_{h,n}f)^{(k)} &= 1/h^k\vec{b}_{n-k}\boldsymbol{D}_B^k\boldsymbol{G}^{-1}\vec{f} \\ &= 1/h^k\vec{b}_{n-k}\boldsymbol{G}^{-1}\boldsymbol{D}^k\boldsymbol{D}_F^k\vec{f}. \end{aligned}$$

This equality together with equality (2.3.14), (2.3.15), and (2.3.13) gives the result of the lemma.

□

The definition of the operator $I_{m,h}^n$ implies that $I_{r,h}^r f^{(r+1)} - f^{(r+1)}$ is orthogonal to every element in $X^{N,r}(\mathbb{T})$. Therefore together with Lemma 2.5, we know

$(S_{h,n}f)^{(r+1)} = I^r_{r,h}f^{(r+1)}$ is the best approximation of $f^{(r+1)}$ from $X^{N,r}(\mathbb{T})$, in the sense that

$$\|f^{(r+1)} - (S_{h,n}f)^{(r+1)}\| \leq \|f^{(r+1)} - v\| \text{ for any } v \in X^{N,r}(\mathbb{T}).$$

This inequality implies that

$$\begin{aligned} \|(f - S_{h,n}f)^{(r+1)}\| &\leq \|f^{(r+1)} - S_{h,r}f^{(r+1)}\| \\ &= Ch^{r+1}/\pi^{r+1}\|f^{(n+1)}\|. \end{aligned} \tag{2.3.16}$$

The result enables us to give the following theorem.

Theorem 2.6. *Let $n = 2r+1$, $r \in \mathbb{N}_0$ and $f \in H^{n+1}(\mathbb{T})$. For $k = 0, \ldots, r+1$, we have*

$$\|(f - S_{h,n}f)^{(k)}\| \leq C(h/\pi)^{n+1-k}\|f^{(n+1)}\|.$$

Proof. Substituting Theorem 2.4 and the inequality (2.3.16) into (2.3.11) implies

$$\begin{aligned} \|(f - S_{h,n}f)^{(k)}\| &\leq \|f - S_{h,n}f\|^{1-k/(r+1)}\|(f - S_{h,n}f)^{(r+1)}\|^{k/(r+1)} \\ &\leq (Ch^{n+1}/\pi^{n+1}\|f^{(n+1)}\|)^{1-k/(r+1)}(Ch^{r+1}/\pi^{r+1}\|f^{(n+1)}\|)^{k/(r+1)} \\ &= C(h/\pi)^{n+1-k}\|f^{(n+1)}\|. \end{aligned}$$

□

The theorem shows that as the mesh size $h \to 0$, the error has an algebraic rate of convergence [82, p.78], which agrees with the result for the h-refinements [8, Lemma 3.3]. However, due to the term $\|f^{(n+1)}\|$ on the right-hand side of the theorem, we cannot conclude that the error converges as the degree $n \to \infty$. As shown in the numerical results in Section 2.7.0.1, there exist functions such that the error diverges. Therefore, we make more assumptions to facilitate convergence, which is shown in the following corollary.

Corollary 2.7. *For $n = 2r+1$, $r \in \mathbb{N}_0$, let $f \in H^{n+1}(\mathbb{T})$. Assume there exists $\sigma > 0$ s.t. $\|f^{(n+1)}\| \leq \sigma^{n+1}\|f\|$. For $k = 0, \ldots, r+1$, we have*

$$\|(f - S_{h,n}f)^{(k)}\| \leq C(h\sigma/\pi)^{n+1-k}\|f\|.$$

The corollary implies that, for a function $f \in C^\infty(\mathbb{T})$ satisfying Markov type inequality $\|f^{(k)}\| \leq \sigma^k \|f\|$, $k \in \mathbb{N}_0$, the error converges exponentially with the degree n, if we choose h such that $h\sigma/\pi < 1$.

The following corollary gives an estimate for the more general case where the function f is of any period l, which is needed in Section 2.5.2. Let $H^{n+1}(l\mathbb{T})$ be the Sobolev space on $l\mathbb{T}$, which is the circle of length l, and $X^{N,n}(l\mathbb{T})$ the periodic spline space with degree n and N sub-intervals of in a period. Denote by $L_2(l\mathbb{T})$ the L_2 space defined on $l\mathbb{T}$.

Corollary 2.8. *Let $f \in H^{n+1}(l\mathbb{T})$, $n = 2r+1$, $r \in \mathbb{N}_0$, and $f_h \in X^{N,n}(l\mathbb{T})$ be the interpolant such that $f(l\xi_i) = f_h(l\xi_i)$. Then*

$$\|(f - f_h)^{(k)}\|_{L_2(l\mathbb{T})} \leq C(l/(N\pi))^{n+1-k} \|f^{(n+1)}\|_{L_2(l\mathbb{T})}.$$

Proof. Let $x = ly$, $g(y) = f(ly)$ and $g_h(y) = f_h(y)$. In this case, we have $g \in \mathcal{H}^{n+1}(\mathbb{T})$, and then g_h is the corresponding B-spline interpolant. Applying Theorem 2.6, we complete the proof. □

Remark 2.9. *In some literature, Bernoulli kernel is used to represent and estimate the error of periodic spline interpolation. For instance [53, Ch. 5] shows that for $f \in W_\infty^{n+1}(\mathbb{T}), k = 0, 1,$*

$$\|(f - S_{h,n}f)^{(k)}\| \leq C(h/\pi)^{n+1-k} \|f^{(n+1)}\|_\infty,$$

where $W_\infty^{n+1}(\mathbb{T})$ is the periodic Sobolev space such that for every $f \in W_\infty^{n+1}(\mathbb{T})$, $f^{(n)}$ is absolute continuous and $\|f^{(n+1)}\|_\infty < \infty$.

The same estimates was also derived by studying the extremal property of the functions from $W_\infty^{n+1}(\mathbb{T})$ that vanishes at the breaking points. The idea is to bound the error with perfect splines (Theorem 5.1.1 [53]) and then estimating the norm of the error by evaluating the norm of the perfect spline.

In our case, we generalise the assumption of the interpolated function from being in the space $W_\infty^{n+1}(\mathbb{T})$ to $H^{n+1}(\mathbb{T})$.

Remark 2.10. *Regarding to estimating the interpolation error in semi-norm, the existing approach include making use of the estimation, for $n = 2r+1$,*

$$\|f - S_{h,n}f\| \leq C(h/\pi)^{r+1}\|f^{(r+1)}\|,$$

and the minimal semi-norm property of periodic spline interpolation as shown in [41], that is, for $n = 2r+1$,

$$\|(f - S_{h,n}f)^{(r+1)}\|^2 = \|f^{(r+1)}\|^2 - \|(S_{h,n}f)^{(r+1)}\|^2,$$

which implies that

$$\|(f - S_{h,n}f)^{(r+1)}\|^2 \leq \|f^{(r+1)}\|^2.$$

Then applying the Kolmogorov inequity, it follows

$$\begin{aligned} \|(f - S_{h,n}f)^{(k)}\| &\leq \|f - S_{h,n}f\|^{1-k/(r+1)}\|(f - S_{h,n}f)^{(r+1)}\|^{k/(r+1)} \\ &\leq C(h/\pi)^{r+1-k}\|f^{(r+1)}\|. \end{aligned} \tag{2.3.17}$$

Remark 2.11. *The best approximation property* (2.3.2) *was also proved with reproducing kernel approach as shown in [26].*

Remark 2.12. *B-spline functions on uniform meshes can also be viewed as radial basis functions (RBF). Literature discussing estimates of RBF interpolations, for example, [15], [58], and [14], also addresses estimates of spline interpolation problems.*

2.4 Multivariate Periodic B-spline Interpolation

2.4.1 Method of Interpolation

Recall that $\mathcal{I} = \{0, \ldots, N-1\}$. For interpolating a multivariate function $f(\boldsymbol{x}) \in \mathcal{H}^{n+1}(\mathbb{T}^d)$, we choose

$$\boldsymbol{\xi}_{\boldsymbol{i}} = (\xi_{i_1}, \xi_{i_2}, \ldots, \xi_{i_d}), \tag{2.4.1}$$

for each $\boldsymbol{i} = (i_1, i_2, \dots, i_d) \in \mathcal{I}^d$, where $\{\xi_k\}_{k\in\mathcal{I}}$, are the points defined in (2.3.1). Denote by $f_h(\boldsymbol{x}) \in X^{N,n}(\mathbb{T}^d)$ the B-spline interpolant of $f(\boldsymbol{x})$, and let

$$f_h(\boldsymbol{x}) = \sum_{\boldsymbol{j}\in\mathcal{I}^d} \beta_{\boldsymbol{j}} b_{\boldsymbol{j},n}(\boldsymbol{x}),$$

such that

$$f_h(\boldsymbol{\xi_i}) = f(\boldsymbol{\xi_i})$$

for each $\boldsymbol{i} \in \mathcal{I}^d$. Let $i = \sigma_N(\boldsymbol{i})$ and $j = \sigma_N(\boldsymbol{j})$, where σ_N is defined in (1.3.4). It follows that

$$\boldsymbol{\mathcal{G}}\vec{\beta} = \vec{f},$$

where entries for the vector $\vec{f}$ are

$$\vec{f_i} = f(\boldsymbol{\xi_i}),$$

and for the matrix $\boldsymbol{\mathcal{G}}$ are

$$\boldsymbol{\mathcal{G}}_{i,j} = b_{\boldsymbol{j},n}(\boldsymbol{\xi_i}). \tag{2.4.2}$$

Let $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{G}}^{-1}$. It follows that

$$\vec{\beta} = \boldsymbol{\mathcal{T}}\vec{f},$$

and therefore

$$f_h(\boldsymbol{x}) = \sum_{i=0}^{N^d-1}\sum_{j=0}^{N^d-1} \boldsymbol{\mathcal{T}}_{i,j} f(\boldsymbol{\xi_j}) b_{\boldsymbol{i},n}(\boldsymbol{x}). \tag{2.4.3}$$

The definition of $b_{\boldsymbol{j},n}$, (2.3.3), and (2.4.2) give

$$\begin{aligned} \boldsymbol{\mathcal{G}}_{i,j} &= b_{j_1,n}(\xi_{i_1}) b_{j_2,n}(\xi_{i_2}) \dots b_{j_d,n}(\xi_{i_d}) \\ &= \boldsymbol{G}_{i_1,j_1} \boldsymbol{G}_{i_2,j_2} \dots \boldsymbol{G}_{i_d,j_d}. \end{aligned}$$

Together with Property 2(i) in Page 8, it follows that

$$\boldsymbol{\mathcal{G}} = \underbrace{\boldsymbol{G} \otimes \boldsymbol{G} \otimes \dots \otimes \boldsymbol{G}}_{d}. \tag{2.4.4}$$

It follows that

$$\boldsymbol{\mathcal{T}} = \underbrace{\boldsymbol{T} \otimes \boldsymbol{T} \otimes \dots \otimes \boldsymbol{T}}_{d}.$$

This property together with (2.4.3), and Property 2(iv) in Page 8 yields

$$f_h(\boldsymbol{x}) = \sum_{i_1=0}^{N-1} \dots \sum_{i_d=0}^{N-1} \sum_{j_1=0}^{N-1} \dots \sum_{j_d=0}^{N-1} \boldsymbol{T}_{i_1,j_1} \dots \boldsymbol{T}_{i_d,j_d} f(\boldsymbol{\xi_j}) b_{\boldsymbol{i},n}(\boldsymbol{x}). \tag{2.4.5}$$

By abuse of notation, we still denote with $S_{h,n}(f;\boldsymbol{x}) = f_h(\boldsymbol{x})$ or $S_{h,n}f = f_h$ the interpolation operator regardless of the dimension of the variables $\boldsymbol{x}$ as it is apparent from the context in the thesis.

Substituting (2.3.5) into the expression (2.4.4), Property 2(iv) implies

$$\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{F}}^{-1} \boldsymbol{\Lambda}_{\mathcal{G}} \boldsymbol{\mathcal{F}},$$

with

$$\boldsymbol{\mathcal{F}} = \underbrace{\boldsymbol{F} \otimes \boldsymbol{F} \otimes \dots \otimes \boldsymbol{F}}_{d},$$

and

$$\boldsymbol{\Lambda}_{\mathcal{G}} = \underbrace{\boldsymbol{\Lambda}_G \otimes \boldsymbol{\Lambda}_G \otimes \dots \otimes \boldsymbol{\Lambda}_G}_{d}.$$

The coefficients for the interpolant f_h are

$$\vec{\beta} = \boldsymbol{\mathcal{F}}^{-1} \boldsymbol{\Lambda}_{\mathcal{G}}^{-1} \boldsymbol{\mathcal{F}} \vec{f}.$$

The matrix $\boldsymbol{\mathcal{G}}$ is invertible since it is the Kronecker product of invertible matrices. Therefore the high-dimensional interpolation problem has a unique solution.

The interpolation process is summarized as follows.

Algorithm 2.13.

1. *Use FFT to obtain $\hat{f} = \boldsymbol{\mathcal{F}} \vec{f}$ and $\vec{v}_{\mathcal{G}} = \vec{v}_G \otimes \dots \otimes \vec{v}_G$, where $\vec{v}_G$ is obtained using step 1 in Algorithm 2.1.*

2. *Calculate $\tilde{f} = \vec{v}_{\mathcal{G}} .* \hat{f}$.*

3. *Apply FFT again to get $\vec{\beta} = \boldsymbol{\mathcal{F}}^{-1} \tilde{f}$.*

2.4.2 Error of Interpolation

For $f \in \mathcal{H}^{n+1}(\mathbb{T}^d)$, we denote by the operator $S_{\boldsymbol{x}}$ and S_{x_s}

$$S_{\boldsymbol{x}} f(\boldsymbol{x}) = S_{h,n}(f;\boldsymbol{x}) \text{ and } S_{x_s} f(\boldsymbol{x}) = S_{h,n}(f;x_s),$$

which means $S_{\boldsymbol{x}} f(\boldsymbol{x})$ is given by (2.4.5) and $S_{x_s} f(\boldsymbol{x})$ has the expression

$$S_{x_s} f(\boldsymbol{x}) = \sum_{i_s=0}^{N-1} \sum_{j_s=0}^{N-1} \boldsymbol{T}_{i_s,j_s} f(x_1,\ldots,x_{s-1},\xi_{j_s},x_{s+1},\ldots,x_d) b_{i_s,n}(x_s).$$

It follows from the linearity of the summation that

$$S_{\boldsymbol{x}} f = S_{x_d} S_{x_{d-1}} \ldots S_{x_1} f.$$

Let I be the identity operator. For a vector $\vec{\alpha} = (\alpha_i)_{i=1}^N$, denote by $|\vec{\alpha}| = \sum\limits_{i=1}^N |\alpha_i|$. Recall that $\vec{\alpha} \le k$ means $\|\vec{\alpha}\|_\infty \le k$, for any $k \in \mathbb{N}_0$. Based on the results in the one-dimensional case, we have the following estimate for the interpolation.

Theorem 2.14. *Letting $f \in \mathcal{H}^{n+1}(\mathbb{T}^d)$ with $n = 2r+1$, $r \in \mathbb{N}_0$, and $\vec{k} \in \mathbb{N}_0^d$ with $\vec{k} \le r+1$, we have*

$$\|D^{\vec{k}}(f - S_{h,n} f)\| \le C \sum_{\vec{\alpha} \in \mathbb{N}_0^d, \|\vec{\alpha}\|_\infty = 1} \left(\frac{h}{\pi}\right)^{|\vec{\alpha}|(n+1) - \vec{\alpha}\vec{k}} \|D^{(n+1)\vec{\alpha} - \vec{\alpha}.*\vec{k} + \vec{k}} f\|.$$

Proof. Recall that

$$S_{\boldsymbol{x}} = S_{x_d} S_{x_{d-1}} \ldots S_{x_1}.$$

It follows that, for $\vec{\alpha} = (\alpha_1, \ldots, \alpha_d)$,

$$\begin{aligned} D^{\vec{k}}(I - S_{\boldsymbol{x}}) &= -D^{\vec{k}} \sum_{\vec{\alpha} \in \mathbb{N}_0^d, \|\vec{\alpha}\|_\infty = 1} (-I + S_{x_d})^{\alpha_d} \ldots (-I + S_{x_1})^{\alpha_1} \\ &= - \sum_{\vec{\alpha} \in \mathbb{N}_0^d, \|\vec{\alpha}\|_\infty = 1} \tfrac{\partial^{k_d}}{\partial x^{k_d}} (-I + S_{x_d})^{\alpha_d} \cdots \tfrac{\partial^{k_1}}{\partial x^{k_1}} (-I + S_{x_1})^{\alpha_1}. \end{aligned}$$

The Cauchy-Schwarz inequality and Theorem 2.6 imply

$$\begin{aligned}\|D^{\vec{k}}(I-S_{\boldsymbol{x}})f\| &\leq \sum_{\vec{\alpha}\in\mathbb{N}_0^d,\|\vec{\alpha}\|_\infty=1} \|\tfrac{\partial^{k_d}}{\partial x^{k_d}}(I-S_{x_d})^{\alpha_d}\dots\tfrac{\partial^{k_1}}{\partial x^{k_1}}(I-S_{x_1})^{\alpha_1}f\| \\ &\leq \sum_{\vec{\alpha}\in\mathbb{N}_0^d,\|\vec{\alpha}\|_\infty=1} (C(\tfrac{h}{\pi})^{\alpha_d(n+1-k_d)} \\ &\|\tfrac{\partial^{\alpha_d(n+1-k_d)+k_d}}{\partial x^{\alpha_d(n+1-k_d)+k_d}}\tfrac{\partial^{k_{d-1}}}{\partial x^{k_{d-1}}}(I-S_{x_{d-1}})^{\alpha_{d-1}}\dots\tfrac{\partial^{k_1}}{\partial x^{k_1}}(I-S_{x_1})^{\alpha_1}f\|) \\ &= \sum_{\vec{\alpha}\in\mathbb{N}_0^d,\|\vec{\alpha}\|_\infty=1} (C(\tfrac{h}{\pi})^{\alpha_d(n+1-k_d)} \\ &\|\tfrac{\partial^{k_{d-1}}}{\partial x^{k_{d-1}}}(I-S_{x_{d-1}})^{\alpha_{d-1}}\dots\tfrac{\partial^{k_1}}{\partial x^{k_1}}(I-S_{x_1}\ )^{\alpha_1}\tfrac{\partial^{\alpha_d(n+1-k_d)+k_d}}{\partial x^{\alpha_d(n+1-k_d)+k_d}}f\|).\end{aligned}$$

Repeatedly applying Theorem 2.6 to the remaining operators in a similar way gives

$$\begin{aligned}\|D^{\vec{k}}(I-S_{\boldsymbol{x}})f\| &\leq \sum_{\vec{\alpha}\in\mathbb{N}_0^d,\|\vec{\alpha}\|_\infty=1} (C(\tfrac{h}{\pi})^{\alpha_d(n+1-k_d)}\dots(\tfrac{h}{\pi})^{\alpha_1(n+1-k_1)} \\ &\|\tfrac{\partial^{\alpha_d(n+1-k_d)+k_d}}{\partial x^{\alpha_d(n+1-k_d)+k_d}}\dots\tfrac{\partial^{\alpha_1(n+1-k_1)+k_1}}{\partial x^{\alpha_1(n+1-k_1)+k_1}}f\|) \\ &= C\sum_{\vec{\alpha}\in\mathbb{N}_0^d,\|\vec{\alpha}\|_\infty=1} (\tfrac{h}{\pi})^{|\vec{\alpha}|(n+1)-\vec{\alpha}\vec{k}}\|D^{(n+1)\vec{\alpha}-\vec{\alpha}.*\vec{k}+\vec{k}}f\|.\end{aligned}$$

□

The following corollary generalizes Corollary 2.7 to the multi-dimensional case, where we also see an exponential convergence as p increases.

Corollary 2.15. *Let $f\in\mathcal{H}^{p+1}(\mathbb{T}^d)$ with $p=2r+1$, $r\in\mathbb{N}_0$. Assume there exist $\sigma>0$ s.t. $\|D^{\vec{s}}f\|\leq C\sigma^{|\vec{s}|}$, for all $\vec{s}\in\mathbb{N}_0^d$, $\vec{s}\leq p+1$. For any $\vec{k}\in\mathbb{N}_0^d$ with $\vec{k}\leq r+1$ and $h\sigma/\pi<1$, we have*

$$\|D^{\vec{k}}(f-S_{h,p}f)\|\leq C\left(\frac{h\sigma}{\pi}\right)^{p+1-\|\vec{k}\|_\infty}.$$

In particular, if $\|D^{\vec{s}}f\|\leq\sigma^{|\vec{s}|}\|f\|$, we have

$$\|D^{\vec{k}}(f-S_{h,p}f)\|\leq C\left(\frac{h\sigma}{\pi}\right)^{p+1-\|\vec{k}\|_\infty}\|f\|.$$

Proof. If $\|\vec{\alpha}\|_\infty=1$, then $|\alpha|\geq 1$ and $\vec{\alpha}\vec{k}\geq\vec{k}$. Since $h\sigma/\pi\leq 1$, we have that

$$\begin{aligned}\|D^{\vec{k}}(f-S_{h,p}f)\| &\leq C\sum_{\vec{\alpha}\in\mathbb{N}_0^d,\|\vec{\alpha}\|_\infty=1}\left(\tfrac{h\sigma}{\pi}\right)^{|\vec{\alpha}|(p+1)-\vec{\alpha}\vec{k}} \\ &\leq C\left(\tfrac{h\sigma}{\pi}\right)^{(p+1)-\|\vec{k}\|_\infty}.\end{aligned}$$

The particular case is proved similarly. □

In the following chapters we would frequently use the case when $\vec{k} = (0, \dots, 0)$ and $\vec{k} = (1, \dots, 1)$ in Theorem 2.14. For the simplicity, we define the operators

$$\begin{aligned} \mathcal{Q}_{0,n}(f) &= \sum_{\vec{\alpha}\in\mathbb{N}_0^d, \|\vec{\alpha}\|_\infty = 1} \left(\tfrac{h}{\pi}\right)^{|\vec{\alpha}|(n+1)} \|D^{(n+1)\vec{\alpha}} f\| \\ \text{and } \mathcal{Q}_{1,n}(f) &= \sum_{\vec{\alpha}\in\mathbb{N}_0^d, \|\vec{\alpha}\|_\infty = 1} \left(\tfrac{h}{\pi}\right)^{|\vec{\alpha}|n} \|D^{n\vec{\alpha}} \nabla f\|. \end{aligned} \tag{2.4.6}$$

As a special case of the theorem, for $n = 2r + 1, r \in \mathbb{N}$, we have

$$\begin{aligned} \|f - S_{h,n} f\| &\le C\mathcal{Q}_{0,n}(f) \\ \text{and } |f - S_{h,n} f|_1 &\le C\mathcal{Q}_{1,n}(f). \end{aligned} \tag{2.4.7}$$

If we further assume that there exist $\sigma > 0$ s.t. $\|D^{\vec{k}} f\| \le C\sigma^{|\vec{k}|}$ for all $\vec{k} \in \mathbb{N}_0^d$, $\vec{k} \le n + 1$. Letting $h\sigma/\pi \le 1$, it follows that

$$\begin{aligned} \mathcal{Q}_{0,n}(f) &\le C \sum_{\vec{\alpha}\in\mathbb{N}_0^d, \|\vec{\alpha}\|_\infty = 1} \left(\tfrac{h\sigma}{\pi}\right)^{|\vec{\alpha}|(n+1)} \le C\left(\tfrac{h\sigma}{\pi}\right)^{n+1} \\ \text{and } \mathcal{Q}_{1,n}(f) &\le Cd\sigma \sum_{\vec{\alpha}\in\mathbb{N}_0^d, \|\vec{\alpha}\|_\infty = 1} \left(\tfrac{h\sigma}{\pi}\right)^{|\vec{\alpha}|n} \le C\left(\tfrac{h\sigma}{\pi}\right)^{n}. \end{aligned} \tag{2.4.8}$$

2.5 Non-periodic B-spline Interpolation

2.5.1 Method of Interpolation

For $n = 2r + 1$, $r \in \mathbb{N}_0$, given any $f \in H^{n+1}([0, 1])$, we consider the interpolation such that interpolant $f_h \in X^{N,n}([0, 1])$ satisfies, for each $i = 0, \dots, N$,

$$f_h(\xi_i) = f(\xi_i),$$

and for each $k = 1, \dots, r$, and $j = 0$ or N,

$$f_h^{(2k)}(\xi_j) = f^{(2k)}(\xi_j). \tag{2.5.1}$$

Denote by $\mathcal{B}_{h,n} : H^{n+1}([0, 1]) \to X^{N,n}([0, 1])$ the operator such that $\mathcal{B}_{h,n} f = f_h$. According to Theorem 3.2 in [35], this problem has a unique solution.

2.5.2 Error of Interpolation

Before giving an error estimate, we define a basis for the space $X^{N,n}([0,1])$. Let $X_E^{N,n}([0,1])$ be a set composed of functions $f \in X^{N,n}([0,1])$ such that

$$f^{(2k)}(\xi_j) = 0, \text{ for } k = 0, \ldots, r, \text{ and for } j = 0, N.$$

The linearity of differentiation implies that $X_E^{N,n}([0,1])$ is a subspace of $X^{N,n}([0,1])$. The dimension of the space is $N-1$, since there are $n+1 = 2(r+1)$ more constraints to the space $X^{N,n}([0,1])$, whose dimension is $N+n$.

Let $\{\phi_i\}, i = 1, \ldots, N-1$, be a basis for the space $X_E^{N,n}[0,1]$ and the polynomial $\psi_i(x) = x^i, i = 0, \ldots, n$. We have the following lemma.

Lemma 2.16. *When $n = 2r+1$, $r \in \mathbb{N}_0$, the set*

$$\{\phi_i\}_{i=1,\ldots,N-1} \cup \{\psi_j\}_{j=0,\ldots,n}$$

forms a basis for space $X^{N,n}[0,1]$.

Proof. To prove that they are linearly independent, assume a linear combination of the basis elements satisfies

$$\sum_{i=1}^{N-1} c_i \phi_i(x) + \sum_{j=0}^{n} d_j \psi_j(x) = 0.$$

Take the $2r$th derivative on both sides of the equation and choose $x = \xi_j$, $j = 0$ or N. Note that $\xi_0 = 0$ and $\xi_N = 1$. Since $\phi_i^{(2r)}(\xi_0) = \phi_i^{(2r)}(\xi_N) = 0$, for $i = 1, \ldots, N-1$, and $\psi_j(x) = x^j$, we are left with two equations, $(n-1)!d_{n-1} = 0$ and $(n-1)!d_{n-1} + n!d_p = 0$ corresponding to the points ξ_0 and ξ_N respectively. It follows from the two equations that $d_{n-1} = d_p = 0$. Using a similar procedure by taking the $2k$th derivative of the equation, we may choose $x = \xi_j$, $j = 0$ or N, for $k = r-1, r-2, \ldots, 0$ in succession. Finally, we have $d_0 = d_1 = \ldots = d_p = 0$. Then it follows from linear independence of $\{\phi_i\}_{i=1,\ldots,N-1}$ that $c_i = 0$ for $i = 1, \ldots, N-1$. Therefore the functions are linearly independent. There are totally $N+n$ linear independent functions and the dimension of the space $X^{N,n}[0,1]$ is $N+n$. Hence, the functions form a basis for the space. $\square$

Since a function in $H^{n+1}(\mathbb{T})$ is of period 1, we only need to consider the function on a interval of a period, say, $[0,1]$. For simplicity, we define the space $H^{n+1}_{per}[0,1]$ consisting of the functions in $H^{n+1}(\mathbb{T})$, where only the part of the function on the domain $[0,1]$ is considered. Similarly, we define $H^{n+1}_{per}[-1,1]$ and $X^{2N,n}_{per}([-1,1])$ based on $H^{n+1}(2\mathbb{T})$ and $X^{2N,n}_{per}(2\mathbb{T})$, respectively. The error of the interpolation is bounded in the following theorem.

Theorem 2.17. *For $n = 2r+1$, $r \in \mathbb{N}_0$ and $f \in H^{n+1}[0,1]$,*

$$\|(\mathcal{B}_{h,n}f - f)^{(k)}\| \leq C \left(\frac{h}{\pi}\right)^{n+1-k} \|f^{(n+1)}\|, \text{ for } k = 0, \dots, r+1. \tag{2.5.2}$$

Proof. The idea of the proof is to express the error of non-periodic interpolation by that of periodic interpolation as shown in (2.5.6), and then using our estimates for periodic interpolation to arrive at the result as shown in (2.5.8).

(a) Express the error by periodic interpolation:

Let $f_h = \mathcal{B}_{h,n}f = \Phi + \Psi$, with $\Phi = \sum\limits_{i=1}^{N-1} c_i\phi_i$ and $\Psi = \sum\limits_{j=0}^{n} d_j\psi_j$, and $f_E = f - \Psi$. It follows immediately that

$$\|(f_h - f)^{(k)}\|_{L_2[0,1]} = \|(\Phi - f_E)^{(k)}\|_{L_2[0,1]}. \tag{2.5.3}$$

Assume further that

$$\widehat{\Phi}(x) = \begin{cases} \Phi(x) & x \in [0,1] \\ -\Phi(-x) & x \in [-1,0) \end{cases}, \tag{2.5.4}$$

and

$$\widehat{f_E}(x) = \begin{cases} f_E(x) & x \in [0,1] \\ -f_E(-x) & x \in [-1,0) \end{cases}. \tag{2.5.5}$$

From (2.5.3) we have

$$\|(f_h - f)^{(k)}\|_{L_2[0,1]} = 1/\sqrt{2}\|(\widehat{\Phi} - \widehat{f_E})^{(k)}\|_{L_2[-1,1]}. \tag{2.5.6}$$

We claim that $\widehat{\Phi}$ is a periodic interpolation of $\widehat{f_E}$ and prove the claim by showing that $\widehat{f_E} \in H^{n+1}_{per}([-1,1])$, $\widehat{\Phi} \in X^{2N,n}_{per}([-1,1])$ and $\widehat{\Phi}(\xi_i) = \widehat{f_E}(\xi_i)$, for $i = -N, \ldots, N$, where $\xi_{-i} = -\xi_i$.

(b) Prove that $\widehat{\Phi} \in X^{2N,n}_{per}([-1,1])$:

From the definition (2.5.4), we know $\widehat{\Phi}$ is a piecewise polynomial defined on interval [-1,1]. We also need to address the smoothness and periodicity of $\widehat{\Phi}$. As for smoothness, due to $\Phi \in X^{N,n}_E[0,1]$, $\Phi^{(2s)}(0) = 0$, which implies $\widehat{\Phi}^{(2s)}(0^+) = \widehat{\Phi}^{(2s)}(0^-) = 0$, for $s = 0, \ldots, r$. From the definition of $\widehat{\Phi}$ we have $\widehat{\Phi}^{(2s+1)}(0^+) = \widehat{\Phi}^{(2s+1)}(0^-)$ for $s = 1, \ldots, r-1$. Then it's derived that $\widehat{\Phi}^{(s)}(0^+) = \widehat{\Phi}^{(s)}(0^-)$ for $s = 0, \ldots, n-1$. Together with the fact that $\widehat{\Phi} \in C^{n-1}[-1,0] \cup C^{n-1}[0,1]$, we have $\widehat{\Phi} \in C^{n-1}[-1,1]$.

As for periodicity, $\Phi \in X^{N,n}_E[0,1]$ implies that $\widehat{\Phi}^{(2s)}(-1^+) = \widehat{\Phi}^{(2s)}(1^-) = 0$, for $s = 0, \ldots, r$. We have $\widehat{\Phi}^{(2s+1)}(-1^+) = \widehat{\Phi}^{(2s+1)}(1^-)$ for $s = 1, \ldots, r-1$ from the definition of $\widehat{\Phi}$. Then it follows that $\widehat{\Phi}^{(s)}(-1^+) = \widehat{\Phi}^{(s)}(1^-)$ for $s = 0, \ldots, n-1$. Also because $\widehat{\Phi}$ is a piecewise polynomial in space $C^{n-1}[-1,1]$, we have $\widehat{\Phi} \in X^{2N,n}_{per}([-1,1])$.

(c) Prove that $\widehat{f_E} \in H^{n+1}_{per}[-1,1]$:

From the definition (2.5.5), we know that $\widehat{f_E}|_{[-1,0]} \in H^{n+1}[-1,0]$ and $\widehat{f_E}|_{[0,1]} \in H^{n+1}[0,1]$. We still need to show the smoothness that $\widehat{f_E} \in C^n[-1,1]$, and the periodicity that $\widehat{f_E}^{(s)}(-1^+) = \widehat{f_E}^{(s)}(1^-) = 0$, for $s = 0, \ldots, n$. As for smoothness, $\Phi \in X^{N,n}_E[0,1]$ and $f_h = \mathcal{B}_{h,n} f$ implies that, for $s = 0, \ldots, r$,

$$\begin{aligned} {f_E}^{(2s)}(0) &= f^{(2s)}(0) - \Psi^{(2s)}(0) \\ &= f^{(2s)}(0) - (f_h^{(2s)}(0) - \Phi^{(2s)}(0)) \\ &= f^{(2s)}(0) - f_h^{(2s)}(0) \\ &= 0. \end{aligned} \tag{2.5.7}$$

This property yields $\widehat{f_E}^{(2s)}(0^+) = \widehat{f_E}^{(2s)}(0^-) = 0$, for $s = 0, \ldots, r$. It's obtained from the definition of $\widehat{f_E}$ that $\widehat{f_E}^{(2s+1)}(0^+) = \widehat{f_E}^{(2s+1)}(0^-)$, for $s = 0, \ldots, r$. Together with the fact that $\widehat{f_E} \in C^n[-1,0] \cup C^n[0,1]$, we have $\widehat{f_E} \in C^n[-1,1]$. As for periodicity, similarly to (2.5.7), we have ${f_E}^{(2s)}(1) = 0$, for $s = 0, \dots, r$, which gives $\widehat{f_E}^{(2s)}(-1^+) = \widehat{f_E}^{(2s)}(1^-) = 0$, for $s = 0, \dots, r$. The definition of $\widehat{f_E}$ implies $\widehat{f_E}^{(2s+1)}(-1^+) = \widehat{f_E}^{(2s+1)}(1^-)$, for $s = 0, \dots, r$. Combining this property with $\widehat{f_E} \in H^{n+1}[-1,0] \cup H^{n+1}[0,1]$ and $\widehat{f_E} \in C^n[-1,1]$, we have $\widehat{f_E} \in H^{n+1}_{per}[-1,1]$.

(d) Prove that $\widehat{\Phi}$ is an interpolant of $\widehat{f_E}$:

Since f_h is an interpolant of f, we have, for $i = 0, \dots, N$,

$$\begin{aligned} f_E(\xi_i) &= f(\xi_i) - \Psi(\xi_i) \\ &= f(\xi_i) - (f_h(\xi_i) - \Phi(\xi_i)) \\ &= \Phi(\xi_i). \end{aligned}$$

Similarly, we have for $i = -N, \dots, 0$ that

$$-f_E(-\xi_i) = -\Phi(-\xi_i).$$

Then it follows from the definitions (2.5.4) and (2.5.5) that $\widehat{\Phi}(\xi_i) = \widehat{f_E}(\xi_i)$, for $i = -N, \dots, N$.

Hence, we conclude that Φ is a periodic B-spline interpolant of $f_E \in H^{n+1}_{per}[-1,1]$.

(e) Estimate the error of non-periodic interpolation:

From Theorem 2.6 of this chapter, we have

$$\|(\widehat{\Phi} - \widehat{f_E})^{(k)}\|_{L_2[-1,1]} \leq C(h/\pi)^{n+1-k} \|\widehat{f_E}^{(n+1)}\|_{L_2[-1,1]}, \tag{2.5.8}$$

where the constant C is independent of function $\widehat{f_E}$. Since Ψ is a polynomial of degree n, the definition of f_E implies

$$\begin{aligned} \|\widehat{f_E}^{(n+1)}\|_{L_2[-1,1]} &= \sqrt{2}\|f_E^{(n+1)}\|_{L_2[0,1]} \\ &= \sqrt{2}\|(f - \Psi)^{(n+1)}\|_{L_2[0,1]} \\ &= \sqrt{2}\|f^{(n+1)}\|_{L_2[0,1]}. \end{aligned} \tag{2.5.9}$$

Combining (2.5.6), (2.5.8) and (2.5.9), we arrived at the result of the theorem.

□

Remark 2.18. *A problem worth considering in the proof is whether the stability of polynomial Φ, which interpolates the function f, may affect the error bound in* (2.5.2). *The answer is no. Notice that the constant C in* (2.5.8), *which is the same constant as in* (2.5.2) *is independent of $\widehat{f_E}$ according to the Theorem 2.6, and therefore it is independent the polynomial Φ. Hence the bound is independent of Ψ.*

2.6 Multivariate Non-periodic B-spline Interpolation

For any $f \in \mathcal{H}^{n+1}([0,1]^d)$, and $\boldsymbol{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$, denote by $\mathcal{B}_{h,n}^{x_i}$ the operator of the univariate B-spline interpolation of function f in terms of variable x_i. Then a tensor-product B-spline interpolant $f_h \in X^{N,n}([0,1]^d)$ is defined to be

$$f_h = \mathcal{B}_{h,n}^{x_1} \dots \mathcal{B}_{h,n}^{x_d} f.$$

We still use the operator $\mathcal{B}_{h,n}$ for the multi-variate interpolation such that $\mathcal{B}_{h,n} f = f_h$.

From the definition of the interpolation operator $\mathcal{B}_{h,n}$, f_h has the property that $f_h(\boldsymbol{\xi_i}) = f(\boldsymbol{\xi_i})$ for $\boldsymbol{i} \in \mathcal{I}^d$ and

$$\begin{aligned} &D^{2\vec{k}.*\vec{\alpha}} f_h((\vec{1} - \vec{\alpha}).*\boldsymbol{\xi_i}) = D^{2\vec{k}.*\vec{\alpha}} f((\vec{1} - \vec{\alpha}).*\boldsymbol{\xi_i}), \\ &\text{and } D^{2\vec{k}.*\vec{\alpha}} f_h((\vec{1} - \vec{\alpha}).*\boldsymbol{\xi_i} + \vec{\alpha}) = D^{2\vec{k}.*\vec{\alpha}} f((\vec{1} - \vec{\alpha}).*\boldsymbol{\xi_i} + \vec{\alpha}), \end{aligned} \tag{2.6.1}$$

where $\vec{\alpha}, \vec{k} \in \mathbb{N}_0^d$, $\|\vec{\alpha}\|_\infty = 1$, $\vec{k} \leq r$, and $\vec{1} = (1, 1, \dots, 1)$. The property (2.6.1) means that all the even degree derivatives of f_h equal these of f at the interpolation points on the boundary. In the special case of $d = 1$, the condition is the same as (2.5.1).

An error estimate of the interpolation is given in the following theorem.

Theorem 2.19. *Letting $f \in \mathcal{H}^{n+1}([0,1]^d)$ with $n = 2r+1$, $r \in \mathbb{N}_0$, and $\vec{k} \in \mathbb{N}_0^d$*

with $\vec{k} \le r+1$, we have

$$\|D^{\vec{k}}(f - \mathcal{B}_{h,n}f)\| \le C \sum_{\vec{\alpha}\in\mathbb{N}_0^d, \|\vec{\alpha}\|_\infty=1} \left(\frac{h}{\pi}\right)^{|\vec{\alpha}|(n+1)-\vec{\alpha}\vec{k}} \|D^{(n+1)\vec{\alpha}-\vec{\alpha}.*\vec{k}+\vec{k}}f\|.$$

Proof. The proof of the theorem is very similar to the proof of Theorem 2.14, only with the modifications that replace $S_{\boldsymbol{x}}$ and S_{x_i} by $\mathcal{B}_{h,n}$ and $\mathcal{B}_{h,n}^{x_i}$, respectively. □

Similar as Corollary 2.15, we have

Corollary 2.20. *Let $f \in \mathcal{H}^{p+1}([0,1]^d)$ with $p = 2r+1$, $r \in \mathbb{N}_0$. Assume there exists $\sigma > 0$, $\|D^{\vec{s}}f\| \le C\sigma^{|\vec{s}|}$ for all $\vec{s} \in \mathbb{N}_0^d$ with $\vec{s} \le p+1$. For any $\vec{k} \in \mathbb{N}_0^d$ with $\vec{k} \le r+1$ and $h\sigma/\pi < 1$, we have*

$$\|D^{\vec{k}}(f - \mathcal{B}_{h,p}f)\| \le C \left(\frac{h\sigma}{\pi}\right)^{p+1-\|\vec{k}\|_\infty}.$$

In particular, if $\|D^{\vec{s}}f\| \le \sigma^{|\vec{s}|}\|f\|$, we have

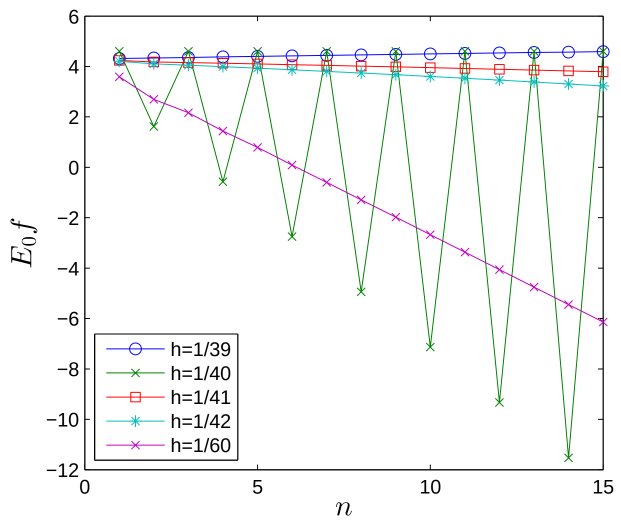
$$\|D^{\vec{k}}(f - \mathcal{B}_{h,p}f)\| \le C \left(\frac{h\sigma}{\pi}\right)^{p+1-\|\vec{k}\|_\infty} \|f\|.$$

2.7 Numerical Experiment Results

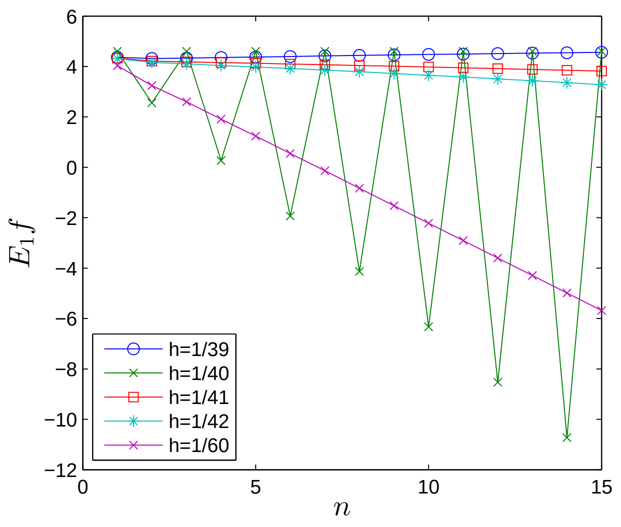
The following experiment is designed to investigate, for B-spline interpolation, how the error converges as the degree of the spline increases and how the different sizes of mesh affect the convergence. Let the error of f be

$$E_0 f = \log\left(\|f - f_h\|/\|f\|\right) \text{ and } E_1 f = \log\left(\|\nabla(f - f_h)\|/\|\nabla f\|\right), \tag{2.7.1}$$

where f_h is $S_{h,n}f$ for periodic interpolation and $\mathcal{B}_{h,n}f$ for non-periodic case. We interpolate a sufficiently regular function and compute the error of approximation. We plot the error against the degree of the splines, and moreover, adjust h to to check the convergence in each case.



(a) Error $E_0 f$



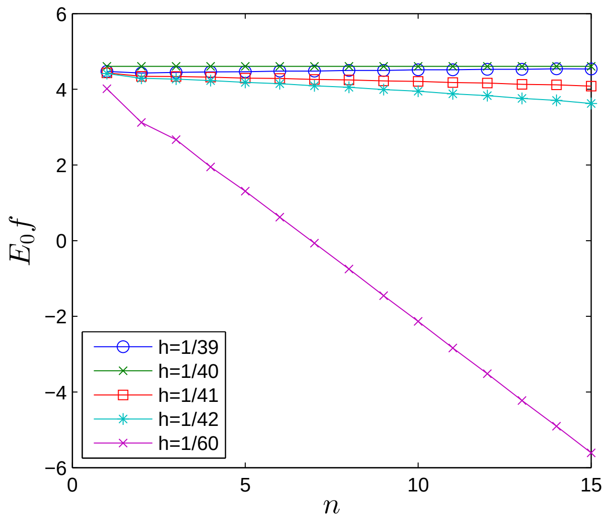
(b) Error $E_1 f$

Figure 2.1: Errors of B-spline interpolation in $\mathbb{R}$

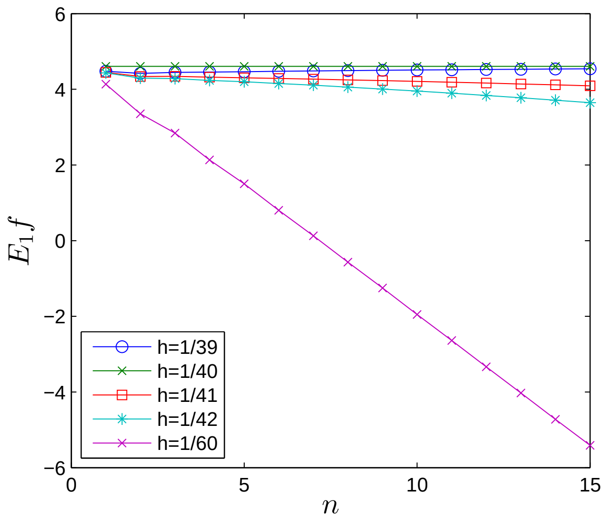
2.7.0.1 Univariate Function

We first choose the interpolated function to be $f = \sin(40\pi x)$ and implement Algorithm 2.1. Figures 2.1 shows that the error $E_0 f$ and $E_1 f$ converges when the value of h range from 1/41 to 1/60, and diverge when $h = 1/39$. That behaviour agrees with Corollary 2.7, which implies that the error $E_0 f$ and $E_1 f$ is bounded by $\mathcal{O}((40h)^{n+1})$ and $\mathcal{O}((40h)^n)$, respectively. Hence, as we increase n, if $h < 1/40$, both the errors are predicted to converge to zero. Moreover, the linear decrease of the logarithm error suggests that the approximation converges to f exponentially.

The figures also show that when $h = 1/40$, the error oscillates as the degree increases. In particular, when the degree p is odd the error stays high and unchanged, but when the degree is even the error converges. This is because, when degree is odd, the interpolation point is $\xi_i = i/40$, and so the corresponding data is $f(\xi_i) = \sin(\pi i) = 0$. The interpolation in this case gives the result of $f_h = 0$ due to an aliasing effect. The phenomenon agrees with Lemma 4.4 [34], which, in our



(a) Error $E_0 f$



(b) Error $E_1 f$

Figure 2.2: Error of interpolation in $\mathbb{R}^2$

case, is interpreted as

$$\begin{cases} \lim\limits_{n\to\infty} S_{h,2n}\sin(40\pi x) = 0 \\ \lim\limits_{n\to\infty} S_{h,2n+1}\sin(40\pi x) = \sin(40\pi x) \end{cases}.$$

2.7.0.2 Multivariate Function

For the interpolation of a two variate function, the interpolated function is chosen to be $f = \sin(40\pi x)\cos(40\pi y)$. In figure 2.2, we observe a similar behaviour of errors in univariate function case. That is because $\|f^{(\vec{\alpha})}\| = (40\pi)^{|\alpha|}\|f\|$ for all $\vec{\alpha} \in \mathbb{N}_0^2$. According to Corollary 2.15, the errors are bounded by the term $\mathcal{O}((40h)^{n+1})$ and $\mathcal{O}((40h)^n)$. So, when $h < 1/40$, the errors converge exponentially.

2.7.0.3 Non-Periodic Function

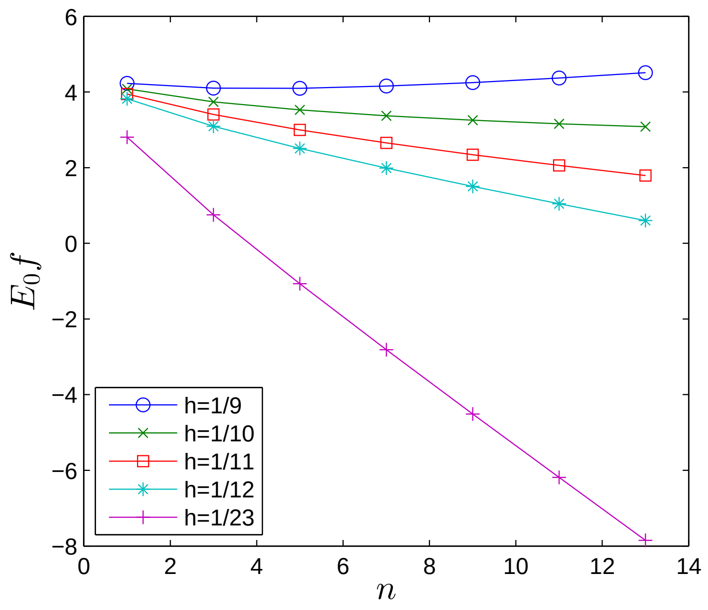


Figure 2.3: Errors of B-spline interpolation

We choose $f = \exp(10\pi x)$, for which $\|f^{(k)}\| = (10\pi)^k \|f\|$ for $k \in \mathbb{N}_0$. According to Corollary 2.20, the error $\|f - \mathcal{B}_{h,n} f\|/\|f\|$ is bounded by $\mathcal{O}((10h)^n)$. That means for $h < 1/10$, as we increase the degree of B-spline n, the error converges to zero. Figure 2.3 confirms this behaviour.

Chapter 3

B-spline Finite Element Method for the Heat Equation

The heat equation is a form of diffusion equation governing the temperature distribution in an object. For more detail, the reader may consult, for example, [31, 74]. We consider two types of boundary conditions, periodic and Dirichlet, where the periodic case is studied first.

3.1 Problem with Periodic Boundary Condition

3.1.1 Model Problem

The problem under consideration is the heat equation

$$\frac{\partial}{\partial t}u(\boldsymbol{x},t) - \Delta u(\boldsymbol{x},t) = \eta(\boldsymbol{x},t),\ (\boldsymbol{x},t) \in \mathbb{T}^d \times [0,\infty), \tag{3.1.1}$$

with initial value

$$u(\boldsymbol{x},0) = u_0(\boldsymbol{x}).$$

The initial value $u_0(\boldsymbol{x})$ and the non-homogeneous term $\eta(\boldsymbol{x},t)$ are assumed to be functions in the spaces $\mathcal{H}^{p+1}(\mathbb{T}^d)$ and $L_2(0,T;\mathcal{H}^{p+1}(\mathbb{T}^d))$ respectively.

A weak formulation corresponding to the heat equation is to find $u(t) \in L_2(0,T;\mathcal{H}^1(\mathbb{T}^d))$ with $\dot{u}(t) \in L_2(0,T;\mathcal{H}^{-1}(\mathbb{T}^d))$ and $u(0) = u_0 \in L_2(\mathbb{T}^d)$ such that

$$\frac{d}{dt}(u(t),v) + a(u(t),v) = (\eta(t),v) \qquad \text{for any } v \in \mathcal{H}^1(\mathbb{T}^d), \tag{3.1.2}$$

where $\mathcal{H}^{-1}(\mathbb{T}^d)$ is the dual space of $\mathcal{H}^1(\mathbb{T}^d)$, and $\dot{u}(t)$ is the derivative of u w.r.t. temporal variable t. According to Proposition 23.28 in [92] the weak formulation has a unique solution $u(t) \in L_2(0,T;\mathcal{H}^1(\mathbb{T}^d))$. Moreover, since the non-homogeneous term $\eta(t) \in L_2(0,T;\mathcal{H}^{p+1}(\mathbb{T}^d))$, using the same proposition it can be derived that the solution $u(t) \in L_2(0,T;\mathcal{H}^{p+1}(\mathbb{T}^d))$.

Given the weak formulation (3.1.2), we derive a numerical scheme to approximate its solution, by discretizing the problem using a B-spline finite element method for the spatial variable and a finite difference method for the temporal variable. We first give the scheme for one-dimensional problem, and then with the results obtained, the scheme for the multi-dimensional case.

3.1.2 Scheme for the Equation on One-dimensional Domain

We are using the Galerkin method shown in Section 1.4.1 to approximate the solution of (3.1.2). An approximation is obtained by finding $u_h(t) \in X^{N,p}(\mathbb{T})$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}(u_h(t),v) + a(u_h(t),v) = (\eta(t),v) \qquad \text{for any } v \in X^{N,p}(\mathbb{T}).$$

Recall that $\mathcal{I} = \{0,\ldots,N-1\}$. Let $u_h(t) = \sum_{j\in\mathcal{I}} \alpha_j(t) b_{j,p}$ and $v = b_{i,p}$ for each $i \in \mathcal{I}$. This leads to a system of differential equations which is written as

$$\boldsymbol{M}\dot{\vec{\alpha}}(t) + \boldsymbol{S}\vec{\alpha}(t) = \vec{l}(t), \tag{3.1.3}$$

with $\dot{\vec{\alpha}}(t) = (\dot{\alpha}_0(t),\ldots,\dot{\alpha}_{N-1}(t))^T$. $\boldsymbol{M}$ and $\boldsymbol{S}$ are the mass and the stiffness matrices with entries

$$\boldsymbol{M}_{i,j} = \int_0^1 b_{i,p}(x) b_{j,p}(x)\mathrm{d}x \text{ and } \boldsymbol{S}_{i,j} = \int_0^1 b'_{i,p}(x) b'_{j,p}(x)\mathrm{d}x.$$

$\vec{l}(t)$ is the load vector with entries

$$l_i(t) = \int_0^1 b_{i,p}(x)\eta(x,t)\mathrm{d}x. \tag{3.1.4}$$

Since it is inconvenient and sometimes impossible to calculate the integrals in the load vector analytically, for each $t \in [0, \infty)$, we approximate the non-homogeneous term $\eta(t)$ with an element $\eta_h(t) \in X^{N,p}(\mathbb{T})$. Letting $\eta_h(t) = \sum_{i\in\mathcal{I}} \beta_i(t) b_{i,p}(x)$, the system (3.1.3) is modified to

$$\boldsymbol{M}\dot{\vec{\alpha}}(t) + \boldsymbol{S}\vec{\alpha}(t) = \boldsymbol{M}\vec{\beta}(t), \tag{3.1.5}$$

with entries for the mass matrix $\boldsymbol{M} \in \mathbb{R}^{N\times N}$,

$$\boldsymbol{M}_{i,j} = \int_0^1 b_{i,p}(x) b_{j,p}(x) dx,$$

and for the stiffness matrix $\boldsymbol{S} \in \mathbb{R}^{N\times N}$,

$$\boldsymbol{S}_{i,j} = \int_0^1 b'_{i,p}(x) b'_{j,p}(x) dx. \tag{3.1.6}$$

Using B-spline interpolation as the approximation for the non-homogeneous term $\eta(t)$ for each $t \in [0, \infty)$, the vector $\vec{\beta}(t)$ is obtained from interpolating $\eta(\cdot, t)$, namely,

$$\vec{\beta}(t) = \boldsymbol{G}^{-1}\vec{\eta}(t),$$

with

$$\eta_j(t) = \eta(\xi_j, t),$$

where $\boldsymbol{G}$ is given in (2.3.3). To evaluate $\boldsymbol{M}$ efficiently, choose $x = 0$ in Property 4(iv) in Page 16. It follows that

$$\boldsymbol{M}_{i,j} = h b_{i-j-(p+1),2p+1}(0). \tag{3.1.7}$$

By using Property 4(vi) in Page 16 for both $b'_{i,p}$ and $b'_{j,p}$ in (3.1.6), and expanding the result, entries of the stiffness matrix are given by

$$\begin{aligned} \boldsymbol{S}_{i,j} &= 1/h^2 (\textstyle\int_0^1 b_{i,p-1}(x) b_{j,p-1}(x) dx + \int_0^1 b_{i,p-1}(x) b_{j+1,p-1}(x) dx \\ &\quad + \textstyle\int_0^1 b_{i+1,p-1}(x) b_{j,p-1}(x) dx + \int_0^1 b_{i+1,p-1}(x) b_{j+1,p-1}(x) dx) \\ &= 1/h (2 b_{i-j-p,2p-1}(0) \\ &\qquad - b_{i-j+1-p,2p-1}(0) - b_{i-j-1-p,2p-1}(0)). \end{aligned} \tag{3.1.8}$$

Some properties of the matrices $\boldsymbol{M}$ and $\boldsymbol{S}$ are given as follows. Property 4(i) in Page 16 implies that

$$\boldsymbol{M}_{i,[i+k]_N} = \boldsymbol{S}_{i,[i+k]_N} = 0 \text{ for } |k| > p, \tag{3.1.9}$$

where $[i]_N = i \mod N$. That means if N is greatly larger than p, the matrix is sparse.

From (3.1.7), (3.1.8) and Property 4(iii) in Page 16, we have that

$$\boldsymbol{M}_{i,[i+k]_N} = \boldsymbol{M}_{i,[i-k]_N} \text{ and } \boldsymbol{S}_{i,[i+k]_N} = \boldsymbol{S}_{i,[i-k]_N}. \tag{3.1.10}$$

Property 4(ii) in Page 16 together with (3.1.7) and (3.1.8) imply that $\boldsymbol{M}$ and $\boldsymbol{S}$ are circulant matrices, that is,

$$\boldsymbol{M}_{i,j} = \boldsymbol{M}_{[i+k]_N,[j+k]_N} \text{ and } \boldsymbol{S}_{i,j} = \boldsymbol{S}_{[i+k]_N,[j+k]_N}. \tag{3.1.11}$$

So is $\boldsymbol{G}$, as shown in (2.3.4). The property of circulant matrices in (1.3.2) implies that the matrices $\boldsymbol{M}$, $\boldsymbol{S}$ and $\boldsymbol{G}$ are able to be diagonalized as

$$\boldsymbol{M} = \boldsymbol{F}^{-1}\boldsymbol{\Lambda}_M\boldsymbol{F} \qquad \boldsymbol{S} = \boldsymbol{F}^{-1}\boldsymbol{\Lambda}_S\boldsymbol{F} \quad \text{and} \quad \boldsymbol{G} = \boldsymbol{F}^{-1}\boldsymbol{\Lambda}_G\boldsymbol{F}. \tag{3.1.12}$$

Matrices $\boldsymbol{\Lambda}_M$, $\boldsymbol{\Lambda}_S$ and $\boldsymbol{\Lambda}_G$ are diagonal matrices, whose diagonal entries are eigenvalues corresponding to $\boldsymbol{M}$, $\boldsymbol{S}$ and $\boldsymbol{G}$ respectively.

Taking the expressions (3.1.12) into scheme (3.1.5), the semi-discrete scheme is modified to

$$\boldsymbol{\Lambda}_M\dot{\widehat{\alpha}}(t) + \boldsymbol{\Lambda}_S\widehat{\alpha}(t) = \boldsymbol{\Lambda}_M\boldsymbol{\Lambda}_G^{-1}\widehat{\eta}(t), \tag{3.1.13}$$

with $\widehat{\alpha}(t) = \boldsymbol{F}\vec{\alpha}(t)$ and $\widehat{\eta}(t) = \boldsymbol{F}\vec{\eta}(t)$. Since all the matrices are diagonal, the system is equivalent to a series of independent scalar ordinary differential equations (ODE).

We choose the θ method to solve the system numerically. The reader may refer to [88] for more discussions. Note that when the time step δt decreases, the θ method converges algebraically with order 2 if $\theta = 1/2$, and with order 1 otherwise. However, as shown later on, the spatial error may converge exponentially when

p is increased, so the error of temporal discretization converges more slowly than the spatial one. We only choose a low order time-stepping scheme because the thesis focuses on the B-spline approximation of the spatial variable, and high order numerical schemes for temporal variables are not in the scope of these topics. A wide array of numerical methods are applicable for the system at this stage. The reader may refer to the literature, for example, [40, 38] for more discussion.

Suppose we want to approximate the solution at time T. We divide the time interval $[0,T]$ into m time steps, with the step size $\delta t = T/m$. Let $t_k = k\delta t$. Applying the θ-method for the equations in (3.1.13), we have the scheme

$$\begin{aligned}\widehat{\alpha}^{k+1} = \;& (\mathbf{\Lambda}_M + (1-\theta)\delta t\mathbf{\Lambda}_S)^{-1}(\mathbf{\Lambda}_M - \theta\delta t\mathbf{\Lambda}_S)\widehat{\alpha}^k \\ & +((\mathbf{\Lambda}_M + (1-\theta)\delta t\mathbf{\Lambda}_S)\mathbf{\Lambda}_G)^{-1}\delta t\mathbf{\Lambda}_M((1-\theta)\widehat{\eta}^{k+1} + \theta\widehat{\eta}^k),\end{aligned} \tag{3.1.14}$$

with $\widehat{\eta}^k = \widehat{\eta}(t_k)$. The initial step $\vec{\alpha}^0$ is obtained by interpolating the initial value u_0. Since all the matrices in (3.1.14) are diagonal matrices, the scheme is equivalent to a series of independent scaler recurrence relations. Therefore, in implementation, one may store the matrices as a vector to save storage space. The algorithm for the B-spline scheme to approximate the solution at a time $T = m\delta t$ is summarized as follows.

Algorithm 3.1. *1. Construct the vector $\vec{v}_M$, $\vec{v}_S$ and $\vec{v}_G$ by applying FFT to the fist row of matrices $\boldsymbol{M}$, $\boldsymbol{S}$ and $\boldsymbol{G}$ respectively which are obtained using the relation* (3.1.7), (3.1.8), *and* (2.3.3) *together with de Boor's algorithm for B-spline evaluation.*

2. Obtain the initial value $\vec{\alpha}^0$ by interpolation Algorithm 2.1, and then use FFT to transform $\vec{\alpha}^0$ by $\widehat{\alpha}^0 = \boldsymbol{F}\vec{\alpha}^0$.

3. For each time step k from 0 to $m-1$,

- *Apply FFT to give $\widehat{\eta}^k = \boldsymbol{F}\vec{\eta}^k$.*
- *Generate $\widehat{\alpha}^{k+1}$ using the scalar iteration shown in scheme* (3.1.14) *where*

the scalar value corresponding to diagonal value of $\mathbf{\Lambda}_M$, $\mathbf{\Lambda}_S$ *and* $\mathbf{\Lambda}_G$ *is given by* $\vec{v}_M$, $\vec{v}_S$ *and* $\vec{v}_G$ *respectively.*

4. *Compute the coefficients for B-spline approximation of the solution* $u(\cdot, T)$ *with applying FFT for* $\vec{\alpha}^m = \boldsymbol{F}^{-1}\widehat{\alpha}^m$.

In each time step the calculation takes $\mathcal{O}(N \log N)$ operations. To see this, we note that the dominant cost comes from calculating $\widehat{\eta}^k$ using the FFT and applying scheme (3.1.14). As mentioned in Section 1.3.1, it costs $\mathcal{O}(N \log N)$ to compute the product of a circulant matrix and a vector using FFT. So the calculation of $\widehat{\eta}^k$ needs $\mathcal{O}(N \log N)$ operations. As for applying the scheme (3.1.14), the scheme is actually equivalent to calculating N scalar iteration equations. The cost for this part is therefore $\mathcal{O}(N)$. In the particular case when the non-homogeneous term $\eta = 0$, the cost of each step is only $\mathcal{O}(N)$.

3.1.3 Scheme for the Equation on Multi-dimensional Domain

We are looking for an approximation by finding $u_h(t) \in X^{N,p}(\mathbb{T}^d)$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}(u_h(t), v) + a(u_h(t), v) = (\eta(t), v) \qquad \text{for any } v \in X^{N,p}(\mathbb{T}^d). \tag{3.1.15}$$

We apply the Galerkin method with multivariate tensor product B-spline $\{b_{i,p}(\boldsymbol{x})\}_{i=0}^{N^d-1}$ defined in (1.5.3) as basis and interpolate the non-homogeneous term $\eta(t)$ and initial value $u_0(\boldsymbol{x})$ with the B-spline. In a similar way to Section 3.1.2, we arrive at a semi-discrete scheme,

$$\boldsymbol{\mathcal{M}}\dot{\vec{\alpha}}(t) + \boldsymbol{\mathcal{S}}\vec{\alpha}(t) = \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{G}}^{-1}\vec{\eta}(t). \tag{3.1.16}$$

$\vec{\eta}_j(t) = \eta(\boldsymbol{\xi}_j, t)$ and $\boldsymbol{\xi}_i \in \mathbb{R}^d$ are the points of the interpolation defined in Section 2.4.1. The mass matrix $\boldsymbol{\mathcal{M}}$ and stiffness matrix $\boldsymbol{\mathcal{S}}$ have the entries

$$\boldsymbol{\mathcal{M}}_{ij} = \int_{\mathbb{T}^d} b_{i,p}(\boldsymbol{x}) b_{j,p}(\boldsymbol{x}) d\boldsymbol{x} \text{ and } \boldsymbol{\mathcal{S}}_{ij} = \int_{\mathbb{T}^d} \nabla b_{i,p}(\boldsymbol{x}) \nabla b_{j,p}(\boldsymbol{x}) d\boldsymbol{x}. \tag{3.1.17}$$

Recall that for each $i, j \in \{0, \dots, N^d-1\}$, $i = \sigma_N(i_1, \dots, i_d)$ and $j = \sigma_N(j_1, \dots, j_d)$, where $\sigma_N(x)$ is the function defined in (1.3.4). Expressing $b_{i,p}(\boldsymbol{x})$ and $b_{j,p}(\boldsymbol{x})$ in equation (3.1.17) using their definition and rearranging the multiple integral gives

$$\begin{aligned}\boldsymbol{\mathcal{M}}_{i,j} &= \int_0^1 b_{i_1,p}(x_1)b_{j_1,p}(x_1)dx_1 \dots \int_0^1 b_{i_d,p}(x_d)b_{j_d,p}(x_d)dx_d \\ &= \boldsymbol{M}_{i_1,j_1} \dots \boldsymbol{M}_{i_d,j_d}.\end{aligned}$$

Together with the definition of the Kronecker product, this equality implies

$$\boldsymbol{\mathcal{M}} = \underbrace{\boldsymbol{M} \otimes \boldsymbol{M} \otimes \cdots \otimes \boldsymbol{M}}_{d}. \tag{3.1.18}$$

Similarly, expanding $b_{i,p}(\boldsymbol{x})$ and $b_{j,p}(\boldsymbol{x})$ using their definition and rearranging the result leads to

$$\begin{aligned}\boldsymbol{\mathcal{S}}_{i,j} &= \sum_{s=1}^{d} \int_{\mathbb{T}^d} \tfrac{\partial}{\partial x_s}(b_{i_1,p}(x_1) \dots b_{i_d,p}(x_d)) \tfrac{\partial}{\partial x_s}(b_{j_1,p}(x_1) \dots b_{j_d,p}(x_d)) d\boldsymbol{x} \\ &= \sum_{s=1}^{d} \underbrace{\boldsymbol{M}_{i_1,j_1} \dots \boldsymbol{M}_{i_{s-1},j_{s-1}} \boldsymbol{S}_{i_s,j_s}}_{s} \boldsymbol{M}_{i_{s+1},j_{s+1}} \dots \boldsymbol{M}_{i_d,j_d},\end{aligned}$$

which implies

$$\boldsymbol{\mathcal{S}} = \sum_{s=1}^{d} \underbrace{\boldsymbol{M} \otimes \cdots \otimes \boldsymbol{M} \otimes \boldsymbol{S}}_{s} \otimes \boldsymbol{M} \otimes \dots \otimes \boldsymbol{M}.$$

As before, we seek to diagonalize the matrices $\boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{G}}$ in the scheme (3.1.16). For the mass matrix $\boldsymbol{\mathcal{M}}$ in (3.1.18), diagonalization formula of $\boldsymbol{M}$ in (3.1.12) and Property 2(iv) in Page 8 gives

$$\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{F}}^{-1} \boldsymbol{\Lambda}_{\mathcal{M}} \boldsymbol{\mathcal{F}}, \tag{3.1.19}$$

with

$$\boldsymbol{\mathcal{F}} = \underbrace{\boldsymbol{F} \otimes \boldsymbol{F} \otimes \cdots \otimes \boldsymbol{F}}_{d},$$

and

$$\boldsymbol{\Lambda}_{\mathcal{M}} = \underbrace{\boldsymbol{\Lambda}_M \otimes \boldsymbol{\Lambda}_M \otimes \cdots \otimes \boldsymbol{\Lambda}_M}_{d}. \tag{3.1.20}$$

Similarly, we found that

$$\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{F}}^{-1} \boldsymbol{\Lambda}_{\mathcal{S}} \boldsymbol{\mathcal{F}} \text{ and } \boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{F}}^{-1} \boldsymbol{\Lambda}_{\mathcal{G}} \boldsymbol{\mathcal{F}}, \tag{3.1.21}$$

where

$$\mathbf{\Lambda}_{\mathcal{S}} = \sum_{s=1}^{d} \underbrace{\mathbf{\Lambda}_M \otimes \cdots \otimes \mathbf{\Lambda}_M \otimes \mathbf{\Lambda}_S}_{s} \otimes \mathbf{\Lambda}_M \otimes \ldots \otimes \mathbf{\Lambda}_M, \tag{3.1.22}$$

and

$$\mathbf{\Lambda}_{\mathcal{G}} = \underbrace{\mathbf{\Lambda}_G \otimes \mathbf{\Lambda}_G \otimes \cdots \otimes \mathbf{\Lambda}_G}_{d}. \tag{3.1.23}$$

Substituting the diagonalization forms of $\boldsymbol{\mathcal{M}}$, $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{G}}$, the time stepping scheme (3.1.16) is modified to

$$\mathbf{\Lambda}_{\mathcal{M}} \dot{\widehat{\alpha}}(t) + \mathbf{\Lambda}_{\mathcal{S}} \widehat{\alpha}(t) = \mathbf{\Lambda}_{\mathcal{M}} \mathbf{\Lambda}_{\mathcal{G}}^{-1} \widehat{\eta}(t), \tag{3.1.24}$$

with $\widehat{\alpha}(t) = \boldsymbol{\mathcal{F}} \vec{\alpha}(t)$ and $\widehat{\eta}(t) = \boldsymbol{\mathcal{F}} \vec{\eta}(t)$. This is a series of independent scalar ODEs.

Applying the θ-method for approximating the solution at the time $T = m\delta t$, we obtain the scheme

$$\begin{aligned} \widehat{\alpha}^{k+1} = \;& (\mathbf{\Lambda}_{\mathcal{M}} + (1-\theta)\delta t \mathbf{\Lambda}_{\mathcal{S}})^{-1} (\mathbf{\Lambda}_{\mathcal{M}} - \theta \delta t \mathbf{\Lambda}_{\mathcal{S}}) \widehat{\alpha}^k \\ & + ((\mathbf{\Lambda}_{\mathcal{M}} + (1-\theta)\delta t \mathbf{\Lambda}_{\mathcal{S}}) \mathbf{\Lambda}_{\mathcal{G}})^{-1} \delta t \mathbf{\Lambda}_{\mathcal{M}} ((1-\theta) \widehat{\eta}^{k+1} + \theta \widehat{\eta}^k), \end{aligned} \tag{3.1.25}$$

with $\widehat{\eta}^k = \boldsymbol{\mathcal{F}} \vec{\eta}^k$, $\vec{\eta}_j^k = \eta(\boldsymbol{\xi}_j, k\delta t)$, and $\widehat{\alpha}^0 = \boldsymbol{\mathcal{F}} \vec{\alpha}^0$. The initial vector $\vec{\alpha}^0$ is obtained by interpolating the initial value u_0.

The algorithm is summarized as

Algorithm 3.2. *1. Compute the vector storing the diagonal of $\mathbf{\Lambda}_{\mathcal{M}}$ using the formula $\vec{v}_{\mathcal{M}} = \vec{v}_M \otimes \ldots \otimes \vec{v}_M$ where $\vec{v}_M$ is obtained using Step 1 in Algorithm 3.1. Use the same procedure to obtain $\vec{v}_{\mathcal{S}}$ and $\vec{v}_{\mathcal{G}}$ corresponding to $\mathbf{\Lambda}_{\mathcal{S}}$ and $\mathbf{\Lambda}_{\mathcal{G}}$, respectively.*

2. Obtain the initial value $\vec{\alpha}^0$ with interpolation in Algorithm 2.13, and then use FFT to transform $\vec{\alpha}^0$ by $\widehat{\alpha}^0 = \boldsymbol{\mathcal{F}} \vec{\alpha}^0$.

3. For each time step k from 0 to $m-1$,

- *Apply FFT to give $\widehat{\eta}^k = \boldsymbol{\mathcal{F}} \vec{\eta}^k$.*
- *Generate $\widehat{\alpha}^{k+1}$ using the scalar relation implied by the scheme (3.1.25) where the scalar value corresponding to the diagonal value of $\mathbf{\Lambda}_{\mathcal{M}}$, $\mathbf{\Lambda}_{\mathcal{S}}$ and $\mathbf{\Lambda}_{\mathcal{G}}$ is given by $\vec{v}_{\mathcal{M}}$, $\vec{v}_{\mathcal{S}}$ and $\vec{v}_{\mathcal{G}}$ respectively.*

4. *Compute the coefficients for B-spline approximation of the solution $u(\cdot, T)$ applying FFT for $\vec{\alpha}^m = \boldsymbol{\mathcal{F}}^{-1}\widehat{\alpha}^m$.*

In each time step, calculation takes $\mathcal{O}(dN^d \log N)$ operations, because the calculation of $\widehat{\eta}^k = \boldsymbol{\mathcal{F}}\vec{\eta}^k$ takes $\mathcal{O}(dN^d \log N)$ operations as shown in Section 1.3.1, and the scheme (3.1.25) takes $\mathcal{O}(dN^d)$ operations. When the non-homogeneous term $\eta = 0$, the cost is only $\mathcal{O}(N^d)$.

3.1.4 Semi-discrete Error

The scheme's error can be viewed as the composition of errors of the spatial variable discretization and temporal discretization. In this section we investigate the former, which is the error of semi-discrete approximation.

Recall that $u(t)$ is the exact solution of the equation (3.1.1) and $u_h(t)$ is the solution of the semi-discrete scheme (3.1.16). Denote by $R_h : \mathcal{H}^{p+1}(\mathbb{T}^d) \to X^{N,p}(\mathbb{T}^d)$ the operator such that

$$a\left(R_h f, v\right) = a\left(f, v\right) \text{ for any } v \in X^{N,p}(\mathbb{T}^d). \tag{3.1.26}$$

Our error analysis follows the approach in [87, Ch.1], which divides the error into two parts as

$$u_h(t) - u(t) = \omega(t) + \rho(t), \tag{3.1.27}$$

with $\omega(t) = u_h(t) - R_h u(t)$ and $\rho(t) = R_h u(t) - u(t)$, and then estimates them separately.

We first seek to bound the term $|\rho(t)|_1$, which as shown in a later paragraph yields an estimation for $\|\rho(t)\|$. It follows from (3.1.26) that $R_h f - f$ is orthogonal to the space $X^{N,p}(\mathbb{T}^d)$ in the sense that

$$a\left(R_h f - f, v\right) = 0 \text{ for any } v \in X^{N,p}(\mathbb{T}^d).$$

This property together with the Cauchy-Schwarz inequality reveals

$$\begin{aligned} |R_h f - f|_1^2 &= a(R_h f - f, v - f) \\ &\le |R_h f - f|_1 |v - f|_1 \text{ for any } v \in X^{N,p}(\mathbb{T}^d). \end{aligned}$$

That is

$$|R_h f - f|_1 \leq |v - f|_1 \text{ for any } v \in X^{N,p}(\mathbb{T}^d).$$

Choosing $v = S_{h,p} f$ gives

$$|R_h f - f|_1 \leq |S_{h,p} f - f|_1. \tag{3.1.28}$$

The following lemma produces a bound for $\|\rho(t)\|$.

Lemma 3.3. *Let $f \in \mathcal{H}^{p+1}(\mathbb{T}^d)$. Then*

$$\|R_h f - f\| \leq C|S_{h,p} f - f|_1.$$

Proof. On the hypercube $\Omega = [0,1]^d$, let ψ be the solution of the equation

$$-\Delta\psi = R_h f - f, \tag{3.1.29}$$

with the boundary conditions, for $s = 1, 2, \ldots, d$,

$$\psi|_{x_s=0} = \psi|_{x_s=1} \text{ and } \tfrac{\partial}{\partial x_s}\psi|_{x_s=0} = \tfrac{\partial}{\partial x_s}\psi|_{x_s=1}. \tag{3.1.30}$$

These conditions together with Green's formula yields

$$a(\psi, v) = -(\Delta\psi, v) \text{ for any } v \in \mathcal{H}^p(\mathbb{T}^d).$$

Then the Cauchy-Schwarz inequality implies

$$\begin{aligned} \|R_h f - f\|^2 &= -\left(R_h f - f, \Delta\psi\right) \\ &= a\left(R_h f - f, \psi\right) \\ &\leq |R_h f - f|_1 |\psi|_1. \end{aligned} \tag{3.1.31}$$

To estimate the term $|\psi|_1$, the Bramble-Hilbert lemma (Lemma 4.27 [39]) together with (3.1.29) imply

$$\begin{aligned} |\psi|_1 &\leq C|\psi|_2 \\ &= C\|R_h f - f\|, \end{aligned}$$

where C is independent of ψ.

Substituting this inequality and (3.1.28) into (3.1.31), and cancelling the term $\|R_h f - f\|$, we arrive at the result of the Lemma. □

Lemma 3.3 implies

$$\|\rho(t)\| \le C|S_{h,p}u(t) - u(t)|_1.$$

A bound for $\|\omega(t)\|$ is given by the following lemma.

Lemma 3.4.

$$\begin{aligned}\|\omega(t)\| \quad &\le C(\|u_0 - S_{h,p}u_0\| + |u_0 - S_{h,p}u_0|_1 \\ &\quad + \textstyle\int_0^t |\dot{u}(\tau) - S_{h,p}\dot{u}(\tau)|_1 \mathrm{d}\tau + \int_0^t \|\eta(\tau) - S_{h,p}\eta(\tau)\| \mathrm{d}\tau).\end{aligned}$$

Proof. For any $\chi \in X^{N,p}(\mathbb{T}^d)$, using the definition of $\omega(t)$, (3.1.15) and (3.1.26), we have

$$\begin{aligned}(\dot{\omega}(t), \chi) + a(\omega(t), \chi) \quad &= (\dot{u}_h(t), \chi) + a(u_h(t), \chi) - \left(\tfrac{\partial}{\partial t}(R_h u(t)), \chi\right) - a(R_h u(t), \chi) \\ &= (\eta_h(t), \chi) - \left(\tfrac{\partial}{\partial t}(R_h u(t)), \chi\right) - a(u(t), \chi).\end{aligned}$$

Rearranging the inequality, the weak formulation (3.1.2) implies

$$\begin{aligned}(\dot{\omega}(t), \chi) + a(\omega(t), \chi) \quad &= (\eta(t), \chi) - a(u(t), \chi) - \left(\tfrac{\partial}{\partial t}(R_h u(t)), \chi\right) + (\eta_h(t) - \eta(t), \chi) \\ &= (\dot{u}(t), \chi) - \left(\tfrac{\partial}{\partial t}(R_h u(t)), \chi\right) + (\eta_h(t) - \eta(t), \chi) \\ &= \left(\tfrac{\partial}{\partial t}(u(t) - R_h u(t)), \chi\right) + (\eta_h(t) - \eta(t), \chi).\end{aligned}$$

Choose $\chi = \omega(t)$. Then

$$(\dot{\omega}(t), \omega(t)) + |\omega(t)|_1^2 = -\left(\frac{\partial}{\partial t}(u(t) - R_h u(t)), \omega(t)\right) + (\eta_h(t) - \eta(t), \omega(t)).$$

The equality together with the fact

$$\|\omega(t)\| \left(\frac{\mathrm{d}}{\mathrm{d}t}\|\omega(t)\|\right) = (\dot{\omega}(t), \omega(t))$$

and the Cauchy-Schwarz inequality leads to

$$\|\omega(t)\|(\frac{\mathrm{d}}{\mathrm{d}t}\|\omega(t)\|) + |\omega(t)|_1^2 \le \|\dot{u}(t) - R_h\dot{u}(t)\|\|\omega(t)\| + \|\eta_h(t) - \eta(t)\|\|\omega(t)\|.$$

Since $|\omega(t)|_1^2$ is non-negative, it follows that

$$\frac{\mathrm{d}}{\mathrm{d}t}\|\omega(t)\| \le \|\dot{u}(t) - R_h\dot{u}(t)\| + \|\eta_h(t) - \eta(t)\|.$$

Integrating the inequality with respect to t gives

$$\|\omega(t)\| \leq \|\omega(0)\| + \int_0^t \|\dot{u}(\tau) - R_h\dot{u}(\tau)\|d\tau + \int_0^t \|\eta_h(\tau) - \eta(\tau)\|d\tau. \qquad (3.1.32)$$

Recall that $\dot{u}(t) \in \mathcal{H}^{p+1}(\mathbb{T}^d)$. Lemma 3.3 implies that

$$\|\dot{u}(t) - R_h\dot{u}(t)\| \leq C|\dot{u}(t) - S_{h,p}\dot{u}(t)|_1.$$

Together with the fact

$$\begin{aligned}\|\omega(0)\| &= \|S_{h,p}u_0 - R_hu_0\| \\ &\leq \|S_{h,p}u_0 - u_0\| + \|R_hu_0 - u_0\| \\ &\leq \|S_{h,p}u_0 - u_0\| + C|S_{h,p}u_0 - u_0|_1\end{aligned} \qquad (3.1.33)$$

and

$$\|\eta_h(t) - \eta(t)\| \leq \|\eta(t) - S_{h,p}\eta(t)\|,$$

inequality (3.1.32) implies the result of lemma.

□

Now that estimates of $\|\rho(t)\|$ and $\|\omega(t)\|$ are obtained, we have the following theorem. Recall the operator $\mathcal{Q}_{0,p}(f)$ and $\mathcal{Q}_{1,p}(f)$ are defined in (2.4.6). We assume that the $p = 2r+1$, $r \in \mathbb{N}_0$ in the following theorem, because its proof requires the result in Theorem 2.14, which require the spline to be odd degree.

Theorem 3.5. *Let $u(t)$ and $u_h(t)$ be solution of the equation* (3.1.1) *and the semi-discrete scheme* (3.1.16) *respectively. For $p = 2r+1$, $r \in \mathbb{N}_0$, we have*

$$\begin{aligned}\|u_h(t) - u(t)\| &\leq C(\mathcal{Q}_{0,p}(u_0) + \mathcal{Q}_{1,p}(u_0) + \mathcal{Q}_{1,p}(u(t)) \\ &\quad + \textstyle\int_0^t \mathcal{Q}_{1,p}(\dot{u}(\tau))d\tau + \int_0^t \mathcal{Q}_{0,p}(\eta(\tau))d\tau).\end{aligned}$$

Proof. The result follows from applying the estimates in (2.4.7) to Lemma 3.3 and Lemma 3.4 which gives estimates of $\|\omega(t)\|$ and $\|\rho(t)\|$, and then substituting the estimates into

$$\|u_h(t) - u(t)\| \leq \|\omega(t)\| + \|\rho(t)\|.$$

□

The following corollary shows that under certain assumptions about the heat equation, the error converges with the degree of spline increasing.

Corollary 3.6. *Let $u(t)$ and $u_h(t)$ be solutions of the equation* (3.1.1) *and the semi-discrete scheme* (3.1.16) *respectively. In the equation* (3.1.1)*, assume $u_0 \in C^\infty(\mathbb{T}^d)$,$\eta(t) \in L_2(0,T;C^\infty(\mathbb{T}^d))$, and there exists $\sigma > 0$ s.t. $\|D^{\vec{\alpha}}u_0\| \le \sigma^{|\vec{\alpha}|}\|u_0\|$ and $\|D^{\vec{\alpha}}\eta(t)\| \le \sigma^{|\vec{\alpha}|}\|\eta(t)\|$ for any $\vec{\alpha} \in \mathbb{N}_0^d$, and $h\sigma/\pi < 1$. For $p = 2r+1$, $r \in \mathbb{N}_0$, we have*

$$\|u_h(t) - u(t)\| \le C\left(\frac{h\sigma}{\pi}\right)^p,$$

where the constant C is independent of p and h, but may be dependent on σ and t.

Proof. We first show that under these assumptions, we have

$$\|D^{\vec{\alpha}}u(t)\| \le C(t)\sigma^{|\vec{\alpha}|} \text{ and } \|D^{\vec{\alpha}}\dot{u}(t)\| \le C(t)\sigma^{|\vec{\alpha}|}, \tag{3.1.34}$$

where $C(t) \in L_2([0,T])$ is independent of $\vec{\alpha}$. For simplicity, by abuse of notation, $C(t)$ does not necessarily have the same value in different occasions. On both side of the equation (3.1.1), by taking the derivative $D^{\vec{\alpha}}$, multiplying $D^{\vec{\alpha}}u(t)$ and integrating over $\mathbb{T}^d$, we have

$$\int_{\mathbb{T}^d} D^{\vec{\alpha}}\dot{u}(t)D^{\vec{\alpha}}u(t)\mathrm{d}\boldsymbol{x} - \int_{\mathbb{T}^d} D^{\vec{\alpha}}\Delta u(t)D^{\vec{\alpha}}u(t)\mathrm{d}\boldsymbol{x} = \int_{\mathbb{T}^d} D^{\vec{\alpha}}\eta(t)D^{\vec{\alpha}}u(t)\mathrm{d}\boldsymbol{x}. \tag{3.1.35}$$

From the Green's Theorem, the second term in the left hand side of the equality is

$$\int_{\mathbb{T}^d}(\Delta D^{\vec{\alpha}}u(t))D^{\vec{\alpha}}u(t)\mathrm{d}\boldsymbol{x} = -\int_{\mathbb{T}^d}(\nabla D^{\vec{\alpha}}u(t))^2\mathrm{d}\boldsymbol{x} + \int_{\partial\mathbb{T}^d}(D^{\vec{\alpha}}u(t))\nabla(D^{\vec{\alpha}}u(t)\cdot\vec{n})\mathrm{d}\boldsymbol{x},$$

where the boundary integral is actually 0 because $D^{\vec{\alpha}}u$ is periodic for all $\vec{\alpha} \le p+1$. Substituting the term into (3.1.35), we have that

$$\frac{1}{2}\frac{d}{dt}\|D^{\vec{\alpha}}u(t)\|^2 + \sum_{j=1}^{d}\|\partial_{x_j}D^{\vec{\alpha}}u(t)\|^2 = \int_{\mathbb{T}^d} D^{\vec{\alpha}}\eta(t)D^{\vec{\alpha}}u(t)\mathrm{d}\boldsymbol{x}.$$

It follows that

$$\|D^{\vec{\alpha}}u(t)\|\frac{d}{dt}\|D^{\vec{\alpha}}u(t)\| \leq \int_{\mathbb{T}^d} D^{\vec{\alpha}}\eta(t)D^{\vec{\alpha}}u(t)\mathrm{d}\boldsymbol{x}.$$

Cauchy-Schwarz inequality together with the above inequality gives

$$\frac{d}{dt}\|D^{\vec{\alpha}}u(t)\| \leq \|D^{\vec{\alpha}}\eta(t)\|.$$

Finally, integrating on both side from 0 to t, we have

$$\begin{aligned}\|D^{\vec{\alpha}}u(t)\| &\leq \|D^{\vec{\alpha}}u_0\| + \textstyle\int_0^t \|D^{\vec{\alpha}}\eta(\tau)\|d\tau\\ &\leq \sigma^{|\vec{\alpha}|}(\|u_0\| + \textstyle\int_0^t \|\eta(\tau)\|d\tau)\\ &\leq C(t)\sigma^{|\vec{\alpha}|}.\end{aligned}$$

As for $\|D^{\vec{\alpha}}\dot{u}(t)\|$, taking the derivative $D^{\vec{\alpha}}$ and L_2 norm on both side of (3.1.1) gives

$$\|D^{\vec{\alpha}}\dot{u}(t) - D^{\vec{\alpha}}\Delta u(t)\| = \|D^{\vec{\alpha}}\eta(t)\|.$$

Applying the triangle inequality leads to

$$\begin{aligned}\|D^{\vec{\alpha}}\dot{u}(t)\| &\leq \|D^{\vec{\alpha}}\Delta u(t)\| + \|D^{\vec{\alpha}}\eta(t)\|\\ &\leq d\sigma^{|\vec{\alpha}|+2}(\|u_0\| + \textstyle\int_0^t \|\eta(\tau)\|d\tau) + \sigma^{|\vec{\alpha}|}\|\eta(t)\|\\ &\leq C(t)\sigma^{|\vec{\alpha}|}.\end{aligned}$$

Theorem 3.5 together with the estimate (2.4.8) and (3.1.34) completes the proof.

□

3.1.5 Stability of the One-dimensional Scheme

To study the convergence of the full scheme (3.1.25), we are interested in the consistency and stability of the time stepping scheme. Since the θ-method is consistent (see [88, p. 53]), we only need to consider stability, that is, when δt is fixed does vector $\vec{\alpha}^k$ remain bounded as $k \to \infty$? The vector $\vec{\alpha}^k$ is said to be bounded if its maximum norm, $\|\vec{\alpha}_i^k\|_\infty$, is bounded as $k \to \infty$.

3.1.5.1 Stability Condition

The object is to find out in the scheme (3.1.14) whether $\vec{\alpha}^k$ remains bounded as $k \to \infty$. Recall that $\vec{\alpha} = \boldsymbol{F}\widehat{\alpha}$. So $\vec{\alpha}^k$ is bounded if $\widehat{\alpha}^k$ is bounded. Let $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_M^{-1}\boldsymbol{\Lambda}_S$, $\lambda_{i;p} = \boldsymbol{\Lambda}_{i,i}$ and $(\lambda_G)_i = (\boldsymbol{\Lambda}_G)_{i,i}$, where we have the sub-index p in $\lambda_{i;p}$ because $\boldsymbol{\Lambda}$ corresponding to spline of degree p. The scheme (3.1.14) may be rewritten as a series of linear recurrence relations, for each $i = 0, \dots, N-1$,

$$\widehat{\alpha}_i^{k+1} = \frac{(1-\theta\delta t\lambda_{i;p})}{(1+(1-\theta)\delta t\lambda_{i;p})}\widehat{\alpha}_i^k + \frac{\delta t(\theta\widehat{\eta}_i^k + (1-\theta)\widehat{\eta}_i^{k+1})}{(1+(1-\theta)\delta t\lambda_{i;p})(\lambda_G)_i}.$$

Their characteristic polynomials show that it's sufficient to ensure that they are bounded as $k \to \infty$ by imposing the condition

$$\left|\frac{1-\theta\delta t\lambda_{i;p}}{1+(1-\theta)\delta t\lambda_{i;p}}\right| \le 1.$$

This is equivalent to

$$\begin{cases} 0 \le \delta t\lambda_{i;p} \le \dfrac{2}{2\theta-1}, & \dfrac{1}{2} < \theta \le 1 \\ 0 \le \delta t\lambda_{i;p}, & \text{otherwise} \end{cases} \quad \text{for } i \in \{0,1,\dots,N-1\}. \tag{3.1.36}$$

We see that the condition is satisfied if δt is adjusted to be small enough. We are interested in how large δt is allowed to remain the condition satisfied. This requires information about the largest value of $\lambda_{i;p}$, which is the object of the next section.

3.1.5.2 Markov-type Inequality for Periodic Spline

The definition of $\boldsymbol{\Lambda}$ together with (3.1.12) gives

$$\boldsymbol{S}\boldsymbol{F} = \boldsymbol{M}\boldsymbol{F}\boldsymbol{\Lambda}.$$

This equation shows $\lambda_{i;p}$ is the eigenvalue of the generalised eigen-problem

$$\boldsymbol{S}\vec{c} = \lambda\boldsymbol{M}\vec{c}, \tag{3.1.37}$$

with the corresponding eigenvector to be the ith column of the Fourier matrix $\boldsymbol{F}$ as defined in (1.3.1). According to the Min-Max Theorem (Page 1091, [48]), the largest eigenvalue $\lambda_{\max;p}$ of the problem is given by

$$\lambda_{\max;p} = \max_{\vec{c} \in \mathbb{R}^N} \frac{\vec{c}^T \boldsymbol{S} \vec{c}}{\vec{c}^T \boldsymbol{M} \vec{c}},$$

where we have the sub-index p in $\lambda_{\max;p}$ because the mass and stiffness matrices are corresponding to the spline with the degree p. For any $\vec{c} \in \mathbb{R}^N$, letting $v = \sum\limits_{i=0}^{N-1} c_i b_{i,p}$, then

$$\begin{aligned} \vec{c}^T \boldsymbol{M} \vec{c} &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \vec{c}_i (b_{i,p}, b_{j,p}) \vec{c}_j \\ &= \Big(\sum_{i=0}^{N-1} \vec{c}_i b_{i,p}, \sum_{j=0}^{N-1} \vec{c}_j b_{j,p} \Big) \\ &= \|v\|^2. \end{aligned}$$

Similarly, we have

$$\vec{c}^T \boldsymbol{S} \vec{c} = \|v'\|^2.$$

It follows that

$$\lambda_{\max;p} = \max_{v \in X^{N,p}(\mathbb{T})} \frac{\|v'\|^2}{\|v\|^2}. \tag{3.1.38}$$

An inequality useful to estimate the right hand side of (3.1.38) is the Markov inequality (see Proposition 3.4.2 [53]), stated as, if $f \in X^{N,n}(\mathbb{T})$ then

$$\frac{\|f^{(k)}\|_\infty}{\|f\|_\infty} \leq \frac{K_{n-k}}{K_n} \left(\frac{\pi}{h} \right)^k. \tag{3.1.39}$$

The norm $\|\cdot\|_\infty$ is the supremum norm and

$$K_m = \frac{4}{\pi} \sum_{k=0}^{\infty} (-1)^{k(m+1)} (2k+1)^{-(m+1)}, m \in \mathbb{N}_0,$$

is a Favard constant. It is infeasible to bound $\lambda_{\max;p}$ in (3.1.38) directly using the Markov inequality (3.1.39), due to their different norms. Therefore, we first derive a Markov inequality in the L_2 norm with the help of (3.1.39).

Theorem 3.7. *Any spline $v \in X^{N,n}(\mathbb{T})$, $n \in \mathbb{N}_0$, satisfies the inequality*

$$\max_{v \in X^{N,n}(\mathbb{T})} \frac{\|v^{(k)}\|}{\|v\|} \leq \sqrt{\frac{K_{2(n-k)+1}}{K_{2n+1}}} \frac{\pi^k}{h^k}.$$

Proof. For any $v \in X^{N,n}(\mathbb{T})$, let

$$w(x) = \int_0^1 v(x+t)v(t)dt.$$

We arrive at the result of the theorem by first showing that $w(x) \in X^{N,2n+1}(\mathbb{T})$ and then substituting it in the Markov inequality (3.1.39). Let $v(t) = \sum\limits_{i\in\mathcal{I}} c_i b_{i,p}(t)$. Using Property 4(iv) in Page 16, we have

$$\begin{aligned} w(x) &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} c_i c_j \int_0^1 b_{i,n}(x+t) b_{j,n}(t)dt \\ &= h\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} c_i c_j b_{i-j-(n+1),2n+1}(x). \end{aligned}$$

This implies $w(x) \in X^{N,2n+1}(\mathbb{T})$, since the expression is a linear combination of basis element in $X^{N,2n+1}(\mathbb{T})$,

The chain rule of differentiation gives

$$\begin{aligned} w'(x) &= \int_0^1 \tfrac{\partial}{\partial x} v(x+t)v(t)dt \\ &= \int_0^1 v'(x+t)v(t)dt \\ &= \int_0^1 \tfrac{\partial}{\partial t} v(x+t)v(t)dt. \end{aligned}$$

Then integration by parts and periodicity of v yields

$$w'(x) = -\int_0^1 v(x+t)v'(t)dt,$$

where boundary terms are cancelled out due to periodicity of $v(t)$. Repeatedly using the procedure, we obtain that

$$w^{(2k)}(x) = \int_0^1 \frac{\partial^k}{\partial t^k} v(x+t) \frac{d^k}{dt^k} v(t)dt, \text{ for } k = 0, \ldots, n,$$

where boundary terms are cancelled out due to periodicity of $v^{(j)}(t), j \in \{0, \ldots, n-1\}$.

Applying the Markov inequality to (3.1.39) gives

$$\begin{aligned} \|v^{(k)}\|^2 &= w^{(2k)}(0) \\ &\leq \|w^{(2k)}\|_\infty \\ &\leq K_{2(n-k)+1}\pi^{2k}/(K_{2n+1}h^{2k})\|w\|_\infty. \end{aligned} \tag{3.1.40}$$

Let $x^\star$ be such that $w(x^\star) = \|w\|_\infty$. The Cauchy-Schwarz inequality and periodicity of v imply

$$\begin{aligned} \|w\|_\infty &= \int_0^1 v(x^\star + t)v(t)dt. \\ &\le \|v(\cdot + x^\star)\|\|v\| \\ &= \|v\|^2. \end{aligned}$$

Taking this estimate into the inequality (3.1.40), the proof is completed. □

Theorem 3.7 together with the equality (3.1.38) implies that $\lambda_{\max;p}$ in eigenproblem (3.1.37) is bounded by

$$\lambda_{\max;p} \le \frac{K_{2p-1}}{K_{2p+1}} \frac{\pi^2}{h^2}. \tag{3.1.41}$$

We also would like to give a lower bound for the maximum eigenvalue. Ahead of that, we show an explicit expression for the eigenvalues. Assume that the eigenvectors in the problem (3.1.37) have the form $\vec{z}_j = \cos(\mu j)$. Substituting the expression into the eigen-problem leads to the following equalities, for $i = 1, \dots, N$,

$$\sum_{j=1}^N S_{i,j} \cos(\mu j) = \lambda \sum_{j=1}^N M_{i,j} \cos(\mu j). \tag{3.1.42}$$

We now study the left hand side of the equation. As shown in (3.1.9), certain entries of the matrix S are zero. Then it follows that

$$\sum_{j=1}^N S_{i,j} \cos(\mu j) = \sum_{r=-p}^p S_{i,[i+r]_N} \cos(\mu[i+r]_N),$$

where $[i]_N = i \mod N$. We assume further that $N\mu = 2k\pi$, which implies $\cos(\mu[i+r]_N) = \cos(\mu(i+r))$. Using the equality (3.1.10) and (3.1.11), and expanding the cosine term, we have

$$\begin{aligned} \sum_{j=1}^N S_{i,j} \cos(\mu j) &= S_{i,i} \cos(\mu i) + \sum_{r=1}^p S_{i,[i+r]_N} (\cos(\mu(i+r)) + \cos(\mu(i-r))) \\ &= S_{0,0} \cos(\mu i) + \sum_{r=1}^p S_{0,r} (\cos(\mu(i+r)) + \cos(\mu(i-r))) \\ &= S_{0,0} \cos(\mu i) + 2\cos(\mu i) \sum_{r=1}^p S_{0,r} \cos(\mu r). \end{aligned} \tag{3.1.43}$$

Similarly, the right hand side of equation (3.1.42) has the form

$$\lambda \sum_{j=1}^{N} M_{i,j} \cos(\mu j) = \lambda \left(M_{0,0} \cos(\mu i) + 2\cos(\mu i) \sum_{r=1}^{p} M_{0,r} \cos(\mu r) \right). \tag{3.1.44}$$

Substituting (3.1.43) and (3.1.44) into (3.1.42) and cancelling the term $\cos(\mu i)$ gives

$$S_{0,0} + 2\sum_{r=1}^{p} S_{0,r} \cos(\mu r) = \lambda \left(M_{0,0} + 2\sum_{r=1}^{p} M_{0,r} \cos(\mu r) \right).$$

We notice that the expression is independent of i, which means that λ remains unchanged for each equality in (3.1.42). Therefore $(\vec{z})_j = \cos(2k\pi j/N)$ is an eigenvector of the eigen-problem with the corresponding eigenvalue

$$\lambda_{k;p} = \frac{S_{0,0} + 2\sum\limits_{r=1}^{p} S_{0,r} \cos(2k\pi r/N)}{M_{0,0} + 2\sum\limits_{r=1}^{p} M_{0,r} \cos(2k\pi r/N)}.$$

Although it is not obvious from the expression to see which eigenvalue is the maximum one, by choosing suitable $\lambda_{k;p}$, we can give a lower bound for the maximum eigenvalue. We show in the following corollary that when N is even, by selecting $k = N/2$, the eigenvalue coincides with the upper bound of the maximum eigenvalues. In this case, the eigenvector is $c_i = (-1)^i$, and a spline with the coefficient is called a perfect spline [11, Ch. 6].

Corollary 3.8. *If the number of sub-intervals N is even, we have*

$$\max_{v \in X^{N,p}(\mathbb{T})} \frac{\|v^{(k)}\|}{\|v\|} = \sqrt{\frac{K_{2(p-k)+1}}{K_{2p+1}}} \frac{\pi^k}{h^k}.$$

Proof. Since the B-spline $b_{j,0}(x)$ is a function with value 1 on $[j, j+h]$ and 0 elsewhere, the perfect spline $u_0(x) = \sum\limits_{j=0}^{N-1} (-1)^j b_{j,0}(x)$ is a periodic function of period $2h$. Therefore $u_0(x)$ can be expressed by the Fourier series,

$$u_0(x) = \frac{4}{\pi} \sum_{k=0}^{\infty} \frac{1}{2k+1} \sin((2k+1)\pi x/h). \tag{3.1.45}$$

The expression (1.5.1) implies the relation $u_p(x) = \int_{x-h}^{x} u_{p-1}(t)dt$, which together with (3.1.45) gives

$$u_p(x) = \begin{cases} (-1)^{(p+1)/2} \frac{2^{p+2}}{\pi^{p+1}} \sum\limits_{k=0}^{\infty} \frac{1}{(2k+1)^{p+1}} \cos((2k+1)\pi x/h) & \text{if p is odd} \\ (-1)^{p/2} \frac{2^{p+2}}{\pi^{p+1}} \sum\limits_{k=0}^{\infty} \frac{1}{(2k+1)^{p+1}} \sin((2k+1)\pi x/h) & \text{if p is even} \end{cases}.$$

It follows that

$$u_p'(x) = \begin{cases} (-1)^{1+(p+1)/2} \frac{2^{p+2}}{\pi^{p}h} \sum\limits_{k=0}^{\infty} \frac{1}{(2k+1)^{p}} \sin((2k+1)\pi x/h) & \text{if p is odd} \\ (-1)^{p/2} \frac{2^{p+2}}{\pi^{p}h} \sum\limits_{k=0}^{\infty} \frac{1}{(2k+1)^{p}} \cos((2k+1)\pi x/h) & \text{if p is even} \end{cases}.$$

When p is odd, since $\int_0^{2h} \cos(i\pi x/h)\cos(j\pi x/h)\mathrm{d}x = \delta_{i,j}$, we have that

$$\begin{aligned} \|u_p\|^2 = & \int_0^b u_p^2(x)\mathrm{d}x \\ = & \frac{N}{2}\frac{2^{2p+4}}{\pi^{2p+2}} \int_0^{2h} (\sum_{k=0}^{\infty} \frac{1}{(2k+1)^{p+1}} \cos^2((2k+1)\pi x/h))\mathrm{d}x \\ = & \frac{N}{2}\frac{2^{2p+4}}{\pi^{2p+2}} \int_0^{2h} \sum_{k=0}^{\infty} \frac{1}{(2k+1)^{2p+2}} \cos^2((2k+1)\pi x/h)\mathrm{d}x \\ = & \frac{N}{2}\frac{2^{2p+4}}{\pi^{2p+2}} \sum_{k=0}^{\infty} \frac{1}{(2k+1)^{2p+2}} \int_0^{2h} \cos^2((2k+1)\pi x/h)\mathrm{d}x \\ = & \frac{N}{2}\frac{2^{2p+4}}{\pi^{2p+2}} \sum_{k=0}^{\infty} \frac{1}{(2k+1)^{2p+2}}. \end{aligned} \tag{3.1.46}$$

Similarly, we have the same expression for the case where p is even. For the same reason, we also have

$$\|u_p'\|^2 = \frac{N}{2}\frac{2^{2p+4}}{\pi^{2p}h^2} \sum_{k=0}^{\infty} \frac{1}{(2k+1)^{2p}}. \tag{3.1.47}$$

Then together with (3.1.46) and (3.1.47), (3.1.38) implies that

$$\begin{aligned} \lambda_{\max;p} \geq & \frac{\|u_p'\|^2}{\|u_p\|^2} \\ = & \frac{\pi^2}{h^2}\frac{K_{2p-1}}{K_{2p+1}}. \end{aligned}$$

Together with Theorem 3.7, the result of the corollary follows. □

The corollary doesn't hold for the case when N is odd. That is because the eigenvector has the expression $\vec{z}_j = \cos(2\pi kj/N)$, and when N is odd the coefficients of the perfect spline do not satisfy the required expression. We conjecture that when N is odd, the eigenvector $\vec{z}_j = \cos((N-1)\pi j/N)$ corresponds to the maximum eigenvalue given by

$$\lambda_{\max;p} = \frac{S_{0,0} + 2\sum_{r=1}^{p} S_{0,r}\cos((N-1)\pi r/N)}{M_{0,0} + 2\sum_{r=1}^{p} M_{0,r}\cos((N-1)\pi r/N)}. \tag{3.1.48}$$

Figure 3.1 shows numerical results that agree with the conjecture.

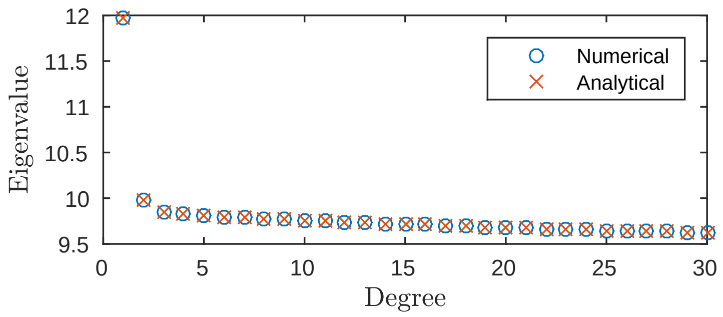


Figure 3.1: When N=61. The comparison between eigenvalues computed numerically and from expression (3.1.48).

Recall that

$$\lambda_{\max;n} = \max_{v\in X^{N,n}(\mathbb{T})} \frac{\|v'\|}{\|v\|}.$$

We show that the constant $\lambda_{\max;s}$ doesn't increase as the degree s increasing, which is given in the following lemma.

Lemma 3.9. *For all $s < n$,*

$$\lambda_{\max;n} \le \lambda_{\max;s}.$$

Proof. For each $f \in X^{N,n}(\mathbb{T})$, we have $f^{(n-s)} \in X^{N,s}(\mathbb{T})$. It follows that

$$\frac{\|f^{(n-s+1)}\|}{\|f^{(n-s)}\|} \le \max_{v\in X^{N,s}(\mathbb{T})} \frac{\|v'\|}{\|v\|} = \lambda_{\max;s}. \tag{3.1.49}$$

Since $f \in H^{n-s+1}(\mathbb{T})$, using the Kolmogorov inequality in (2.3.10), which is for any $g \in H^{n-s+1}(\mathbb{T})$,

$$\|g^{(k)}\| \leq \|g\|^{1-k/(n-s+1)} \|g^{(n-s+1)}\|^{k/(n-s+1)}, \text{ for all } k \leq n-s+1, \tag{3.1.50}$$

we have that

$$\|f^{(n-s)}\| \leq \|f\|^{1-\frac{n-s}{n-s+1}} \|f^{(n-s+1)}\|^{\frac{n-s}{n-s+1}}. \tag{3.1.51}$$

This inequality together with (3.1.49) gives

$$\|f^{(n-s+1)}\| \leq \lambda_{\max;s}^{n-s+1} \|f\|.$$

Substituting the inequality into the Kolmogorov inequality (3.1.51), we have

$$\|f'\| \leq \lambda_{\max;s} \|f\|.$$

Since this inequality holds for all $f \in X^{N,n}(\mathbb{T})$, then

$$\lambda_{\max;n} = \max_{f \in X^{N,n}(\mathbb{T})} \frac{\|f'\|}{\|f\|} \leq \lambda_{\max;s}.$$

the result of the lemma follows. □

The condition (3.1.36) together with (3.1.41) shows that when $\theta \in (0.5, 1]$, the time stepping scheme (3.1.14) is stable if

$$\delta t \leq \frac{2K_{2p+1}h^2}{(2\theta-1)K_{2p-1}\pi^2}. \tag{3.1.52}$$

Corollary 3.8, show that when N is even, the condition is sharp. Since the mass and stiffness matrices are positive definite, $\lambda_{\max;p}$ is always non-negative. So when $\theta \in [0, 0.5]$, the scheme is always stable.

3.1.6 Stability of the Multi-dimensional Scheme

Similar to the stability analysis in Section 3.1.5, we let $\mathbf{\Gamma} = \mathbf{\Lambda}_{\mathcal{M}}^{-1}\mathbf{\Lambda}_{\mathcal{S}}$, $\gamma_{i;p} = (\mathbf{\Gamma})_{i,i}, i = 0, \ldots, N^d - 1$ and $(\lambda_{\mathcal{G}})_i = (\mathbf{\Lambda}_{\mathcal{G}})_{i,i}$. Since all matrices in the scheme

(3.1.25) are diagonal $\widehat{\alpha}^0 = \boldsymbol{\mathcal{F}}^{-1}\vec{\alpha}^0$, stability of the scheme is equivalent to that of the following recurrence relations, for each $i = 0, \dots, N^d - 1$,

$$\widehat{\alpha}_i^{k+1} = \frac{(1-\theta\delta t\gamma_{i;p})}{(1+(1-\theta)\delta t\gamma_{i;p})}\widehat{\alpha}_i^k + \frac{\delta t(\theta\widehat{\eta}_i^k + (1-\theta)\widehat{\eta}_i^{k+1})}{(1+(1-\theta)\delta t\gamma_{i;p})(\lambda_{\mathcal{G}})_i}.$$

To ensure these relations remain bounded as $k \to \infty$, the following condition is sufficient

$$\begin{cases} 0 \le \delta t\gamma_{i;p} \le \dfrac{2}{2\theta-1}, & \dfrac{1}{2} < \theta \le 1 \\ 0 \le \delta t\gamma_{i;p}, & \text{else} \end{cases}, \text{ for } i = 0, \dots, N^d - 1. \tag{3.1.53}$$

The maximum value of $\gamma_{i;p}$ needs to be investigated to understand how small δt is enough for the scheme to be stable.

The definition of $\boldsymbol{\Lambda}_{\mathcal{M}}$, $\boldsymbol{\Lambda}_{\mathcal{S}}$ and Property 2(iv) in Page 8 gives

$$\begin{aligned} \boldsymbol{\Gamma} &= \boldsymbol{\Lambda}_{\mathcal{M}}^{-1}\boldsymbol{\Lambda}_{\mathcal{S}} \\ &= \sum_{s=1}^{d} \underbrace{\boldsymbol{I} \otimes \dots \otimes \boldsymbol{I} \otimes \boldsymbol{\Lambda}}_{s} \otimes \boldsymbol{I} \otimes \dots \otimes \boldsymbol{I} \\ &= \underbrace{\boldsymbol{\Lambda} \oplus \boldsymbol{\Lambda} \oplus \dots \oplus \boldsymbol{\Lambda}}_{d}, \end{aligned}$$

where $\oplus$ the is the Kronecker sum as introduced in Property 2(iv) in Page 8. The expression together with Property 2(iv) in Page 8 reveals that the largest value of $\{\gamma_{i;p}\}$ is

$$\gamma_{\max;p} = d\lambda_{\max;p} \le d\frac{K_{2p-1}}{K_{2p+1}}\frac{\pi^2}{h^2}.$$

Thus, from the inequality (3.1.53), when $\theta \in (0.5, 1]$, the time stepping scheme (3.1.25) is stable if

$$\delta t \le \frac{2}{(2\theta-1)d\lambda_{\max;p}} \le \frac{2K_{2p+1}h^2}{(2\theta-1)dK_{2p-1}\pi^2}. \tag{3.1.54}$$

To summarize,

$$\begin{cases} \theta \in [0, 0.5] & \text{always stable} \\ \theta \in (0.5, 1] & \text{stable if (3.1.54) holds} \end{cases}.$$

Since $\lambda_{\max;p}$ decrease w.r.t. p as shown in (3.9), if we increase the spline's degree, the condition implies that the scheme's stability gets better, that is, allowing a larger value of the mesh size for a fixed step size.

3.1.7 Error of the Full Scheme

We derive the error of the full scheme following the approach in [56, 39]. The full scheme is for $k = 0, \dots, m-1$,

$$\left(\frac{U^{k+1}-U^k}{\delta t}, \chi\right) + a((1-\theta)U^{k+1} + \theta U^k, \chi) = (\eta_h(t_k), \chi) \text{ for any } \chi \in X^{N,p}(\mathbb{T}^d). \tag{3.1.55}$$

We see that U^m is an approximation of the true solution at time T. We are interested in the error of the approximation, which is $\|U^m - u(T)\|$. Analogous to the analysis of the semi-discrete error, we divide the error into two parts as

$$U^m - u(T) = \omega^m + \rho^m, \tag{3.1.56}$$

and then estimate them separately, where $\omega^k = U^k - R_h u(t_k)$ and $\rho^k = R_h u(t_k) - u(t_k)$. Lemma 3.3 directly leads to an estimate of $\|\rho^m\|$,

$$\|\rho^m\| \le C|S_{h,p}u(T) - u(T)|_1.$$

An estimate of ω^m is given by the following lemma.

Lemma 3.10. *Let* $\theta \le 1/2$ *in the scheme* (3.1.25). *We have that*

$$\begin{aligned}\|\omega^m\| \quad &\le C(\|S_{h,p}u_0 - u_0\| + |S_{h,p}u_0 - u_0|_1 \\ &\quad + \textstyle\int_0^T |S_{h,p}\dot{u}(s) - \dot{u}(s)|_1 \mathrm{d}s + \delta t \int_0^T \|\ddot{u}(s)\|\mathrm{d}s \\ &\quad + \delta t \sum_{k=0}^{m-1} \|S_{h,p}\eta(t_k) - \eta(t_k)\|).\end{aligned}$$

Proof. The full scheme (3.1.55) can be rewritten as, for any $\chi \in X^{N,p}(\mathbb{T})$,

$$\left(\frac{U^{k+1}-U^k}{\delta t}, \chi\right) + a((1-\theta)U^{k+1} + \theta U^k, \chi) = (\eta(t_k), \chi) + (\eta_h(t_k) - \eta(t_k), \chi).$$

Together with the equation (3.1.26) and (3.1.2), it follows that

$$\left(\frac{\omega^{k+1}-\omega^k}{\delta t}, \chi\right) + a\left((1-\theta)\omega^{k+1} + \theta\omega^k, \chi\right) = (\psi^k, \chi), \tag{3.1.57}$$

where

$$\psi^k = \frac{R_h u(t_{k+1}) - R_h u(t_k)}{\delta t} - ((1-\theta)\dot{u}(t_{k+1}) + \theta \dot{u}(t_k)) + \eta_h(t_k) - \eta(t_k).$$

Let

$$\psi_1^k = \frac{R_h u(t_{k+1}) - R_h u(t_k)}{\delta t} - \frac{u(t_{k+1}) - u(t_k)}{\delta t},$$

and

$$\psi_2^k = \frac{u(t_{k+1}) - u(t_k)}{\delta t} - ((1-\theta)\dot{u}(t_{k+1}) + \theta \dot{u}(t_k)).$$

It follows that $\psi^k = \psi_1^k + \psi_2^k + \eta_h(t_k) - \eta(t_k)$.

Choosing $\chi = (1-\theta)\omega^{k+1} + \theta\omega^k$ in (3.1.57) and using the Cauchy-Schwarz inequality, we have

$$\frac{1}{\delta t}\left(\omega^{k+1} - \omega^k, (1-\theta)\omega^{k+1} + \theta\omega^k\right) \le \|\psi^k\| \|\theta\omega^{k+1} + (1-\theta)\omega^k\|.$$

On the other hand, rearranging the inner product and using the Cauchy-Schwarz inequality gives

$$\begin{aligned}\left(\omega^{k+1} - \omega^k, (1-\theta)\omega^{k+1} + \theta\omega^k\right) &\ge (1-\theta)\|\omega^{k+1}\|^2 + (1-2\theta)\|\omega^k\|\|\omega^{k+1}\| - \theta\|\omega^k\| \\ &= (\|\omega^{k+1}\| - \|\omega^k\|)((1-\theta)\|\omega^{k+1}\| + \theta\|\omega^k\|).\end{aligned}$$

Combining the two inequalities yields

$$\|\omega^{k+1}\| - \|\omega^k\| \le \delta t \|\psi^k\|,$$

that is,

$$\|\omega^{k+1}\| \le \|\omega^k\| + \delta t \|\psi^k\|.$$

By repeated application, the inequality implies

$$\begin{aligned}\|\omega^m\| &\le \|\omega^0\| + \delta t \sum_{k=0}^{m-1} \|\psi^k\| \\ &\le \|\omega^0\| + \delta t \sum_{k=0}^{m-1} \|\psi_1^k\| \\ &\quad + \delta t \sum_{k=0}^{m-1} \|\psi_2^k\| + \delta t \sum_{k=0}^{m-1} \|\eta_h(t_k) - \eta(t_k)\|.\end{aligned}$$

We now seek to estimate the four summands in the right hand side separately. Using the same estimate in (3.1.33) leads to

$$\|\omega^0\| \leq \|S_{h,p}u_0 - u_0\| + C|S_{h,p}u_0 - u_0|_1. \tag{3.1.58}$$

Since the Ritz operator R_h in (3.1.26) and temporal derivative commute, we have

$$\psi_1^k = \frac{1}{\delta t}\int_{t_k}^{t_{k+1}} R_h\dot{u}(s)\mathrm{d}s - \frac{1}{\delta t}\int_{t_k}^{t_{k+1}} \dot{u}(s)\mathrm{d}s,$$

and

$$\begin{aligned} \delta t\sum_{k=0}^{m-1}\|\psi_1^k\| &\leq \sum_{k=0}^{m-1}\int_{t_k}^{t_{k+1}}\|R_h\dot{u}(s)-\dot{u}(s)\|\mathrm{d}s \\ &= \int_0^T \|R_h\dot{u}(s)-\dot{u}(s)\|\mathrm{d}s \\ &\leq C\int_0^T |S_{h,p}\dot{u}(s)-\dot{u}(s)|_1\mathrm{d}s. \end{aligned} \tag{3.1.59}$$

As for ψ_2^k, Taylor expansion gives

$$\begin{aligned} \delta t\psi_2^k &= (1-\theta)(u(t_{k+1}) - u(t_k) - \delta t\dot{u}(t_{k+1})) + \theta(u(t_{k+1}) - u^k - \delta t\dot{u}(t_k)) \\ &= -(1-\theta)\int_{t_k}^{t_{k+1}}(s-t_k)\ddot{u}(s)\mathrm{d}s + \theta\int_{t_k}^{t_{k+1}}(t_{k+1}-s)\ddot{u}(s)\mathrm{d}s \\ &= \int_{t_k}^{t_{k+1}}(t_{k+1}-(1-\theta)\delta t - s)\ddot{u}(s)\mathrm{d}s, \end{aligned}$$

and it follows that

$$\begin{aligned} \delta t\sum_{k=0}^{m-1}\|\psi_2^k\| &\leq \sum_{k=0}^{m-1}\int_{t_k}^{t_{k+1}}\|(t_{k+1}-(1-\theta)\delta t - s)\ddot{u}(s)\|\mathrm{d}s \\ &\leq \delta t\int_0^T\|\ddot{u}(s)\|\mathrm{d}s, \end{aligned} \tag{3.1.60}$$

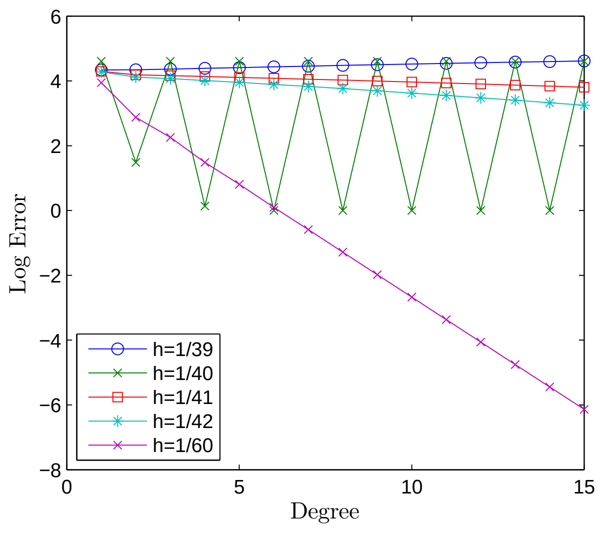
since $|t_{k+1} - (1-\theta)\delta t - s| \leq \delta t$ for $s \in (t_k, t_{k+1})$.

Since $\eta_h(t) = S_{h,p}\eta(t)$, by summing up the estimates (3.1.58), (3.1.59), and (3.1.60), we complete the proof. □

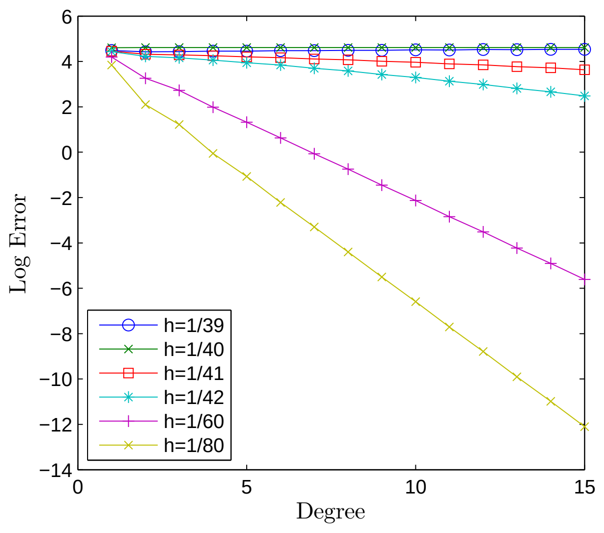
Then the following theorem gives an estimation of the error in the full scheme.

Theorem 3.11. *Let $u(T)$ and U^m be the solution of the equation* (3.1.1) *and the scheme* (3.1.25) *at time T respectively. Assuming that $u_0(t)$ and $\eta(t)$ satisfy the assumption of Corollary 3.6 and $\theta \leq 1/2$ in* (3.1.25), *we have*

$$\|u(T) - U^m\| \leq C\left(\left(\frac{h\sigma}{\pi}\right)^p + \delta t\right).$$



(a) Scheme in $\mathbb{R}$



(b) Scheme in $\mathbb{R}^2$

Figure 3.2: The error of semi-discrete solution.

Proof. The proof follows from the equation (3.1.56), Lemma 3.3, and Lemma 3.10. Specially for the term $\delta t \sum_{k=0}^{m-1} \|S_{h,p}\eta(t_k) - \eta(t_k)\|$ in Lemma 3.10, Corollary 2.15 and the error of Riemann sum [54, Chapter 4], imply that

$$\begin{aligned}\delta t \sum_{k=0}^{m-1} \|S_{h,p}\eta(t_k) - \eta(t_k)\| &\leq C(\tfrac{h\sigma}{\pi})^{p+1}\delta t \sum_{k=0}^{m-1} \|\eta(t_k)\| \\ &= C(\tfrac{h\sigma}{\pi})^{p+1}(\int_0^T \|\eta(s)\|\mathrm{d}s + \mathcal{O}(\delta t)) \\ &\leq C(\tfrac{h\sigma}{\pi})^{p+1} \int_0^T \|\eta(s)\|\mathrm{d}s + C(\tfrac{h\sigma}{\pi})^{p+1}\delta t.\end{aligned}$$

All the other terms are estimated by Corollary 2.15 and (3.1.34). By summing up the estimates, the proof is completed.

□

3.1.8 Numerical Experiment Results

3.1.8.1 Semi-discrete Error

Our first experiment concerns the difference between the solution of the heat equation (3.1.1) and the semi-discrete approximation, which is the solution of (3.1.16).

We are interested in the convergence when raising the spline degree and also in how the mesh size affects the convergence.

We consider the heat equation in one-dimensional and two dimensional cases, where the true solutions are set to be $u = \sin(40\pi x + t)$ and $u = \sin(40\pi x + 40\pi y + t)$, respectively. Since the ODE system (3.1.24) is equivalent to a series of scalar ODEs, and in our case, there exist an analytical expression for each ODE, we obtain the semi-discrete solution. We calculate the error $E_0 u$ for various degrees of B-spline p, and for different mesh sizes h, where E_0 is defined in (2.7.1). The result is presented by plotting the error against the B-spline degree for each h.

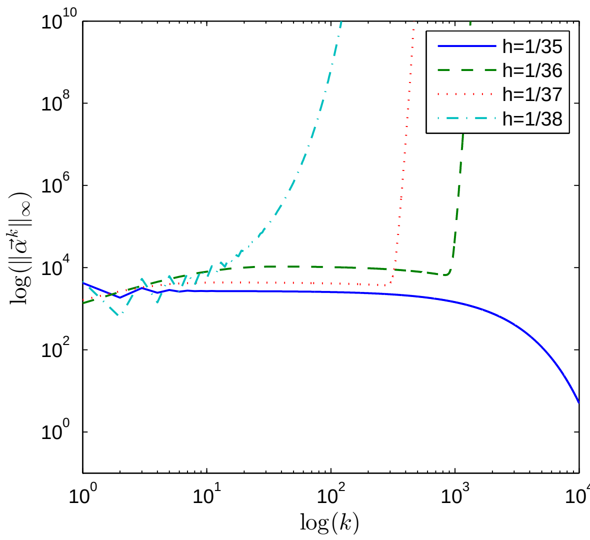
Figure 3.2a and 3.2b shows when $h < 1/40$, the error converges linearly. The behaviour of the error is as anticipated from the error analysis in Section 3.1.4. Corollary 3.6 implies that the error is bounded by $\mathcal{O}((40h)^p)$. Then as p increases, if $h < 1/40$, the error converges. The linear convergence rate implies the error converges exponentially since we take a logarithm transformation of the error. The oscillation for the case of $h = 1/40$ arises from the interpolation of the initial value and non-homogeneous term, which are explained in Section 2.7.0.1.

3.1.8.2 Stability

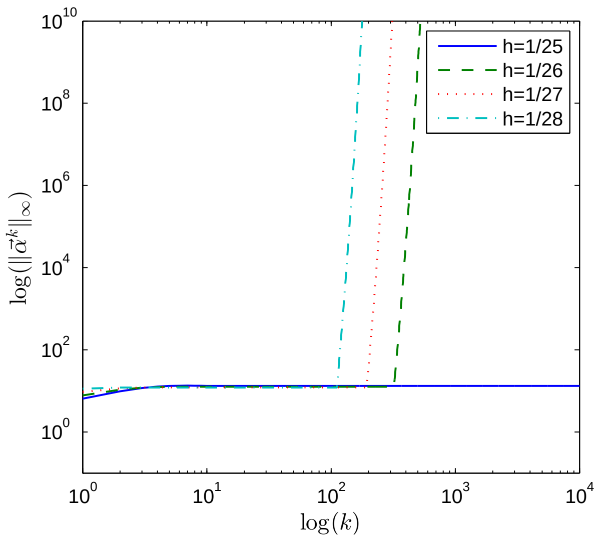
The next experiment tests the sharpness of the stability condition of the time-stepping scheme (3.1.25) for the problem in $\mathbb{R}$ and $\mathbb{R}^2$ respectively.

We implement Algorithm 3.2, and calculate the maximum norm of the coefficient $\vec{\alpha}^k$ for each time step k. The number of iterations are set large enough such that the trend of $\|\vec{\alpha}^k\|_\infty$ is obvious to spot. A figure of $log(k)$ against $log(\|\vec{\alpha}^k\|_\infty)$ for different h is depicted, where the value of $h = 1/N$ is adjusted by choosing consecutive numbers of N, and the threshold value of h in stability condition (3.1.52) lies between these values. In this way, we can check whether or not the stability condition is sharp.

Figure 3.3a shows the result for the one dimensional scheme when $\delta t = 0.0002$, $p = 4$ and $\theta = 0.9$. The solid line corresponding to $h = 1/35$ (approximately



(a) Scheme in $\mathbb{R}$



(b) Scheme in $\mathbb{R}^2$

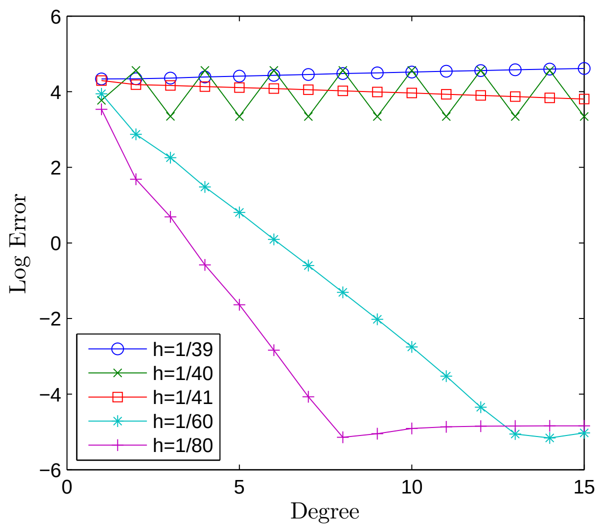
Figure 3.3: Stability of the θ-scheme

0.0286) converged, while the others did not. The result agrees with the stability analysis in Section 3.1.5, where the condition (3.1.52) implies if the mesh size h is larger than the threshold value 0.0281, the time-stepping is stable. The solid line shows a case that satisfies the condition, while the others do not.

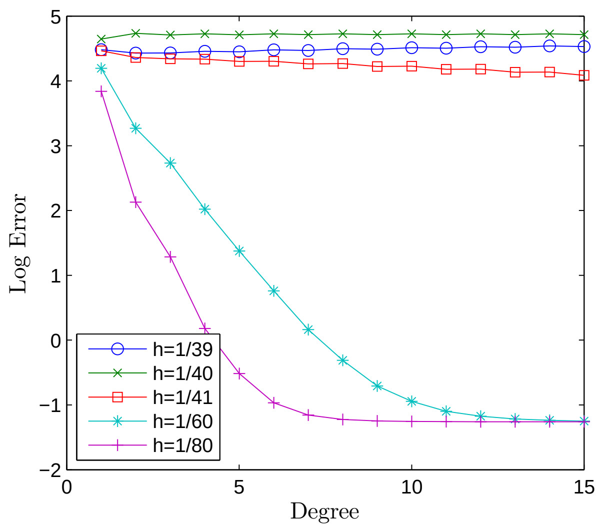
A similar result is shown in Figure 3.3b for the two-dimensional case in the same setting of parameters. Using the condition (3.1.54), we calculate that the scheme is stable if $h > 0.397$. Only the case shown in the solid line corresponding to the case $h = 0.04$ satisfies the condition. The results of these two tests agree with our analysis and suggest that the condition is sharp.

3.1.8.3 Error of Full Scheme

We now consider the full error of the same problem in Section 3.1.8.1. As shown in Figure 3.4a and 3.4b, as the degree increases, if $h < 1/40$, the error converges. When $h = 1/60$, the errors converge at first, then stop converging when the errors decrease to a certain level. The errors behave as we expected from Theorem 3.11.



(a) Scheme in $\mathbb{R}$



(b) Scheme in $\mathbb{R}^2$

Figure 3.4: Error of full scheme

The full scheme's error consists of the errors caused by both spatial discretization and temporal discretization. By increasing the degree of the spline, only the spatial part decreases. When the spatial part is negligible compared with the temporal one, the convergence of the full error stops because the temporal error dominates.

3.2 Problem with Dirichlet Boundary Condition

3.2.1 Model Problem

Let $\Omega = [0,1]^d$. The problem under consideration is the heat equation

$$\frac{\partial}{\partial t}u(\boldsymbol{x},t) - \Delta u(\boldsymbol{x},t) = \eta(\boldsymbol{x},t),\ (\boldsymbol{x},t) \in \Omega \times [0,\infty), \tag{3.2.1}$$

with initial value

$$u(\boldsymbol{x},0) = u_0(\boldsymbol{x}) \in \mathcal{H}_0^{p+1}(\Omega),$$

Dirichlet boundary condition

$$u(\boldsymbol{x},t) = 0 \text{ for } \boldsymbol{x} \in \partial\Omega,$$

and non-homogeneous term

$$\eta(t) \in L_2(0,T;\mathcal{H}_0^{p+1}(\Omega)).$$

A weak formulation of the problem is finding $u(t) \in L_2(0,T;\mathcal{H}_0^1(\Omega))$ with $\dot{u}(t) \in L_2(0,T;\mathcal{H}^{-1}(\Omega))$ and $u(0) = u_0 \in L_2(\Omega)$ such that

$$\frac{d}{dt}(u(t),v) + a(u(t),v) = (\eta(t),v) \qquad \text{for any } v \in \mathcal{H}_0^1(\Omega). \tag{3.2.2}$$

Analogous to the periodic case, because of Proposition 23.28 in [92], and the regularity of u_0 and η, we have $u(t) \in L_2(0,T;\mathcal{H}_0^{p+1}(\Omega))$.

3.2.2 B-spline Scheme

Let $\mathcal{J}_0 = \{0,1,\ldots,N+p-3\}$. Using the Galerkin method, we approximate the solution by finding $u_h(t) \in X_0^{N,p}(\Omega)$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}(u_h(t),v) + a(u_h(t),v) = (\eta(t),v) \qquad \text{for any } v \in X_0^{N,p}(\Omega). \tag{3.2.3}$$

Approximating $\eta(t)$ by $\mathcal{B}_{h,p}\eta(t)$, we have the matrix form,

$$\boldsymbol{\mathcal{M}}\dot{\vec{\alpha}}(t) + \boldsymbol{\mathcal{S}}\vec{\alpha}(t) = \boldsymbol{\mathcal{M}}\vec{l}(t). \tag{3.2.4}$$

Letting $i = \sigma_{N+p-2}(\boldsymbol{i})$ and $j = \sigma_{N+p-2}(\boldsymbol{j})$ for $\boldsymbol{i},\boldsymbol{j} \in \mathcal{J}_0^d$, the mass matrix $\boldsymbol{\mathcal{M}}$ and stiffness matrix $\boldsymbol{\mathcal{S}}$ have the entries

$$\boldsymbol{\mathcal{M}}_{ij} = \int_\Omega \psi_{\boldsymbol{i},p}(\boldsymbol{x})\psi_{\boldsymbol{j},p}(\boldsymbol{x})d\boldsymbol{x} \text{ and } \boldsymbol{\mathcal{S}}_{ij} = \int_\Omega \nabla\psi_{\boldsymbol{i},p}(\boldsymbol{x})\nabla\psi_{\boldsymbol{j},p}(\boldsymbol{x})d\boldsymbol{x}. \tag{3.2.5}$$

Similarly to the periodic case, the tensor product structure of basis $\psi_{\boldsymbol{i},p}(\boldsymbol{x})$ implies that

$$\boldsymbol{\mathcal{M}} = \underbrace{\boldsymbol{M} \otimes \boldsymbol{M} \otimes \cdots \otimes \boldsymbol{M}}_{d}, \tag{3.2.6}$$

and

$$\boldsymbol{\mathcal{S}} = \sum_{s=1}^{d} \underbrace{\boldsymbol{M} \otimes \cdots \otimes \boldsymbol{M} \otimes \boldsymbol{S}}_{s} \otimes \boldsymbol{M} \otimes \ldots \otimes \boldsymbol{M},$$

where

$$\boldsymbol{M}_{i,j} = \int_0^1 \psi_{i,p}(x)\psi_{j,p}(x)\mathrm{d}x \text{ and } \boldsymbol{S}_{i,j} = \int_0^1 \psi'_{i,p}(x)\psi'_{j,p}(x)\mathrm{d}x.$$

The expression (3.2.6) implies that we can evaluate $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{S}}$ from $\boldsymbol{M}$ and $\boldsymbol{S}$, respectively. Unlike the periodic cases, $\boldsymbol{M}$ and $\boldsymbol{S}$ are no longer circulant matrices, since the basis functions near the boundaries of Ω have different shapes to the interior ones. We do not have explicit expressions to compute the matrices, but since they are integrals of polynomials, quadrature rules can be applied to evaluate them exactly. One standard way in FEM is to apply a Gaussian quadrature rule to evaluate element matrices on each of the subintervals and then assembly to construct the global matrices. However, since the B-spline is of high smoothness on the whole domain, more efficient quadrature strategies exist by reducing the employed quadrature points [6, 47, 78]. That is because the quadrature is designed to give exact integrals for splines, and the high smoothness of splines reduce the number of conditions the quadrature rule satisfying. We use the quadrature rule described in [6].

We still apply the θ method to solve the ODE systems (3.2.4), which gives

$$(\boldsymbol{\mathcal{M}} + (1-\theta)\delta t\boldsymbol{\mathcal{S}})\vec{\alpha}^{k+1} = (\boldsymbol{\mathcal{M}} - \theta\delta t\boldsymbol{\mathcal{S}})\vec{\alpha}^{k} + \delta t\boldsymbol{\mathcal{M}}((1-\theta)\vec{l}^{k+1} + \theta\vec{l}^{k}), \quad (3.2.7)$$

where $\vec{l}^k$ is the coefficient of $\mathcal{B}_{h,p}\eta(t_k)$, and the initial value α_0 is the coefficient of $\mathcal{B}_{h,p}u_0$.

3.2.3 Error of the Semi-discrete Scheme

The error analysis is very similar to the periodic case in Section 3.1.4, and we only list the results and the different lines in the proof.

Let $R_h : \mathcal{H}_0^{p+1}(\Omega) \to X_0^{N,p}(\Omega)$ the operator such that

$$a\left(R_h f, v\right) = a\left(f, v\right) \text{ for any } v \in X_0^{N,p}(\Omega). \quad (3.2.8)$$

With the same idea of deriving (3.1.28), we have for every $f \in \mathcal{H}_0^{p+1}(\Omega)$,

$$|R_h f - f|_1 \leq |\mathcal{B}_{h,p} f - f|_1. \quad (3.2.9)$$

We split the error in the same form as (3.1.27). The following lemma provides a bound for $\|\rho(t)\|$.

Lemma 3.12. *For $n = 2r+1, r \in \mathbb{N}_0$, and $f \in \mathcal{H}^{p+1}(\Omega)$,*

$$\|R_h f - f\| \le C|\mathcal{B}_{h,p} f - f|_1.$$

Proof. Friedrich's inequality (Lemma 3.6 [39]) implies that for any $v \in H^1(\Omega)$, one has

$$\|v\|^2 \le C\left(|v|_1^2 + \left(\int_{\partial\Omega} v \mathrm{d}\boldsymbol{x}\right)^2\right).$$

Since $R_h f - f \in \mathcal{H}_0^p(\Omega) \subset H^1(\Omega)$, from the Friedrich's inequality we have that

$$\|R_h f - f\| \le C|R_h f - f|_1,$$

where the boundary integral term is 0 because $R_h f - f$ vanishes at boundaries. By substituting (3.2.9) into the inequality, the proof is completed. □

From Lemma 3.12, we have

$$\|\rho(t)\| \le C|\mathcal{B}_{h,p} f - f|_1.$$

A bound for $\|\omega(t)\|$ is given by the following lemma.

Lemma 3.13.

$$\begin{aligned}\|\omega(t)\| \;&\le C(\|u_0 - \mathcal{B}_{h,p}u_0\| + |u_0 - \mathcal{B}_{h,p}u_0|_1 \\ &\quad + \textstyle\int_0^t |\dot{u}(\tau) - \mathcal{B}_{h,p}\dot{u}(\tau)|_1 \mathrm{d}\tau + \int_0^t \|\eta(\tau) - \mathcal{B}_{h,p}\eta(\tau)\|\mathrm{d}\tau).\end{aligned}$$

Proof. The proof is similar to the proof of Lemma 3.4 with the following changes.

- Replace all the spaces $\mathcal{H}^{p+1}(\mathbb{T}^d)$ and $X^{N,p}(\mathbb{T}^d)$ by $\mathcal{H}_0^{p+1}(\Omega)$ and $X_0^{N,p}(\Omega)$, respectively.
- Replace all the operators $S_{h,p}$ by $\mathcal{B}_{h,p}$.

- Replace the reference of equations (3.1.15), (3.1.26), and (3.1.2) by (3.2.3), (3.2.8), and (3.2.2), respectively.

□

The following theorem gives an estimate for the semi-discrete scheme.

Theorem 3.14. *Let $u(t)$ and $u_h(t)$ be solutions of the equation* (3.2.1) *and the equation* (3.2.4) *respectively. For $p = 2r + 1$, $r \in \mathbb{N}_0$, we have*

$$\begin{aligned}\|u_h(t) - u(t)\| &\leq C(\mathcal{Q}_{0,p}(u_0) + \mathcal{Q}_{1,p}(u_0) + \mathcal{Q}_{1,p}(u(t)) \\ &\quad + \textstyle\int_0^t \mathcal{Q}_{1,p}(\dot{u}(\tau))d\tau + \int_0^t \mathcal{Q}_{0,p}(\eta(\tau))d\tau).\end{aligned}$$

Proof. By applying Theorem 2.19 together with (2.4.6) to Lemma 3.12 and Lemma 3.13, we have estimates of $\|\omega(t)\|$ and $\|\rho(t)\|$. Substituting the estimates into

$$\|u_h(t) - u(t)\| \leq \|\omega(t)\| + \|\rho(t)\|,$$

we complete the proof. □

Corollary 3.15. *Let $u(t)$ and $u_h(t)$ be solutions of the equation* (3.2.1) *and the semi-discrete scheme* (3.2.4) *respectively. In the equation* (3.2.1)*, assume for $u_0(t) \in C^1(0, T; C_c^\infty(\Omega))$ and $\eta(t) \in L_2(0, T; C_c^\infty(\Omega))$, there exists $\sigma > 0$ s.t. $\|D^{\vec{\alpha}} u_0\| \leq \sigma^{|\vec{\alpha}|}\|u_0\|$, and $\|D^{\vec{\alpha}}\eta(t)\| \leq \sigma^{|\vec{\alpha}|}\|\eta(t)\|$ for any $\vec{\alpha} \in \mathbb{N}_0^d$. Letting $p = 2r+1$, $r \in \mathbb{N}_0$ and $h\sigma/\pi < 1$, we have*

$$\|u_h(t) - u(t)\| \leq C\left(\frac{h\sigma}{\pi}\right)^p,$$

where the constant C is independent of p and h, but may be dependent on σ and t.

Proof. We derive that

$$\|D^{\vec{\alpha}} u(t)\| \leq C(t)\sigma^{|\vec{\alpha}|} \text{ and } \|D^{\vec{\alpha}}\dot{u}(t)\| \leq C(t)\sigma^{|\vec{\alpha}|}, C(t) \in L_2([0, T]), \qquad (3.2.10)$$

in the same approach as deriving (3.1.34) in Lemma 3.6, where the boundary integral term is 0 when applying Green's Theorem because $D^{\vec{\alpha}}u$ vanish on the boundary of Ω. Therefore u satisfy the condition for (2.4.8), and so do u_0 and η. Together with Theorem 3.14, the result follows. $\square$

3.2.4 Error of the Full Scheme

The full scheme corresponding to the Dirichlet boundary condition case is

$$\left(\frac{U^{k+1}-U^k}{\delta t},\chi\right)+a((1-\theta)U^{k+1}+\theta U^k,\chi)=(\eta_h(t_k),\chi) \text{ for any } \chi\in X^{N,p}(\Omega). \tag{3.2.11}$$

We still write the error in the same form of (3.1.56), where R_h is given in (3.2.8). From Lemma 3.12, we have

$$\|\rho^m\|\leq C|\mathcal{B}_{h,p}u(T)-u(T)|.$$

An estimate of ω^m is given by the following lemma.

Lemma 3.16. *Let $\theta\leq 1/2$ in the scheme* (3.2.7). *We have that*

$$\begin{aligned}\|\omega^m\| \;&\leq C(\|\mathcal{B}_{h,p}u_0-u_0\|+|\mathcal{B}_{h,p}u_0-u_0|_1\\ &\quad+\textstyle\int_0^T|\mathcal{B}_{h,p}\dot{u}(s)-\dot{u}(s)|_1\mathrm{d}s+\delta t\int_0^T\|\ddot{u}(s)\|\mathrm{d}s\\ &\quad+\delta t\sum_{k=0}^{m-1}\|\mathcal{B}_{h,p}\eta(t_k)-\eta(t_k)\|).\end{aligned}$$

Proof. The proof of the lemma resembles the proof of Lemma 3.10 except for the following modifications.

- Replace the spaces $\mathcal{H}^{p+1}(\mathbb{T}^d)$ and $X^{N,p}(\mathbb{T}^d)$ by $\mathcal{H}_0^{p+1}(\Omega)$ and $X_0^{N,p}(\Omega)$, respectively.
- Replace reference of equations numbered (3.1.55), (3.1.26) and (3.1.2) by (3.2.11), (3.2.8) and (3.2.2), respectively.

□

Then the error of the full scheme is bounded in the following theorem.

Theorem 3.17. *Let $u(T)$ and U^m be the solution of the equation* (3.1.1) *and scheme* (3.2.7) *at time T respectively. Assume that $u(t)$ and $\eta(t)$ satisfy the assumptions of Corollary 3.15 and $\theta \leq 1/2$. We have*

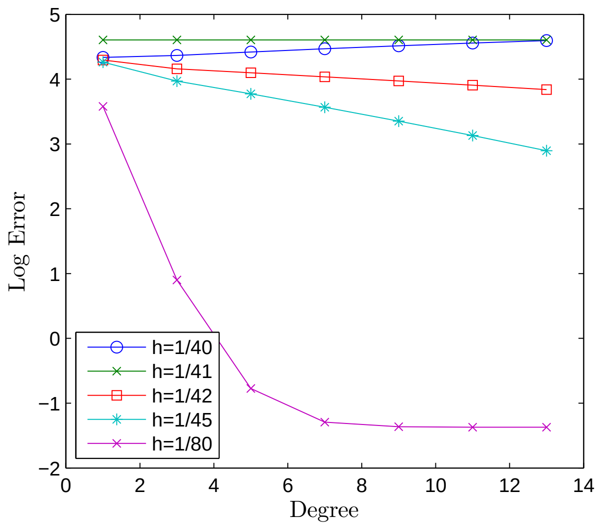
$$\|u(T) - U^m\| \leq C\left(\left(\frac{h\sigma}{\pi}\right)^p + \delta t\right).$$

Proof. The proof of the theorem are very similar to that of Theorem 3.11 with the following modification.

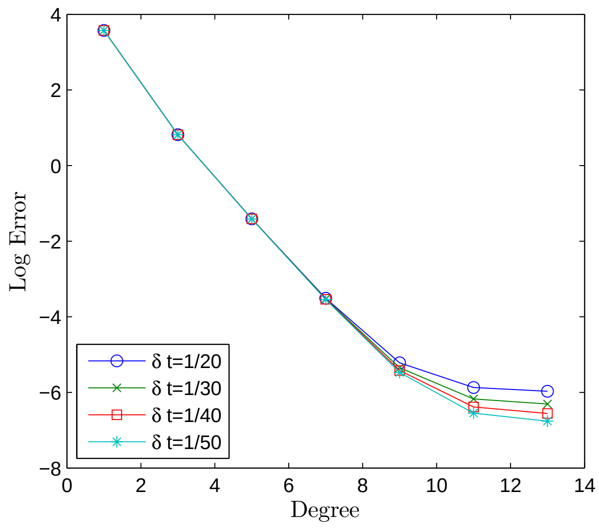
- Replace Lemma 3.3, Lemma 3.10, and Corollary 2.15 by Lemma 3.12, Lemma 3.16, and Corollary 2.20, respectively.
- Replace operator $S_{h,p}$ by $\mathcal{B}_{h,p}$.
- Replace (3.1.34) by assumptions of $u(t)$ and (3.2.10).

□

3.2.5 Numerical Experiment Results



(a) Convergence with different mesh size



(b) Convergence with different time-step size

Figure 3.5: Scheme for heat equation with Dirichlet condition

The experiment is similar to that in Section 3.1.8.3, except that we use the scheme in Section 3.2.2 and choose the true solution to be $u(x,t) = \cos(t)\sin(41\pi x)$. Although vanishing on boundary, $u(t)$ is not in $H^{p+1}(\mathbb{T})$, since its first derivatives are not periodic. We plot the convergence of errors for the schemes with various mesh size h and various time-step size δt in Figure 3.5a and Figure 3.5b, respectively. Figure 3.5a shows that the error converges only when $h < 1/41$, which agrees with Corollary 3.15, but when $h = 1/80$ the error stops converging when the degree is high. Figure 3.5b confirms that this is caused by the temporal errors, which agrees with Theorem 3.17.

Chapter 4

B-spline Finite Element Method for the Wave Equation

In this chapter, we investigate the B-spline FEM for the wave equations. The treatment is studied similar to that in Chapter 3. For more details of the wave equation, the reader may refer to [31, 74].

4.1 Problem with Periodic Boundary Condition

4.1.1 Model Problem

The problem under consideration is the wave equation

$$\frac{\partial^2}{\partial t^2}u(\boldsymbol{x},t) - \Delta u(\boldsymbol{x},t) = \eta(\boldsymbol{x},t),\ (\boldsymbol{x},t) \in \mathbb{T}^d \times [0,\infty), \tag{4.1.1}$$

with initial value

$$u(\boldsymbol{x},0) = u_0(\boldsymbol{x}),$$

and

$$\frac{\partial}{\partial t}u(\boldsymbol{x},0) = v_0(\boldsymbol{x}).$$

We assume $u_0, v_0 \in \mathcal{H}^{p+1}(\mathbb{T}^d)$ and $\eta(t) \in L_2(0,T;\mathcal{H}^{p+1}(\mathbb{T}^d))$.

A weak formulation for the problem is finding $u(t) \in L_2(0,T;\mathcal{H}^1(\mathbb{T}^d))$ with $\dot{u}(t) \in L_2(0,T;L_2(\mathbb{T}^d))$, $\ddot{u}(t) \in L_2(0,T;\mathcal{H}^{-1}(\mathbb{T}^d))$ satisfying

$$\frac{d^2}{dt^2}(u(t),v) + a(u(t),v) = (\eta(t),v) \qquad \text{for any } v \in \mathcal{H}^1(\mathbb{T}^d), \tag{4.1.2}$$

with $u(0) = u_0 \in \mathcal{H}^1(\mathbb{T}^d)$, and $\dot{u}(0) = v_0 \in L_2(\mathbb{T}^d)$. According to Theorem 24.A in [92], the problem has a unique solution. Moreover, since the initial value $u_0(\boldsymbol{x}) \in \mathcal{H}^{p+1}(\mathbb{T}^d)$, this theorem also implies that the solution $u \in L_2(0,T;\mathcal{H}^{p+1}(\mathbb{T}^d))$.

4.1.2 B-spline Scheme

The approximation in $X^{N,p}(\mathbb{T}^d)$ to the solution of (4.1.2) is given by finding $u_h(t) \in X^{N,p}(\mathbb{T}^d)$ such that

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}(u_h(t),v) + a(u_h(t),v) = (\eta(t),v) \qquad \text{for any } v \in X^{N,p}(\mathbb{T}^d).$$

Similar to the heat equation case, we approximate the non-homogeneous term by $\eta_h(t) \in X^{N,p}(\mathbb{T}^d)$ obtained from the B-spline interpolation. Then the B-spline approximation is $u_h(t) \in X^{N,p}(\mathbb{T}^d)$ such that

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}(u_h(t),v) + a(u_h(t),v) = (\eta_h(t),v) \qquad \text{for any } v \in X^{N,p}(\mathbb{T}^d). \tag{4.1.3}$$

Applying the Galerkin method leads to the ODE system

$$\boldsymbol{\mathcal{M}}\ddot{\vec{\alpha}}(t) + \boldsymbol{\mathcal{S}}\vec{\alpha}(t) = \vec{l}(t), \tag{4.1.4}$$

with $\ddot{\vec{\alpha}}(t) = (\ddot{\alpha}_0(t), \ldots, \ddot{\alpha}_{N-1}(t))^T$. $\boldsymbol{\mathcal{M}}$, $\boldsymbol{\mathcal{S}}$ and $\vec{l}(t)$ are the mass matrix, stiffness matrix, and load vector as in (3.1.17) and (3.1.4).

Recall that $\boldsymbol{\xi}_j \in \mathbb{R}^d$ are the points for interpolation defined in (2.4.1) and $b_{j,p}$ are multi-variate B-spline with scalar index defined in (1.5.3). Let $\eta_h(t) = \sum_{j=0}^{N^d-1} \beta_j(t) b_{j,p}$, where vector $\vec{\beta}(t)$ is

$$\vec{\beta}(t) = \boldsymbol{\mathcal{G}}^{-1}\vec{\eta}(t),$$

with

$$\vec{\eta}_j(t) = \eta(\boldsymbol{\xi}_j, t).$$

Taking these expressions into the scheme (4.1.3) leads to the semi-discretization form

$$\boldsymbol{\mathcal{M}}\ddot{\vec{\alpha}}(t) + \boldsymbol{\mathcal{S}}\vec{\alpha}(t) = \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{G}}^{-1}\vec{\eta}(t).$$

To diagonalize the matrices, substituting the expressions (3.1.19), (3.1.21) into the equation leads to

$$\boldsymbol{\Lambda}_{\mathcal{M}}\ddot{\widehat{\alpha}}(t) + \boldsymbol{\Lambda}_{\mathcal{S}}\widehat{\alpha}(t) = \boldsymbol{\Lambda}_{\mathcal{M}}\boldsymbol{\Lambda}_{\mathcal{G}}^{-1}\widehat{\eta}(t), \tag{4.1.5}$$

with $\widehat{\alpha}(t) = \boldsymbol{\mathcal{F}}\vec{\alpha}(t)$ and $\widehat{\eta}(t) = \boldsymbol{\mathcal{F}}\vec{\eta}(t)$. $\boldsymbol{\Lambda}_{\mathcal{M}}$, $\boldsymbol{\Lambda}_{\mathcal{S}}$, $\boldsymbol{\Lambda}_{\mathcal{G}}$ and $\boldsymbol{\mathcal{F}}$ are as defined in (3.1.22), (3.1.20) and (3.1.23). Since the matrices in (4.1.5) are diagonal, the equation is a series of second order ODEs.

Recall that $t_k = \delta tk$. We choose Newmark's method to solve the problem numerically. For more detail on the method, the reader may see, for example, [91, 84]. Applying the method gives the two-step scheme

$$\begin{aligned}(\boldsymbol{\Lambda}_{\mathcal{M}} + \kappa\Delta t^2\boldsymbol{\Lambda}_{\mathcal{S}})\widehat{\alpha}^{k+1} = \ & \{2\boldsymbol{\Lambda}_{\mathcal{M}} - (0.5 + \upsilon - 2\kappa)\Delta t^2\boldsymbol{\Lambda}_{\mathcal{S}}\}\widehat{\alpha}^{k} \\ & -\{\boldsymbol{\Lambda}_{\mathcal{M}} + (0.5 - \upsilon + \kappa)\Delta t^2\boldsymbol{\Lambda}_{\mathcal{S}}\}\widehat{\alpha}^{k-1} \\ & +\Delta t^2\boldsymbol{\Lambda}_{\mathcal{M}}\boldsymbol{\Lambda}_{\mathcal{G}}^{-1}\{\kappa\widehat{\eta}^{k+1} + (0.5 + \upsilon - 2\kappa)\widehat{\eta}^{k} \\ & +(0.5 - \upsilon + \kappa)\widehat{\eta}^{k-1}\},\end{aligned} \tag{4.1.6}$$

where $\kappa, \upsilon \in [0, 1]$ are Newmark parameters and $\widehat{\eta}^k = \widehat{\eta}(t_k)$. The initial value $\vec{\alpha}^0$ is obtained by interpolating the initial value u_0, that is,

$$\vec{\alpha}^0 = \boldsymbol{\mathcal{G}}^{-1}\vec{u_0}, \tag{4.1.7}$$

with $(\vec{u_0})_i = u_0(\boldsymbol{\xi}_i)$. Using the forward difference to approximate the differentiation, that is

$$u'(0) \approx \frac{u(t_1) - u(0)}{\delta t},$$

and interpolating both sides with B-splines, we have

$$\vec{\alpha}^1 = \boldsymbol{\mathcal{G}}^{-1}(\vec{u_0} + \delta t\vec{v_0}), \tag{4.1.8}$$

with $(\vec{v_0})_i = v_0(\boldsymbol{\xi}_i)$.

Algorithm 4.1. *1. Compute the diagonal matrices $\vec{v}_{\mathcal{G}}$, $\vec{v}_{\mathcal{M}}$ and $\vec{v}_{\mathcal{S}}$ using Step 1 in Algorithm 3.2.*

2. Obtain the initial value $\vec{\alpha}^0$ and $\vec{\alpha}^1$ with the equation (4.1.7) *and* (4.1.8), *and then use FFT shown in Section 1.3.1 to obtain $\widehat{\alpha}^0 = \boldsymbol{\mathcal{F}}\vec{\alpha}^0$ and $\widehat{\alpha}^1 = \boldsymbol{\mathcal{F}}\vec{\alpha}^1$.*

3. For each time step k from 1 to $m-1$,

- *Apply FFT to give $\hat{\eta}^k = \boldsymbol{\mathcal{F}}\vec{\eta}^k$.*
- *Compute $\widehat{\alpha}^{k+1}$ using the scalar relation implied by the scheme* (4.1.6) *where the scalar value corresponding to diagonal value of $\boldsymbol{\Lambda}_{\mathcal{M}}$, $\boldsymbol{\Lambda}_{\mathcal{S}}$ and $\boldsymbol{\Lambda}_{\mathcal{G}}$ is given by $\vec{v}_{\mathcal{M}}$, $\vec{v}_{\mathcal{S}}$ and $\vec{v}_{\mathcal{G}}$ respectively.*

4. Compute the coefficients for B-spline approximation of the solution $u(\cdot, T)$ with $\vec{\alpha}^m = \boldsymbol{\mathcal{F}}^{-1}\widehat{\alpha}^m$.

Each time step costs $\mathcal{O}(dN^d \log N)$ operations for the same reason as that for Algorithm 3.2. In the case when the non-homogeneous term $\eta = 0$, the cost is only $\mathcal{O}(N^d)$.

4.1.3 Error of the Semi-discrete Scheme

Similar to the error analysis for the heat equation, we split the error into two parts

$$u(t) - u_h(t) = \rho(t) + \omega(t), \tag{4.1.9}$$

where $\rho(t) = u(t) - R_h u(t)$, $\omega(t) = R_h u(t) - u_h(t)$. $\|\rho(t)\|$ is estimated by Lemma 3.3. Estimation of $\|\omega(t)\|$ is given by following Lemma.

Lemma 4.2.

$$\begin{aligned}\|\dot{\omega}(t)\| + \|\omega(t)\| \le\ & C(|v_{0,h} - S_{h,p}v_0|_1 + \|v_0 - S_{h,p}v_0\| + |u_0 - S_{h,p}u_0|_1 \\ & + \textstyle\int_0^t \|\eta(\tau) - S_{h,p}\eta(\tau)\| + |\ddot{u}(\tau) - S_{h,p}\ddot{u}(\tau)|_1 \mathrm{d}\tau).\end{aligned}$$

Proof. Taking the difference of the equation (4.1.2) and (4.1.3) leads to

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}(u(t)-u_h(t),\nu)+a(u(t)-u_h(t),\nu)=(\eta(t)-\eta_h(t),\nu) \qquad \text{for any } \nu \in X^{N,p}(\mathbb{T}^d).$$

Together with the definition of $R_h u$ in (3.1.26), this implies

$$\begin{aligned} a(\omega(t),\nu) &= a(R_h u(t)-u_h(t),\nu) \\ &= a(u(t)-u_h(t),\nu) \\ &= (\eta(t)-\eta_h(t),\nu)-\tfrac{\mathrm{d}^2}{\mathrm{d}t^2}(u(t)-u_h(t),\nu). \end{aligned} \tag{4.1.10}$$

Taking a sum of (4.1.10) with the equation

$$\begin{aligned} \tfrac{\mathrm{d}^2}{\mathrm{d}t^2}(\omega(t),\nu) &= \tfrac{\mathrm{d}^2}{\mathrm{d}t^2}(R_h u(t)-u(t)+u(t)-u_h(t),\nu) \\ &= \tfrac{\mathrm{d}^2}{\mathrm{d}t^2}(-\rho(t),\nu)+\tfrac{\mathrm{d}^2}{\mathrm{d}t^2}(u(t)-u_h(t),\nu). \end{aligned}$$

This leads to

$$(\ddot{\omega}(t),\nu)+a(\omega(t),\nu)=-(\ddot{\rho}(t),\nu)+(\eta(t)-\eta_h(t),\nu).$$

Choosing $\nu=\dot{\omega}(t)$ and using the triangle inequality, it follows that

$$\frac{\mathrm{d}}{\mathrm{d}t}(\|\dot{\omega}(t)\|^2+|\omega(t)|_1^2)\le 2\|\eta(t)-\eta_h(t)\|\|\dot{\omega}(t)\|+2\|\ddot{\rho}(t)\|\|\dot{\omega}(t)\|.$$

Integrating w.r.t. t gives

$$\begin{aligned} \|\dot{\omega}(t)\|^2+|\omega(t)|_1^2 \le\ & \|\dot{\omega}(0)\|^2+|\omega(0)|_1^2 \\ & +2\textstyle\int_0^t(\|\eta(s)-\eta_h(s)\|+\|\ddot{\rho}(s)\|)\|\dot{\omega}(s)\|\mathrm{d}s \\ \le\ & \|\dot{\omega}(0)\|^2+|\omega(0)|_1^2 \\ & +2\textstyle\int_0^t\|\eta(s)-\eta_h(s)\|+\|\ddot{\rho}(s)\|\mathrm{d}s \displaystyle\max_{s\in[0,t]}\|\dot{\omega}(s)\| \\ \le\ & \|\dot{\omega}(0)\|^2+|\omega(0)|_1^2 \\ & +2\left(\textstyle\int_0^t\|\eta(s)-\eta_h(s)\|+\|\ddot{\rho}(s)\|\mathrm{d}s\right)^2+\tfrac{1}{2}\left(\displaystyle\max_{s\in[0,t]}\|\dot{\omega}(s)\|\right)^2. \end{aligned}$$

The equation holds for any $t\in[0,T]$, which leads to

$$\begin{aligned} \left(\max_{s\in[0,T]}\|\dot{\omega}(s)\|\right)^2+\left(\max_{s\in[0,T]}|\omega(t)|_1\right)^2 \le\ & \|\dot{\omega}(0)\|^2+|\omega(0)|_1^2 \\ & +2\left(\textstyle\int_0^T\|\eta(s)-\eta_h(s)\|+\|\ddot{\rho}(s)\|\mathrm{d}s\right)^2 \\ & +\tfrac{1}{2}\left(\displaystyle\max_{s\in[0,T]}\|\dot{\omega}(s)\|\right)^2. \end{aligned}$$

Since T is arbitrary, choosing $T = t$ it follows that

$$\begin{aligned}\left(\max_{s\in[0,t]} \|\dot\omega(s)\|\right)^2 + \left(\max_{s\in[0,T]} |\omega(t)|_1\right)^2 \leq\ & 2\|\dot\omega(0)\|^2 + 2|\omega(0)|_1^2 \\ & +4\left(\int_0^t \|\eta(s)-\eta_h(s)\| + \|\ddot\rho(s)\|\mathrm{d}s\right)^2.\end{aligned}$$

Hence,

$$\begin{aligned}\|\dot\omega(t)\|^2 + |\omega(t)|_1^2 \leq\ & 2\|\dot\omega(0)\|^2 + 2|\omega(0)|_1^2 \\ & +4\left(\int_0^t \|\eta(s)-\eta_h(s)\| + \|\ddot\rho(s)\|\mathrm{d}s\right)^2.\end{aligned}$$

Since $R_h u(t)$ and $u_h(t)$ are same on the boundary of the domain, $\omega(t) = R_h u(t) - u_h(t)$ vanishes at the boundary. Friedrich's inequality (Lemma 3.6 [39]) implies that $\|\omega(t)\| \leq C|\omega(t)|_1$. It follows that

$$\begin{aligned}\|\dot\omega(t)\| + \|\omega(t)\| \leq\ & C(\|\dot\omega(t)\| + |\omega(t)|_1) \\ \leq\ & C(\|R_h v_0 - v_{0,h}\| + |R_h u_0 - u_{0,h}|_1 \\ & + \int_0^t \|\eta(s)-\eta_h(s)\| + \|\ddot\rho(s)\|\mathrm{d}s).\end{aligned} \tag{4.1.11}$$

Now we estimate the summands on the right hand side of the inequality individually. Lemma 3.3 gives

$$\|\ddot\rho(s)\| \leq C|\ddot u(s) - S_{h,p}\ddot u(s)|_1.$$

Since $v_{0,h} = S_{h,p}v_0$ and $u_{0,h} = S_{h,p}u_0$, together with Lemma 3.3, it follows that

$$\begin{aligned}\|R_h v_0 - v_{0,h}\| \leq\ & \|R_h v_0 - v_{0,h}\| + \|S_{h,p}v_0 - v_0\| \\ \leq\ & C|S_{h,p}v_0 - v_{0,h}|_1 + \|S_{h,p}v_0 - v_0\|,\end{aligned}$$

and

$$\begin{aligned}|R_h u_0 - u_{0,h}|_1 \leq\ & |R_h u_0 - u_0|_1 + |S_{h,p}u_0 - u_0|_1 \\ \leq\ & 2|S_{h,p}u_0 - u_0|_1.\end{aligned}$$

Taking the estimates into inequality (4.1.11) completes the proof.

□

Theorem 4.3. *Let $u_h(t)$ be the semi-discrete approximation in (4.1.4) to the wave equation (4.1.1). Assume the degree of spline $p = 2r+1$, $r \in \mathbb{N}_0$. We have*

$$\begin{aligned}\|u(t) - u_h(t)\| \leq\ & C(\mathcal{Q}_{0,p}(v_0) + \mathcal{Q}_{1,p}(v_0) + \mathcal{Q}_{1,p}(u_0) + \mathcal{Q}_{1,p}(u) \\ & + \int_0^t \mathcal{Q}_{0,p}(\eta(\tau)) + \mathcal{Q}_{1,p}(\ddot u(\tau))\mathrm{d}\tau).\end{aligned}$$

Proof. The result follows from the equation (4.1.9), Lemma 3.3, and Lemma 4.2. □

Corollary 4.4. *Let $u_h(t)$ be the semi-discrete approximation in* (4.1.4) *to the wave equation* (4.1.1). *Assume $u_0, v_0 \in C^\infty(\mathbb{T}^d)$ and $\eta(t) \in C^1(0,T;C^\infty(\mathbb{T}^d))$, s.t. $\|D^{\vec{\alpha}}u_0\| \le \sigma^{|\vec{\alpha}|}\|u_0\|$, $\|D^{\vec{\alpha}}v_0\| \le \sigma^{|\vec{\alpha}|}\|v_0\|$ and $\|D^{\vec{\alpha}}\eta(t)\| \le \sigma^{|\vec{\alpha}|}\|\eta(t)\|$ for any $\vec{\alpha} \in \mathbb{N}_0^d$. For $p = 2r+1$, $r \in \mathbb{N}_0$ and $h\sigma/\pi < 1$, we have*

$$\|u_h(t) - u(t)\| \le C\left(\frac{h\sigma}{\pi}\right)^p,$$

where the constant C is independent of p and h, but may be dependent on σ and t.

Proof. We first show that the assumptions for the initial value and the non-homogeneous term yield the following estimates

$$\|D^{\vec{\beta}}u(t)\| \le C_1(t)\sigma^{|\vec{\beta}|}, \text{ and } \|D^{\vec{\beta}}\ddot{u}(t)\| \le C_2(t)\sigma^{|\vec{\beta}|} \text{ for any } \vec{\beta} \ne 0 \in \mathbb{N}_0^d, \quad (4.1.12)$$

where $C_1(t) \in C^2([0,T])$ and $C_2(t) \in C^1([0,T])$ are independent of $\vec{\alpha}$. Note that, by abuse of notation both $C_1(t)$ and $C_2(t)$ do not necessarily remain the same value in different occasions. On both side of the equation (4.1.1), by taking derivative $D^{\vec{\alpha}}$, multiplying $D^{\vec{\alpha}}\dot{u}(t)$ and integrating over $\mathbb{T}^d$, we have

$$\int_{\mathbb{T}} D^{\vec{\alpha}}\ddot{u}(t)D^{\vec{\alpha}}\dot{u}(t)\mathrm{d}\boldsymbol{x} - \int_{\mathbb{T}} D^{\vec{\alpha}}\Delta u(t)D^{\vec{\alpha}}\dot{u}(t)\mathrm{d}\boldsymbol{x} = \int_{\mathbb{T}} D^{\vec{\alpha}}\eta(t)D^{\vec{\alpha}}\dot{u}(t)\mathrm{d}\boldsymbol{x}. \quad (4.1.13)$$

From the Green's Theorem, the second term in the left hand side of the equality is

$$\int_{\mathbb{T}}(\Delta D^{\vec{\alpha}}u(t))D^{\vec{\alpha}}\dot{u}(t)\mathrm{d}\boldsymbol{x} = -\int_{\mathbb{T}} \nabla D^{\vec{\alpha}}u(t)\nabla D^{\vec{\alpha}}\dot{u}(t)\mathrm{d}\boldsymbol{x} + \int_{\partial\mathbb{T}}(D^{\vec{\alpha}}\dot{u}(t))\nabla(D^{\vec{\alpha}}u(t)\cdot\vec{n})\mathrm{d}\boldsymbol{x},$$

where the boundary integral is actually 0 because $D^{\vec{\alpha}}u$ and $D^{\vec{\alpha}}\dot{u}$ is periodic for all $\vec{\alpha} \le p+1$. It follows that

$$\begin{aligned}\tfrac{d}{dt}(\|D^{\vec{\alpha}}\dot{u}(t)\|^2 + \sum_{j=1}^{d}\|\partial_{x_j}D^{\vec{\alpha}}u(t)\|^2) &\le 2\textstyle\int_{\mathbb{T}^d} D^{\vec{\alpha}}\eta(t)D^{\vec{\alpha}}\dot{u}(t)dx \\ &\le \|D^{\vec{\alpha}}\eta(t)\|^2 + \|D^{\vec{\alpha}}\dot{u}(t)\|^2,\end{aligned}$$

where the boundary term vanish when applying Green's Theorem because $D^{\vec{\alpha}}u$ is periodic for all $\alpha \le p+1$.

Gronwall's inequality [31] implies that

$$\begin{aligned}\|D^{\vec{\alpha}}\dot{u}(t)\|^2 + \sum_{j=1}^{d}\|\partial_{x_j}D^{\vec{\alpha}}u(t)\|^2 &\le \textstyle\int_0^t \|D^{\vec{\alpha}}\eta(\tau)\|^2 \mathrm{d}\tau + \|D^{\vec{\alpha}}v_0\|^2 + \sum_{j=1}^{d}\|\partial_{x_j}D^{\vec{\alpha}}u_0\|^2 \\ &\le \sigma^{2|\vec{\alpha}|}(\textstyle\int_0^t \|\eta(\tau)\|^2\mathrm{d}\tau + \|v_0\|^2 + d\sigma\|u_0\|^2) \\ &\le (C_1(t)\sigma^{|\vec{\alpha}|+1})^2.\end{aligned}$$

It follows from the inequality that for any $j = 1, \dots, d$,

$$\|\partial_{x_j}D^{\vec{\alpha}}u(t)\|^2 \le C_1(t)\sigma^{|\vec{\alpha}|+1}.$$

Since the inequality is true for any $\vec{\alpha} \in \mathbb{N}_0^d$, by letting $\vec{\beta} = \vec{\alpha} + \boldsymbol{e}^j$, we have, for any $\vec{\beta} \ne 0 \in \mathbb{N}_0^d$,

$$\|D^{\vec{\beta}}u(t)\|^2 \le C_1(t)\sigma^{|\vec{\beta}|},$$

where $\boldsymbol{e}^j = (\underbrace{0, \dots, 0, 1}_{j}, 0, \dots, 0)$.

As for $\|D^{\vec{\alpha}}\ddot{u}(t)\|$, taking derivative $D^{\vec{\alpha}}$ and L_2 norm on both side of the equation (4.1.1) gives

$$\|D^{\vec{\alpha}}\ddot{u}(t) - D^{\vec{\alpha}}\Delta u(t)\| = \|D^{\vec{\alpha}}\eta(t)\|.$$

Applying triangle inequality leads to

$$\begin{aligned}\|D^{\vec{\alpha}}\ddot{u}(t)\| &\le \|D^{\vec{\alpha}}\Delta u(t)\| + \|D^{\vec{\alpha}}\eta(t)\| \\ &\le C_2(t)\sigma^{|\vec{\alpha}|},\end{aligned}$$

where $C_2(t)$ is dependant on σ and t. Theorem 4.3 together with the estimate (4.1.12) and (2.4.8) completes the proof. □

4.1.4 Stability

Since all matrices in scheme (4.1.6) are diagonal, it is equivalent to a series of recurrence relations, for each $i = 0, \dots, N^d - 1$,

$$\begin{aligned}(1 + \kappa\delta t^2\gamma_i)\widehat{\alpha}^{k+1} = \ & \{2 - (0.5 + \upsilon - 2\kappa)\delta t^2\gamma_i\}\widehat{\alpha}^k \\ & -\{1 + (0.5 - \upsilon + \kappa)\delta t^2\gamma_i\}\widehat{\alpha}^{k-1} \\ & +\delta t^2(\gamma_i^G)^{-1}\{\kappa\widehat{\eta}^{k+1} + (0.5 + \upsilon - 2\kappa)\widehat{\eta}^k + (0.5 - \upsilon + \kappa)\widehat{\eta}^{k-1}\},\end{aligned}$$

where γ_i and γ_i^G are as defined in Section 3.1.6. The relation is stable if the roots of its characteristic function,

$$(1+\kappa\delta t^2\gamma_i)\zeta^2+(-2+(0.5+\upsilon-2\kappa)\delta t^2\gamma_i)\zeta+(1+(0.5-\upsilon+\kappa)\delta t^2\gamma_i)=0, \quad (4.1.14)$$

lie in the unit disc. Letting $\zeta = (\xi+1)/(\xi-1)$ and substituting into equation (4.1.14) gives

$$\delta t^2\gamma_i\xi^2+(2\upsilon-1)\delta t^2\gamma_i\xi+4+2(2\kappa-\upsilon)\delta t^2\gamma_i=0. \quad (4.1.15)$$

If the roots of equation (4.1.15) have non-positive real part, roots of equation (4.1.14) lie in the unit disc. From Routh-Hurwitz's theorem [48, p.1086], all the roots of equation (4.1.15) have non-positive real parts if and only if

$$\begin{cases} \delta t^2\gamma_i > 0 \\ (2\upsilon-1)\delta t^2\gamma_i \geq 0 \\ 4+2(2\kappa-\upsilon)\delta t^2\gamma_i \geq 0 \end{cases}.$$

The conditions show that when $\upsilon < 1/2$ the scheme is always unstable regardless of the value of δt. When $\upsilon \geq 1/2$ and $\upsilon \leq 2\kappa$, the scheme is stable independent of δt. When $\upsilon \geq 1/2$ and $\upsilon > 2\kappa$, to ensure the scheme is stable, we need $\delta t \leq 2^{1/2}[(\upsilon-2\kappa)\gamma_i]^{-1/2}$, for each $i = 0, \dots, N^d-1$. In this case, it's sufficient to ensure stability by having $\delta t \leq 2^{1/2}[(\upsilon-2\kappa)\gamma_{\max}]^{-1/2}$. As shown in Section 3.1.6, $\gamma_{\max} = d(K_{2p-1}/K_{2p+1})(\pi^2/h^2)$. We deduce that when $\upsilon \geq 1/2$ and $\upsilon > 2\kappa$, the time stepping scheme is stable if

$$\delta t \leq \sqrt{2/(d(\upsilon-2\kappa))(K_{2p+1}/K_{2p-1})}(h/\pi). \quad (4.1.16)$$

The stability condition is summarized as

$$\begin{cases} \upsilon < 1/2 & \text{unstable} \\ \upsilon \geq 1/2 \text{ and } \upsilon \leq 2\kappa & \text{always stable} \\ \upsilon \geq 1/2 \text{ and } \upsilon > 2\kappa & \text{stable if (4.1.16) holds} \end{cases}.$$

Similar to the heat equation case, we see a slight improvement for the stability condition as we increase the degree of spline.

4.1.5 Error of the Full Scheme

We only consider the case when $\kappa = 1/4$ and $\upsilon = 1/2$, which is one of the most common choices [84]. In this case, the scheme is unconditionally stable. For simplicity, we first introduce some notations. Recall that the time interval $[0, T]$ is divided into m time-steps with size $\delta t = T/m$ and $t_k = k\delta t$. Let $t_{k+1/2} = (t_{k+1} + t_k)/2$. Let D_t^+ and D_t^- be respectively the forward and backward difference operator

$$D_t^+ v(t_k) = \frac{v(t_{k+1}) - v(t_k)}{\delta t} \text{ and } D_t^- v(t_k) = \frac{v(t_k) - v(t_{k-1})}{\delta t}.$$

It follows that $D_t^- D_t^+$ is the second order central difference,

$$D_t^- D_t^+ v(t_k) = \frac{v(t_{k+1}) - 2v(t_k) + v(t_{k-1})}{\delta t^2}.$$

Define the operator

$$D_t^{\kappa,\upsilon} v(t_k) = \frac{v(t_{k+1}) + 2v(t_k) + v(t_{k-1})}{4}.$$

Then the full scheme is

$$\left(D_t^- D_t^+ U^k, \chi\right) + a(D_t^{\kappa,\upsilon} U^k, \chi) = (D_t^{\kappa,\upsilon} \eta(t_k), \chi) \text{ for any } \chi \in X^{N,p}(\mathbb{T}^d). \quad (4.1.17)$$

We let

$$U^k - u(t_k) = \omega^k + \rho^k, \quad (4.1.18)$$

where $\omega^k = U^k - R_h u(t_k)$ and $\rho^k = R_h u(t_k) - u(t_k)$. Let $\omega^{k+1/2} = (\omega^k + \omega^{k+1})/2$. Then the following theorem gives estimates of the approximation of the full scheme.

Theorem 4.5. *Let $u(t_m)$ and U^m be the solution of the equation* (4.1.1) *and scheme* (4.1.6) *at time T respectively. Assume u_0, v_0 and $\eta(t)$ satisfy the assumptions of Corollary 4.4. For $p = 2r + 1$, $r \in \mathbb{N}_0$ and $\delta t \le 1$, we have*

$$\|D_t^+ U^m - \dot{u}(t_{m+1/2})\| + \|U^{m+1/2} - u(t_{m+1/2})\| \le C\left(\left(\frac{h\sigma}{\pi}\right)^p + \delta t\right).$$

Proof. The triangle inequality implies that

$$\begin{aligned}&\|D_t^+U^m - \dot u(t_{m+1/2})\| + \|U^{m+1/2} - u(t_{m+1/2})\| \le \|D_t^+\omega^m\| + \|\omega^{m+1/2}\| \\ &+\|D_t^+\rho^m\| + \|D_t^+u(t_m) - \dot u(t_{m+1/2})\| + \|\rho^m\| + \|\rho^{m+1}\|.\end{aligned} \tag{4.1.19}$$

We first estimate the terms $\|D_t^+\omega^m\|$ and $\|\omega^{m+1/2}\|$. The scheme (4.1.17), definition (3.1.26), and weak formulation (4.1.2) imply that

$$\begin{aligned}&\big(D_t^-D_t^+\omega^k, \chi\big) + a(D_t^{\kappa,\upsilon}\omega^k, \chi)\\ &\quad= -\big(D_t^-D_t^+R_hu(t_k), \chi\big) - a(D_t^{\kappa,\upsilon}R_hu(t_k), \chi) + (D_t^{\kappa,\upsilon}\eta_h(t_k), \chi)\\ &\quad= -\big(D_t^-D_t^+R_hu(t_k), \chi\big) - a(D_t^{\kappa,\upsilon}u(t_k), \chi) + (D_t^{\kappa,\upsilon}\eta_h(t_k), \chi)\\ &\quad= -\big(D_t^-D_t^+R_hu(t_k), \chi\big) + (D_t^{\kappa,\upsilon}\ddot u(t_k), \chi) + (D_t^{\kappa,\upsilon}(\eta_h(t_k) - \eta(t_k)), \chi).\end{aligned}$$

Let $\Psi^k = D_t^{\kappa,\upsilon}\ddot u(t_k) - D_t^-D_t^+R_hu(t_k) + D_t^{\kappa,\upsilon}(\eta_h(t_k) - \eta(t_k))$. It follows that

$$(D_t^-D_t^+\omega^k, \chi) + a(D_t^{\kappa,\upsilon}\omega^k, \chi) = (\Psi^k, \chi). \tag{4.1.20}$$

We choose $\chi = D_t^+(\omega^k + \omega^{k-1})$ in the equation (4.1.20). Expanding the operator D_t^-, the first term in the left hand side of the equation is

$$\begin{aligned}\big(D_t^-D_t^+\omega^k, D_t^+(\omega^k + \omega^{k-1})\big) &= \tfrac{1}{\delta t}\big(D_t^+\omega^k - D_t^+\omega^{k-1}, D_t^+\omega^k + D_t^+\omega^{k-1}\big)\\ &= D_t^-\|D_t^+\omega^k\|.\end{aligned}$$

The second term is

$$\begin{aligned}a(D_t^{\kappa,\upsilon}\omega^k, D_t^+(\omega^k + \omega^{k-1})) &= a\big(\tfrac{\omega^{k+1/2}+\omega^{k-1/2}}{2}, 2\tfrac{\omega^{k+1/2}-\omega^{k-1/2}}{\delta t}\big)\\ &= D_t^-|\omega^{k+1/2}|_1^2.\end{aligned}$$

Then the left hand side of (4.1.20) is

$$D_t^-(\|D_t^+\omega^k\|^2 + |\omega^{k+1/2}|_1^2). \tag{4.1.21}$$

The Cauchy-Schwarz inequality implies that the right hand side of (4.1.20) has the following bounds,

$$\begin{aligned}(\Psi^k, D_t^+(\omega^k + \omega^{k-1})) &\le \|\Psi^k\|\|D_t^+\omega^k\| + \|\Psi^k\|\|D_t^+\omega^{k-1}\|\\ &\le (\|\Psi^k\|^2 + \|D_t^+\omega^k\|^2)/2 + (\|\Psi^k\|^2 + \|D_t^+\omega^{k-1}\|^2)/2\\ &= \|\Psi^k\|^2 + 1/2(\|D_t^+\omega^k\|^2 + \|D_t^+\omega^{k-1}\|^2).\end{aligned} \tag{4.1.22}$$

Let $\alpha_k = \|D_t^+\omega^k\|^2 + |\omega^{k+1/2}|_1^2$. From (4.1.20), (4.1.21) and (4.1.22), we have the inequality

$$D_t^-\alpha^k \le \|\Psi^k\|^2 + 1/2(\alpha^k + \alpha^{k-1}).$$

Expanding the operator D_t^-, we have that

$$\alpha^k - \alpha^{k-1} \le \delta t\|\Psi^k\|^2 + \delta t/2(\alpha^k + \alpha^{k-1}),$$

which can be written as

$$\alpha^k \le \beta\alpha^{k-1} + \frac{\delta t}{1-\delta t/2}\|\Psi^k\|^2,$$

with $\beta = \frac{1+\delta t/2}{1-\delta t/2}$. Since $\beta > 1$, the recurrence relation implies that

$$\begin{aligned}\alpha^m \quad &\le \beta^m\alpha^0 + \tfrac{\delta t}{1-\delta t/2}\sum_{j=0}^{m-1}\beta^j\|\Psi^{k-j}\|^2 \\ &\le \beta^m\alpha^0 + \tfrac{\delta t}{1-\delta t/2}\beta^m\sum_{j=1}^{m}\|\Psi^j\|^2.\end{aligned}$$

From the inequality that

$$\beta^m = \left(\tfrac{1+\delta t/2}{1-\delta t/2}\right)^m = \left(1 + \tfrac{\delta t}{1-\delta t/2}\right)^m \le \exp\left(\tfrac{\delta t}{1-\delta t/2}m\right) = \exp\left(\tfrac{T}{1-\delta t/2}\right),$$

it follows that

$$\alpha^m \le C(\alpha^0 + \delta t\sum_{j=1}^{m}\|\Psi^j\|^2).$$

Friedrich's inequality (Lemma 3.6 [39]) implies that

$$\begin{aligned}\|D_t^+\omega^m\| + \|\omega^{m+1/2}\| \quad &\le C(\|D_t^+\omega^m\| + |\omega^{m+1/2}|_1) \\ &\le C(\|D_t^+\omega^0\| + |\omega^{1/2}|_1 + \delta t\sum_{j=1}^{m}\|\Psi^j\|) \\ &\le C(\|D_t^+\omega^0\| + |\omega^0|_1 + |\omega^1|_1 + \delta t\sum_{j=1}^{m}\|\Psi^j\|).\end{aligned} \tag{4.1.23}$$

We now estimate the right hand side terms of the inequality. The inequality (3.1.28) and Corollary 2.15 imply that

$$\begin{aligned}|\omega^0|_1 \quad &= |R_hu_0 - S_{h,p}u_0|_1 \\ &\le |R_hu_0 - u_0 + u_0 - S_{h,p}u_0|_1 \\ &\le 2|u_0 - S_{h,p}u_0|_1 \\ &\le C(\tfrac{h\sigma}{\pi})^p\|u_0\|.\end{aligned} \tag{4.1.24}$$

Lemma 3.3, Corollary 2.15, a standard estimate on truncation error of the forward difference [44], and the definition of initial value (4.1.8) lead to

$$\begin{aligned}\|D_t^+\omega^0\| &= \|R_h D_t^+ u(t_0) - D_t^+ U^0\| \\ &\le |D_t^+ u_0 - S_{h,p} D_t^+ u_0|_1 + \|D_t^+ u_0 - S_{h,p} D_t^+ u_0\|. \\ &\le C(\tfrac{h\sigma}{\pi})^p \|D_t^+ u_0\| \\ &= C(\tfrac{h\sigma}{\pi})^p \|u_0 + \mathcal{O}(\delta t)\| \\ &\le C(\tfrac{h\sigma}{\pi})^p \|u_0\| + C(\tfrac{h\sigma}{\pi})^p \delta t.\end{aligned} \tag{4.1.25}$$

The definition of initial value (4.1.8), the inequality (3.1.28), and Taylor expansion yield

$$\begin{aligned}|\omega^1|_1 &= |R_h u(t_1) - U^1|_1 \\ &\le |R_h u(t_1) - (U^0 + \delta t S_{h,p} v_0)|_1 \\ &\le |R_h u(t_1) - u(t_1)|_1 + |u(t_0 + \delta t) - (S_{h,p} u_0 + \delta t S_{h,p} v_0)|_1 \\ &\le |R_h u(t_1) - u(t_1)|_1 + |u(t_0) - S_{h,p} u_0 + \delta t(v_0 - S_{h,p} v_0) + \mathcal{O}(\delta t^2)|_1 \\ &\le |R_h u(t_1) - u(t_1)|_1 + |u_0 - S_{h,p} u_0|_1 + \delta t |v_0 - S_{h,p} v_0|_1 + C\delta t^2.\end{aligned}$$

Theorem 2.14 together with the inequality (4.1.12), (2.4.8), and Corollary 2.15 gives

$$|\omega^1|_1 \le C(\frac{h\sigma}{\pi})^p(\|u(t_0\| + \|u_0\| + \delta t\|v_0\|) + C(\delta t). \tag{4.1.26}$$

Now we estimate the term $\delta t \sum\limits_{k=1}^{m} \|\Psi^k\|$. Let $\Psi^k = \Psi_1^k + \Psi_2^k + D_t^{\kappa,\upsilon}(\eta_h(t_k) - \eta(t_k))$, where $\Psi_1^k = D_t^- D_t^+ u(t_k) - D_t^- D_t^+ R_h u(t_k)$ and $\Psi_2^k = D_t^{\kappa,\upsilon}\ddot{u}(t_k) - D_t^- D_t^+ u(t_k)$. Theorem 2.14 together with the estimate (4.1.12), and (2.4.8) yield that

$$\begin{aligned}\delta t \sum_{k=1}^{m} \|\Psi_1^k\| &\le \delta t \sum_{k=1}^{m} \|D_t^- D_t^+ u(t_k) - S_{h,p} D_t^- D_t^+ u(t_k)\| \\ &\le \delta t \sum_{k=1}^{m} \mathcal{Q}_{0,p}(D_t^- D_t^+ u(t_k))\end{aligned}$$

The truncation error of central difference [44] and a error bound for Riemann sums [54, Chapter 4] imply that

$$\begin{aligned}\delta t \sum_{k=1}^{m} \|\Psi_1^k\| &\le C(\tfrac{h\sigma}{\pi})^{p+1} \delta t \sum_{k=1}^{m} \|D_t^- D_t^+ C_1(t_k)\| \\ &\le C(\tfrac{h\sigma}{\pi})^{p+1} \delta t \sum_{k=1}^{m} \|\ddot{C}_1(t_k)\| + \mathcal{O}(\delta t^2) \\ &\le C(\tfrac{h\sigma}{\pi})^{p+1} \int_0^T \|\ddot{C}_1(t)\| \mathrm{d}t + C\delta t.\end{aligned}$$

Taylor expansion gives that

$$\begin{aligned}\Psi_2^k &= 1/4(\ddot{u}(t_k+h)+\ddot{u}(t_k-h)+2\ddot{u}(t_k)) - D_t^- D_t^+ u(t_k)\\ &= 1/4(\ddot{u}(t_k)+h\dddot{u}(t_k)+\mathcal{O}(\delta t^2)\\ &+\ddot{u}(t_k)-h\dddot{u}(t_k)+2\ddot{u}(t_k)+\mathcal{O}(\delta t^2))-(\ddot{u}(t_k)+\mathcal{O}(\delta t^2))\\ &\le C\delta t^2.\end{aligned}$$

Then it follows that

$$\delta t\sum_{j=1}^{m}\|\Psi^j\| \le C\left(\frac{h\sigma}{\pi}\right)^{p+1} + C\delta t. \tag{4.1.27}$$

Corollary 2.15, Taylor expansion, and an error bound for Riemann sums [54, Chapter 4] imply that

$$\begin{aligned}\delta t\sum_{k=1}^{m-1}\|S_{h,p}D_t^{\kappa,\upsilon}\eta(t_k)-D_t^{\kappa,\upsilon}\eta(t_k)\| &\le C(\tfrac{h\sigma}{\pi})^{p+1}\delta t\sum_{k=1}^{m-1}\|D_t^{\kappa,\upsilon}\eta(t_k)\|\\ &\le C(\tfrac{h\sigma}{\pi})^{p+1}(\delta t\sum_{k=1}^{m-1}\|\eta(t_k)\|+\mathcal{O}(\delta t))\\ &= C(\tfrac{h\sigma}{\pi})^{p+1}(\int_0^T\|\eta(s)\|\mathrm{d}s+\mathcal{O}(\delta t))\\ &\le C(\tfrac{h\sigma}{\pi})^{p+1}\int_0^T\|\eta(s)\|\mathrm{d}s + C(\tfrac{h\sigma}{\pi})^{p+1}\delta t.\end{aligned} \tag{4.1.28}$$

The equality (4.1.23), together with (4.1.24), (4.1.25), (4.1.26), (4.1.27), and (4.1.28) give

$$\|D_t^+\omega^m\| + \|\omega^{m+1/2}\| \le C\left(\frac{h\sigma}{\pi}\right)^{p} + C\delta t. \tag{4.1.29}$$

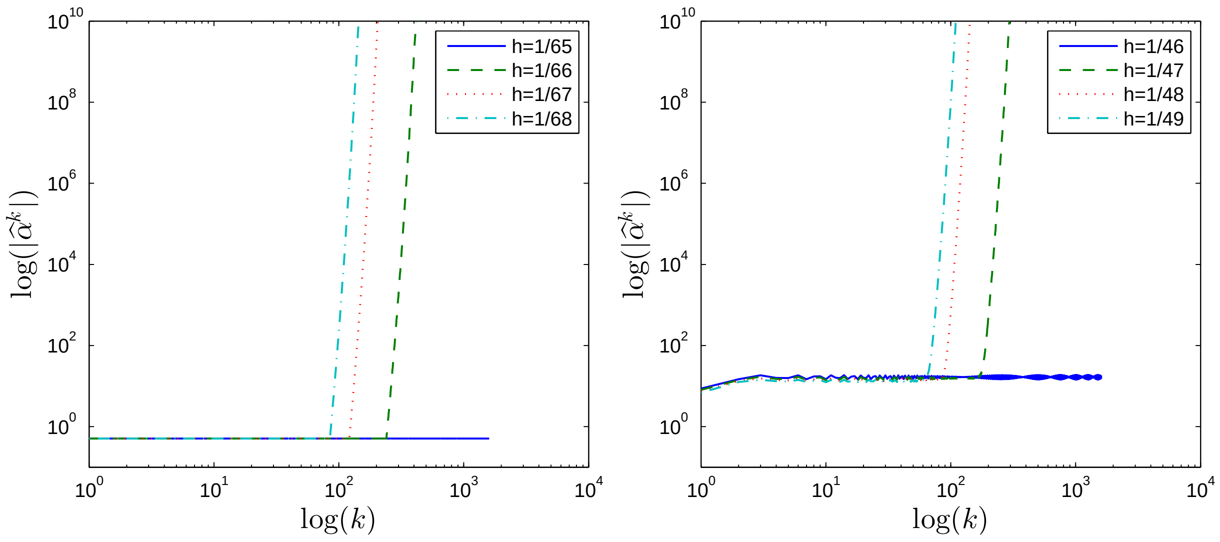
Similarly, the terms $\|\rho^m\|$, $\|\rho^{m+1}\|$, and $\|D_t^+\rho^m\|$ are estimated using Lemma 3.3, Corollary 2.15, and (4.1.12), which give

$$\|\rho^m\| + \|\rho^{m+1}\| + \|D_t^+\rho^m\| \le C\left(\frac{h\sigma}{\pi}\right)^{p} + C\delta t.$$

The estimate together with (4.1.29) and (4.1.19) complete the proof.

□

Although the theorem only gives the error at the time point $t_{m+1/2}$, the estimate



(a) Scheme in $\mathbb{R}$ (b) Scheme in $\mathbb{R}^2$

Figure 4.1: Stability of the Newmark Scheme

at t_m may also be obtained in the following way.

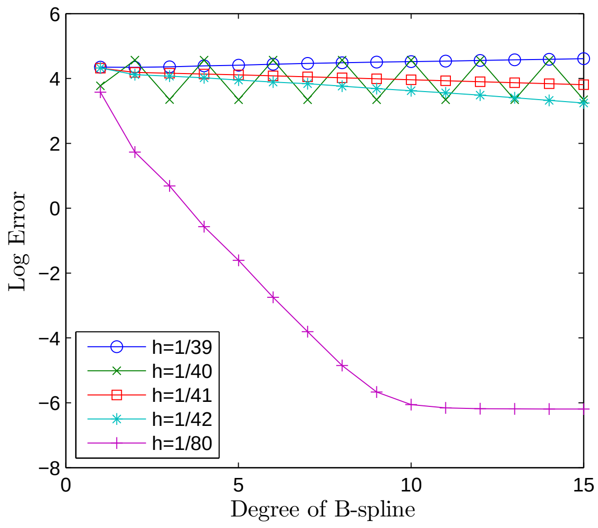
$$\begin{aligned}&\|\tfrac{1}{4}(U^{m+1}+2U^{m+1}+U^{m-1})-u(t_m)\| \le \tfrac{1}{2}\|U^{m+1/2}-u(t_{m+1/2})\|\\&+\tfrac{1}{2}\|U^{m-1/2}-u(t_{m-1/2})\| + \|\tfrac{1}{2}(u(t_{m+1/2})+u(t_{m-1/2})-u(t_m)))\|\\&\le C\left(\tfrac{h\sigma}{\pi}\right)^p + C\delta t.\end{aligned}$$

4.1.6 Numerical Experiment Results

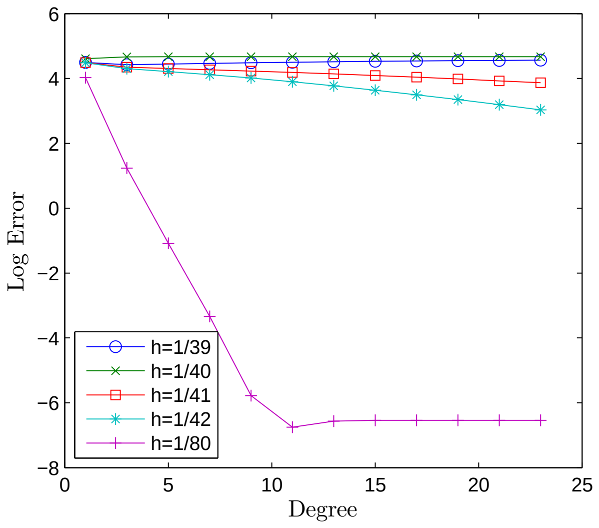
4.1.6.1 Stability

We first test the stability condition by implementing Algorithm 4.1, calculating the maximum norm of coefficient $\vec{\alpha}^k$ for each time step k, and plotting $\log(k)$ against $\log(\|\vec{\alpha}^k\|_\infty)$ for different h.

The parameters of the Newmark scheme are set to be $\upsilon = 0.5$ and $\kappa = 0.2$, and time step $\delta t = 0.0125$. According to the condition (4.1.4), the schemes in $\mathbb{R}$ and $\mathbb{R}^2$ are stable when $h > 0.0152$ and $h > 0.0215$, respectively. Figures 4.1a and 4.1b show the results. The solid line in the plot corresponds to the case where the condition is satisfied ($h = 1/65$ and $h = 1/46$ for the schemes in $\mathbb{R}$ and



(a) Scheme in $\mathbb{R}$



(b) Scheme in $\mathbb{R}^2$

Figure 4.2: Convergence of scheme with different h

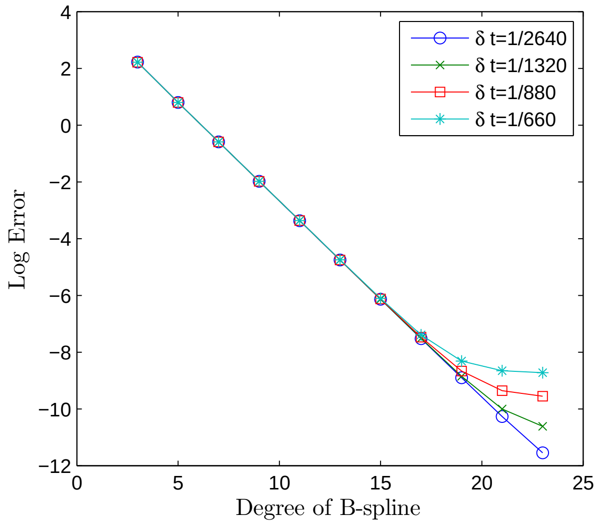
$\mathbb{R}^2$ respectively) while the others does not. This confirms the stability condition analysis. Moreover, since the dashed line corresponds to the scheme which chooses one more mesh-point than the solid line, it can be concluded that the stability condition is sharp.

4.1.6.2 Error

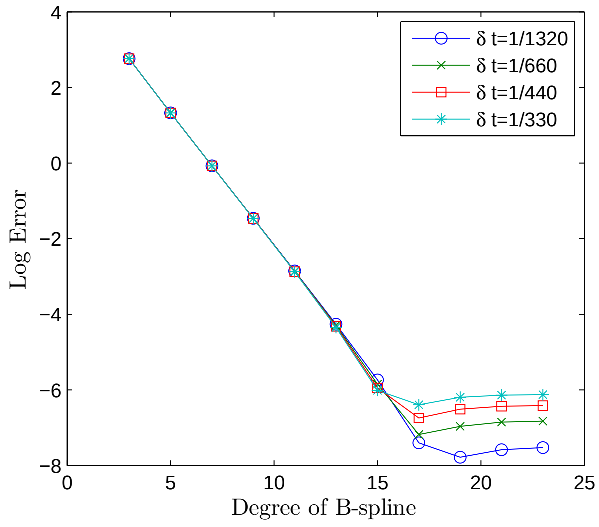
Similar to the experiment in Section 3.1.8.3, we now test the convergence of the full scheme (4.1.6).

We choose the solution of the equation (4.1.1) to be $u = \cos(40\pi x + t)$ and $u = \cos(40\pi x + 40\pi y + t)$, implement the Algorithm 4.1 and plot the error of the approximation against the degree of the B-spline for different mesh size.

As shown in Figure 4.2a and 4.2b, as the degree n increases, only if the mesh size $h < 1/40$ does the error converge. That complies with the result in Corollary 4.4. We can further observe that the convergence stops when reaching a level around 10^{-7}. From Theorem 4.5, this may be caused by the error from temporal



(a) Scheme in $\mathbb{R}$



(b) Scheme in $\mathbb{R}^2$

Figure 4.3: Convergence of scheme with different δt

discretization. To confirm the statement, we also plot degree against error for fixed mesh size but different time step size. As shown in Figure 4.3a and 4.3b, smaller δt improves the convergence of spatial approximation.

4.2 Problem with Dirichlet Boundary Condition

4.2.1 Model Problems

Recall that $\Omega = [0,1]^d$. We consider the wave equation

$$\frac{\partial^2}{\partial t^2} u(\boldsymbol{x},t) - \Delta u(\boldsymbol{x},t) = \eta(\boldsymbol{x},t),\ (\boldsymbol{x},t) \in \Omega \times [0,\infty), \tag{4.2.1}$$

with initial value

$$u(\boldsymbol{x},0) = u_0(\boldsymbol{x}) \in \mathcal{H}_0^{p+1}(\Omega) \text{ and } \frac{\partial}{\partial t} u(\boldsymbol{x},0) = v_0(\boldsymbol{x}) \in \mathcal{H}_0^{p+1}(\Omega),$$

Dirichlet boundary condition

$$u(\boldsymbol{x},t) = 0 \text{ for } \boldsymbol{x} \in \partial\Omega,$$

and the non-homogeneous term

$$\eta(t) \in L_2(0,T;\mathcal{H}_0^{p+1}(\Omega)).$$

A weak formulation corresponding to the problem is finding $u(t) \in L_2(0,T;\mathcal{H}_0^1(\Omega))$ with $\dot{u}(t) \in L_2(0,T;L_2(\Omega))$ and $\ddot{u}(t) \in L_2(0,T;\mathcal{H}^{-1}(\Omega))$ s.t.

$$\frac{d^2}{dt^2}(u(t),v) + a(u(t),v) = (\eta(t),v) \qquad \text{for any } v \in \mathcal{H}_0^1(\Omega), \tag{4.2.2}$$

with $u(0) = u_0 \in \mathcal{H}_0^1(\Omega)$ and $\dot{u}(0) = v_0 \in L_2(\Omega)$.

Analogous to the periodic case, because of Theorem 24.A of [92], the problem has a unique solution. Moreover, since the initial value $u_0 \in \mathcal{H}_0^{p+1}(\Omega)$, this theorem also implies that the solution $u(t) \in L_2(0,T;\mathcal{H}_0^{p+1}(\Omega))$.

4.2.2 B-spline Scheme

The Galerkin approximation in $X_0^{N,p}(\Omega)$ to the solution of (4.2.2) is given by finding $u_h(t) \in X_0^{N,p}(\Omega)$ such that

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}(u_h(t),v) + a(u_h(t),v) = (\eta(t),v) \qquad \text{for any } v \in X_0^{N,p}(\Omega). \tag{4.2.3}$$

Let $\mathcal{B}_{h,p}\eta(t) = \sum_{\boldsymbol{i}\in\mathcal{J}_0^d} l_{\boldsymbol{i}}(t)\psi_{\boldsymbol{i},p}$ be the B-spline interpolant of $\eta(t)$ in the space $X_0^{N,p}(\Omega)$ as shown in Section 3.2.2. Approximating $\eta(t)$ by $\mathcal{B}_{h,p}\eta(t)$ and substituting each basis $\psi_{\boldsymbol{i},p}$, we have the semi-discrete scheme,

$$\boldsymbol{\mathcal{M}}\ddot{\vec{\alpha}}(t) + \boldsymbol{\mathcal{S}}\vec{\alpha}(t) = \boldsymbol{\mathcal{M}}\vec{l}(t). \tag{4.2.4}$$

$\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{S}}$ are the mass and stiffness matrices given in (3.2.5), respectively. Applying Newmark's method to solve the problem, we have the two-step scheme as follows

$$\begin{aligned}(\boldsymbol{\mathcal{M}} + \kappa\delta t^2\boldsymbol{\mathcal{S}})\vec{\alpha}^{k+1} = \ & \{2\boldsymbol{\mathcal{M}} - (0.5+\upsilon-2\kappa)\delta t^2\boldsymbol{\mathcal{S}}\}\vec{\alpha}^k \\ & -\{\boldsymbol{\mathcal{M}} + (0.5-\upsilon+\kappa)\delta t^2\boldsymbol{\mathcal{S}}\}\vec{\alpha}^{k-1} \\ & +\delta t^2\boldsymbol{\mathcal{M}}\{\kappa\vec{l}^{k+1} + (0.5+\upsilon-2\kappa)\vec{l}^k + (0.5-\upsilon+\kappa)\vec{l}^{k-1}\},\end{aligned} \tag{4.2.5}$$

where κ and υ are the Newmark parameters. The vector $\vec{l}^k$ is the coefficient of $\mathcal{B}_{h,p}\eta(t_k)$. The initial values $\vec{\alpha}^0$ and $\vec{\alpha}^1$ are the coefficients of $\mathcal{B}_{h,p}v_0$ and $\mathcal{B}_{h,p}(u_0 + \delta t v_0)$, respectively.

4.2.3 Error of the Semi-discrete Scheme

Since the error analysis in this section is similar to that in Section 4.1.3, we only give the results and list the modifications in the proofs. We still write the error in the form of (4.1.9). Lemma 3.12 gives a bound for $\|\rho(t)\|$. Estimation of $\|\omega(t)\|$ is obtained from the following Lemma.

Lemma 4.6.

$$\begin{aligned}\|\dot{\omega}(t)\| + \|\omega(t)\| \leq\ & C(|v_{0,h} - \mathcal{B}_{h,p}v_0|_1 + \|v_0 - \mathcal{B}_{h,p}v_0\| + |u_0 - \mathcal{B}_{h,p}u_0|_1 \\ & + \textstyle\int_0^t \|\eta(\tau) - \mathcal{B}_{h,p}\eta(\tau)\| + |\ddot{u}(\tau) - \mathcal{B}_{h,p}\ddot{u}(\tau)|_1 \mathrm{d}\tau).\end{aligned}$$

Proof. Our proof of this lemma is the same as that of Lemma 4.2 with the following changes.

- Replace all the spaces $\mathcal{H}^{p+1}(\mathbb{T}^d)$ and $X^{N,p}(\mathbb{T}^d)$ by $\mathcal{H}_0^{p+1}(\Omega)$ and $X_0^{N,p}(\Omega)$, respectively.
- Replace all the operators $S_{h,p}$ by $\mathcal{B}_{h,p}$.
- Replace Lemma 3.3 by Lemma 3.12.
- Replace (4.1.2), (4.1.3) and (3.1.26) by (4.2.2), (4.2.3) and (3.2.8).

□

Theorem 4.7. *Let u_h be the semi-discrete approximation in* (4.2.4) *to the wave equation* (4.2.1)*, then*

$$\begin{aligned}\|u(t) - u_h(t)\| \leq\ & C(\mathcal{Q}_{0,p}(v_0) + \mathcal{Q}_{1,p}(v_0) + \mathcal{Q}_{1,p}(u_0) \\ & + \mathcal{Q}_{1,p}(u) + \textstyle\int_0^t \mathcal{Q}_{0,p}(\eta(\tau)) + \mathcal{Q}_{1,p}(\ddot{u}(\tau))\mathrm{d}\tau).\end{aligned}$$

Proof. The result follows from the equation (3.1.27), Lemma 3.12, and Lemma 4.6. □

Corollary 4.8. *Let $u_h(t)$ be the semi-discrete approximation in (4.2.4) to the wave equation (4.2.1). Assume for $u(t) \in C^2(0,T;C_0^\infty(\Omega))$ and $\eta(t) \in C^1(0,T;C_0^\infty(\Omega))$ there exist $\sigma > 0$ s.t. $\|D^{\vec{\alpha}}u_0\| \le \sigma^{|\vec{\alpha}|}\|u_0\|$, $\|D^{\vec{\alpha}}v_0\| \le \sigma^{|\vec{\alpha}|}\|v_0\|$ and $\|D^{\vec{\alpha}}\eta(t)\| \le \sigma^{|\vec{\alpha}|}\|\eta(t)\|$ for any $\vec{\alpha} \in \mathbb{N}_0^d$. For $p = 2r+1$, $r \in \mathbb{N}_0$ and $h\sigma/\pi < 1$, we have*

$$\|u_h(t) - u(t)\| \le C\left(\frac{h\sigma}{\pi}\right)^p,$$

where the constant C is independent of p and h, but may be dependent on σ and t.

Proof. We derive that

$$\|D^{\vec{\beta}}u(t)\| \le C\sigma^{|\vec{\beta}|}, \text{ and } \|D^{\vec{\beta}}\ddot{u}(t)\| \le C\sigma^{|\vec{\beta}|} \text{ for any } \vec{\beta} \ne 0 \in \mathbb{N}_0^d, \tag{4.2.6}$$

in the same way as deriving (4.1.12) in Lemma 4.4 , where the boundary integral term is 0 when applying Green's Theorem because $D^{\vec{\alpha}}u$ vanish on the boundary of Ω. Theorem 4.7 together with (2.4.8) and (4.2.6) completes the proof.

□

4.2.4 Error of the Full Scheme

Recall that the full scheme is

$$\left(D_t^- D_t^+ U^k, \chi\right) + a(D_t^{\kappa,\upsilon}U^k, \chi) = (D_t^{\kappa,\upsilon}\eta(t_k), \chi) \text{ for any } \chi \in X_0^{N,p}(\Omega). \tag{4.2.7}$$

Writing the error as in (4.1.18), An estimation of the error in the full scheme is obtained by the following theorem.

Theorem 4.9. *Let $u(t_m)$ and U^m be the solution of the equation (4.2.1) and scheme (4.2.5) at time T respectively. Assume $u(t)$ and $\eta(t)$ satisfy the assumptions of Corollary 4.8. For $p = 2r+1$, $r \in \mathbb{N}_0$ and $\delta t \le 1$, we have*

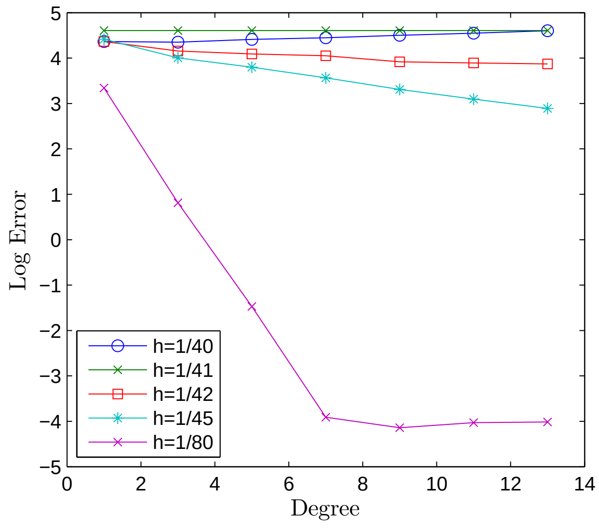
$$\|D_t^+ U^m - \dot{u}(t_{m+1/2})\| + \|U^{m+1/2} - u(t_{m+1/2})\| \le C\left(\left(\frac{h\sigma}{\pi}\right)^p + \delta t\right).$$

Proof. The proof is the same as that of Theorem 4.5, with the following modifications.

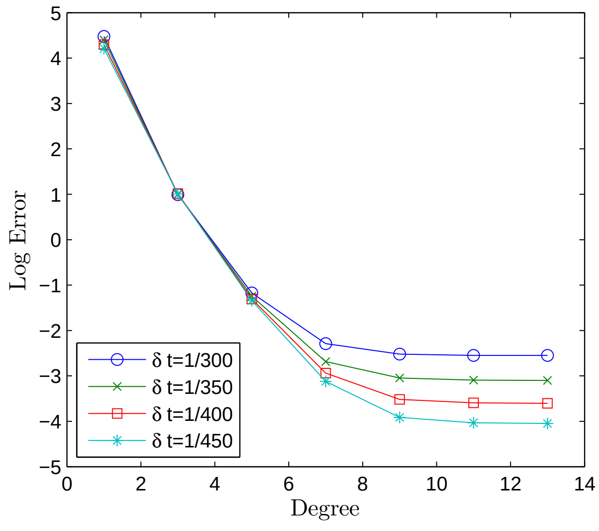
- Replace all the spaces $\mathcal{H}^{p+1}(\mathbb{T}^d)$ and $X^{N,p}(\mathbb{T}^d)$ by $\mathcal{H}_0^{p+1}(\Omega)$ and $X_0^{N,p}(\Omega)$, respectively.
- Replace all the operators $S_{h,p}$ by $\mathcal{B}_{h,p}$.
- Replace Lemma 3.3 and Corollary 2.15 by Lemma 3.12 and Corollary 2.20.
- Replace equations (4.1.17), (3.1.26), and (4.1.2) by (4.2.7), (3.2.8), and (4.2.2).
- Replace the estimate (4.1.12) by assumptions of $u(t)$ and (4.2.6).

$\square$

4.2.5 Numerical Experiment Results



(a) Convergence with different mesh size



(b) Convergence with different time-step size

Figure 4.4: Scheme for wave equation with Dirichlet condition

We take a similar test to that in Section 4.1.6.2 using $u(x,t) = \cos(t)\sin(41\pi x)$ as the true solution which is in the space $H^{p+1}([0,1])$ but not in $H^{p+1}(\mathbb{T})$. Figure 4.4a shows that the error converges exponentially when $h < 1/41$, but the convergence stops as degrees grow larger than 7. Figure 4.4b suggest this may be caused by the error of temporal approximation. The results agree with Corollary 4.8 and Theorem 4.9.

Chapter 5

Optimal Assembly Procedure of Bernstein-Bézier Spline Finite Elements

5.1 Introduction

In the previous chapters, we considered spline finite element methods for the problems whose coefficients are constant. In these cases, the stiffness matrices are able to be computed with explicit formulas, but for a problem with variable coefficients such as the elliptic equation shown in (5.1.8), since the integrals in the expressions are not always able to be evaluated analytically, explicit formulas may not be available. Numerical methods such as quadrature rules have to be applied for an approximation. A common procedure approximates the integrals locally on each element first to attain the element matrices and then assembly them to construct the global matrices.

However, in general, when the polynomial degree is high, the cost of evaluating the element matrices can be huge. For instance, the matrix corresponding to polynomials of degree n in d dimension contains n^{2d} entries, and each entry requires

$\mathcal{O}(n^d)$ operations since the quadrature rule adopts no less than $\mathcal{O}(n^d)$ quadrature points. In total, the construction of the element matrix will cost $\mathcal{O}(n^{3d})$ operations. Therefore, an algorithm for efficiently computation the element matrices is required.

The sum factorization method [65, 62] is generally applied to reduce the cost in spectral methods with high order polynomials [28, 51, 89]. [5] makes use of the method to evaluate the element matrices arising in FEM with tensor product B-splines as shape functions. In [2], the authors develop algorithms that evaluate the element matrices on simplicial elements in the optimal complexity $\mathcal{O}(n^{2d})$, in the sense that the number of operations is at the same order of the matrix entry number. In their algorithm, Bernstein-Bézier splines are employed as a basis facilitating the possibility of utilizing the sum factorization procedure, which is the main reason of the efficiency. Another advantage attributed to the basis is that the product of two Bernstein polynomials is still a Bernstein polynomial, which is exploited to simplify the evaluation of the element matrices to that of Bernstein polynomial moments.

Following their approach, our work aims to develop algorithms for constructing the element matrices in the finite element space of Bernstein-Bézier splines on quadrilaterals and hexahedrons in the optimal complexity $\mathcal{O}(n^{2d})$. The chapter is arranged as follows. The rest of the section reviews the Bernstein-Bézier spline shape functions and their corresponding finite element methods. In Section 5.2, we first develop the algorithm for evaluating the Bernstein polynomial moments, and then based on this, we derive the algorithm for constructing the element matrices. In Section 5.3, we show the results of some tests concerning the efficiency and accuracy of the algorithms. Note that only divisions and multiplications are counted as operations in our discussion.

5.1.1 Bernstein-Bézier Spline Shape Function on Quadrilaterals and Hexahedrons

In our work, the Bernstein-Bézier spline shape functions on quadrilaterals or hexahedrons are obtained by an isoparametric mapping of the tensor product Bernstein polynomials. Bernstein polynomials are one of the important types of polynomial representation. They have many useful properties, such as positivity, continuity, and unity of partition. There are many computational algorithms based on the representation, for instance, de Casteljau's algorithm [75, 55], which provides a stable way to evaluate a polynomial in Bernstein form. Because of the desirable properties, Bernstein polynomials play an important role in computer aided design with Bézier curves or surfaces [75]. They also have profound applications in differential equations and approximation theory [66, 10].

5.1.1.1 Tensor Product Bernstein Polynomials

The Bernstein polynomial of degree n is defined as

$$B_i^n(x) = \binom{n}{i} x^i (1-x)^{n-i}, \quad i = 0, \ldots, n.$$

We let $B_i^n(x) = 0$, when $i < 0$ or $i > n$. The polynomials are linearly independent [81], and so form a basis for the space of polynomial with a degree no larger than n. Some useful properties follow directly from the definition. The product of two Bernstein polynomials gives another one with higher degree,

$$\begin{aligned} B_i^n(x) B_j^m(x) &= \binom{n}{i}\binom{m}{j} x^{i+j}(1-x)^{n+m-i-j} \\ &= \binom{n}{i}\binom{m}{j} / \binom{n+m}{i+j} B_{i+j}^{n+m}(x), \\ &= \binom{i+j}{i}\binom{n+m-i-j}{n-i} / \binom{n+m}{n} B_{i+j}^{n+m}(x). \end{aligned} \tag{5.1.1}$$

The derivative of a Bernstein polynomial is a linear combination of lower degree ones,

$$\begin{aligned} (B_i^n(x))' &= \tfrac{n!}{(n-i)!(i-1)!} x^{i-1}(1-x)^{n-i} + \tfrac{n!}{(n-i-1)!(i)!} x^i (1-x)^{n-1-i} \\ &= n(B_{i-1}^{n-1}(x) - B_i^{n-1}(x)). \end{aligned} \tag{5.1.2}$$

The integral of a Bernstein polynomial $B_i^n(x)$ over [0,1] for each $i = 0, \ldots, n$ is

$$\int_0^1 B_i^n(x)\mathrm{d}x = 1/(n+1). \tag{5.1.3}$$

The definition of Bernstein polynomials is extended to higher dimension by tensor products. For $\boldsymbol{i} = (i_1, \ldots, i_d)$ and $\boldsymbol{n} = (n_1, \ldots, n_d)$,

$$B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{t}) = \prod_{1\le r\le d} B_{i_r}^{n_r}(t_r). \tag{5.1.4}$$

Let

$$\binom{\boldsymbol{n}}{\boldsymbol{i}} = \binom{n_1}{i_1}\cdots\binom{n_d}{i_d},$$

and $(\boldsymbol{e}^k)_j = \delta_{k,j}$, where $\delta_{k,j}$ is Kronecker delta. The multivariate polynomials $B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{t})$ have similar properties to (5.1.1), (5.1.2) and (5.1.3), which are

$$B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{t})B_{\boldsymbol{j}}^{\boldsymbol{m}}(\boldsymbol{t}) = \binom{\boldsymbol{i}+\boldsymbol{j}}{\boldsymbol{i}}\binom{\boldsymbol{n}+\boldsymbol{m}-\boldsymbol{i}-\boldsymbol{j}}{\boldsymbol{n}-\boldsymbol{i}}\Big/\binom{\boldsymbol{n}+\boldsymbol{m}}{\boldsymbol{n}}B_{\boldsymbol{i}+\boldsymbol{j}}^{\boldsymbol{n}+\boldsymbol{m}}(\boldsymbol{t}), \tag{5.1.5}$$

$$\frac{\partial}{\partial t_k}B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{t}) = n_k(-B_{\boldsymbol{i}}^{\boldsymbol{n}-\boldsymbol{e}^k}(\boldsymbol{t}) + B_{\boldsymbol{i}-\boldsymbol{e}^k}^{\boldsymbol{n}-\boldsymbol{e}^k}(\boldsymbol{t})), \tag{5.1.6}$$

and

$$\int_{[0,1]^d} B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \prod_{r=1}^{d} 1/(n_r+1). \tag{5.1.7}$$

5.1.1.2 Bernstein-Bézier Spline Shape Functions on Quadrilaterals

We first give an isogeometric mapping from the unit square $[0,1]^2$ to a quadrilateral. Given the four vertices of a quadrilateral $\{x_{ij}\}_{0\le i,j\le 1} \in \mathbb{R}^2$, let $B_0(t) = 1-t$ and $B_1(t) = t$ and define the mapping

$$\phi_D(\boldsymbol{t}) = \sum_{0\le i,j\le 1} x_{i,j}B_i(t_1)B_j(t_2).$$

We assume the four vertices are such that the quadrilateral is convex, which is equivalent to that the mapping $\phi_D(\boldsymbol{t})$ is bijective [32]. Then $D = \phi_D([0,1]^2)$ is the quadrilateral with the four vertices $x_{i,j} = \phi_D((i,j))$ for $0 \le i,j \le 1$. The Bernstein-Bézier shape functions on D are given by

$$\psi_{i,j}^{n,D}(\boldsymbol{x}) = B_{i,j}^{(n,n)}(\phi_D^{-1}(\boldsymbol{x})) \text{ for } \boldsymbol{x} \in D.$$

Denote by $\mathcal{P}_D^n$ the space spanned by all the shape functions $\psi_{i,j}^{n,D}$ for $i,j = 0,\ldots,n$.

5.1.1.3 Bernstein-Bézier Spline Shape Functions on Hexahedrons

We give the shape functions on a hexahedron in a similar manner to the quadrilateral case.

Given eight distinct points $\{x_{ijk}\}_{0\le i,j,k\le 1} \in \mathbb{R}^3$, define the mapping

$$\phi_H(\boldsymbol{t}) = \sum_{0\le i,j,k\le 1} x_{i,j,k} B_i(t_1) B_j(t_2) B_k(t_3).$$

If $\phi_H(\boldsymbol{t})$ is invertible on the cube $[0,1]^3$, then $H = \phi_H([0,1]^3)$ is called a hexahedron. The invertibility of the mapping $\phi_H(\boldsymbol{t})$ is not in the scope of the work, but the reader may check literature such as [94, 52] for the discussions about its invertibility conditions. The shape functions on H are given by

$$\psi^{n,H}_{i,j,k}(\boldsymbol{x}) = B^{(n,n,n)}_{i,j,k}(\phi_H^{-1}(\boldsymbol{x})) \text{ for } \boldsymbol{x} \in H \text{ and } i,j,k = 0,\dots,n.$$

Denote by $\mathcal{P}^n_H$ the space spanned by all the shape functions $\psi^{n,H}_{i,j,k}$ for $i,j,k = 0,\dots,n$. The faces of a hexahedron are not necessarily flat, where a face is flat provided their four vertices lay in a plane. A hexahedron with all faces flat is described as an ordinary hexahedron.

5.1.2 Bernstein-Bézier Spline Finite Element Method

Consider the elliptic equation

$$\begin{aligned} -\,\mathbf{div}(\boldsymbol{A}\nabla u) + bu &= \eta \text{ in } \Omega \\ u &= 0 \text{ on } \Gamma, \end{aligned} \tag{5.1.8}$$

where b is a continuous function and $\boldsymbol{A}$ is a $d \times d$ matrix-valued function which is continuous and positive definite on a polygonal domain $\Omega \subset \mathbb{R}^d$. Its weak formulation is given by finding $u \in H^1_0(\Omega)$ such that

$$\int_\Omega \nabla v \boldsymbol{A} \nabla u \mathrm{d}\boldsymbol{x} + \int_\Omega bvu \mathrm{d}\boldsymbol{x} = \int_\Omega \eta v \mathrm{d}\boldsymbol{x} \quad \text{for each } v \text{ in } H^1_0(\Omega).$$

Our work only considers the cases when $d = 2$ and $d = 3$. When $\Omega \subset \mathbb{R}^2$, the domain is discretized into quadrilaterals and when $\Omega \subset \mathbb{R}^3$, into hexahedrons.

For simplicity, we use K to stand for a quadrilateral or hexahedron, and $\mathcal{P}_K^n$ the corresponding shape functions space with basis $\psi_{\boldsymbol{i}}^{n,H}$. The discretization is denoted by $\Sigma = \{K_i\}_{i=1}^N$. Let $\mathcal{P}_\Sigma^n \subset H_0^1(\Omega)$ be the space such that $f|_{K_i} \in \mathcal{P}_{K_i}^n$. The solution u is approximated by u_h satisfying

$$\int_\Omega \nabla v \boldsymbol{A} \nabla u_h \mathrm{d}\boldsymbol{x} + \int_\Omega b v u_h \mathrm{d}\boldsymbol{x} = \int_\Omega \eta v \mathrm{d}\boldsymbol{x}, \quad \text{for each } v \text{ in } \mathcal{P}_\Sigma^n.$$

When applying the Galerkin method shown in Section 1.4.1, to obtain the global linear systems by assembling local element matrices, we need to compute for each element K the load vector $\boldsymbol{L}$ with components $\boldsymbol{L}_{\boldsymbol{i}} = \int_K \eta \phi_{\boldsymbol{i}}^{n,K} \mathrm{d}\boldsymbol{x}$, mass matrix $\boldsymbol{M}$ with components $\boldsymbol{M}_{\boldsymbol{i},\boldsymbol{j}} = \int_K b \phi_{\boldsymbol{i}}^{n,K} \phi_{\boldsymbol{j}}^{n,K} \mathrm{d}\boldsymbol{x}$ and stiffness matrix $\boldsymbol{S}$ with components $\boldsymbol{S}_{\boldsymbol{i},\boldsymbol{j}} = \int_K \nabla \phi_{\boldsymbol{i}}^{n,K} \boldsymbol{A} \nabla \phi_{\boldsymbol{j}}^{n,K} \mathrm{d}\boldsymbol{x}$.

The objective of the chapter is to derive algorithms for evaluating the mass matrix $\boldsymbol{M}$, stiffness matrix $\boldsymbol{S}$ and load vector $\boldsymbol{L}$ in optimal complexity. Note that the problem of assembling and solving the global linear system is not in the scope of the work, but standard preconditioning techniques [13, 39] are applicable for the problem.

5.2 Algorithms for Generating Element Matrices

In this section, we give algorithms for constructing the element matrices based on the Bernstein polynomial moments. Therefore, the first concern is to develop the algorithm for efficiently evaluating the moments.

5.2.1 Moment

The Bernstein polynomial moment of degree n is defined as

$$\mu_{\boldsymbol{i}}^{\boldsymbol{n}}(f, [0,1]^d) = \int_{[0,1]^d} B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{t}) f(\boldsymbol{t}) \mathrm{d}\boldsymbol{t}.$$

Using (5.1.4), the moment is rewritten as

$$\begin{aligned}\mu_{\boldsymbol{i}}^{\boldsymbol{n}}(f,[0,1]^d) = \;& \textstyle\int_0^1 \mathrm{d}t_d B_{i_d}^{n_d}(t_d)\\ & \times \textstyle\int_0^1 \mathrm{d}t_{d-1} B_{i_{d-1}}^{n_{d-1}}(t_{d-1})\\ & \cdots \\ & \times \textstyle\int_0^1 \mathrm{d}t_2 B_{i_2}^{n_2}(t_2)\\ & \times \textstyle\int_0^1 \mathrm{d}t_1 B_{i_1}^{n_1}(t_1) f(t_1,t_2,\ldots,t_d).\end{aligned} \tag{5.2.1}$$

If the function $f(\boldsymbol{x})$ is constant, the moment $\mu_{\boldsymbol{i}}^{\boldsymbol{n}}(1,[0,1]^d)$ is given directly by equation (5.1.7). However, in general, the integrals may not be calculated analytically, so quadrature rules are applied to obtain an approximation. We use the q-point Gaussian quadrature rule

$$\int_0^1 g(s)\mathrm{d}s \approx \sum_{j=1}^{q} w_j g(\varsigma_j),$$

where (w_j,ς_j) are the standard Gauss weights and nodes on the interval [0,1]. Applying the quadrature rule for each integral in (5.2.1) the moment $\mu_{\boldsymbol{i}}^{\boldsymbol{n}}(f,[0,1]^d)$ is approximated by

$$\begin{aligned}\hat{\mu}_{\boldsymbol{i}}^{\boldsymbol{n}}(f,[0,1]^d) = \;& \sum_{j_d=1}^{q} w_{j_d} B_{i_d}^{n_d}(\varsigma_{j_d})\\ & \times \sum_{j_{d-1}=1}^{q} w_{j_{d-1}} B_{i_{d-1}}^{n_{d-1}}(\varsigma_{j_{d-1}})\\ & \cdots \\ & \times \sum_{j_2=1}^{q} w_{j_2} B_{i_2}^{n_2}(\varsigma_{j_2})\\ & \times \sum_{j_1=1}^{q} w_{j_1} B_{i_1}^{n_1}(\varsigma_{j_1}) f(\varsigma_{j_1},\varsigma_{j_2},\ldots,\varsigma_{j_d}),\end{aligned}$$

where we choose $q \geq n+1$ with $n = \max\{n_l\}_{l=1,\ldots,d}$. The quadrature rule gives an exact integral for polynomials of degree no higher than $2q-1$. Equivalently, the moment $\hat{\mu}_{\boldsymbol{i}}^{\boldsymbol{n}}(f,[0,1]^d)$ is obtained by evaluating $\boldsymbol{F}^d(i_1,i_2\ldots,i_d)$ from the following recursion relations,

$$\begin{aligned}
\boldsymbol{F}^0(j_1, j_2, \ldots, j_d) &= f(\varsigma_{j_1}, \ldots, \varsigma_{j_d}) \\
\boldsymbol{F}^1(i_1, j_2 \ldots, j_d) &= \sum_{j_1=1}^{q} w_{j_1} B_{i_1}^{n_1}(\varsigma_{j_1}) \boldsymbol{F}^0(j_1, j_2, \ldots, j_d) \\
\boldsymbol{F}^2(i_1, i_2 \ldots, j_d) &= \sum_{j_2=1}^{q} w_{j_2} B_{i_2}^{n_2}(\varsigma_{j_2}) \boldsymbol{F}^1(i_1, j_2 \ldots, j_d) \\
&\ldots \\
\boldsymbol{F}^d(i_1, i_2 \ldots, i_d) &= \sum_{j_d=1}^{q} w_{j_d} B_{i_d}^{n_d}(\varsigma_{j_d}) \boldsymbol{F}^{d-1}(i_1, i_2 \ldots, i_{d-1}, j_d).
\end{aligned} \tag{5.2.2}$$

The relations suggest that the computation of $\boldsymbol{F}^l$ only relies on the information in $\boldsymbol{F}^{l-1}$ along with the value of the basis at quadrature points. In practical implementation, instead of generating individually the moment $\hat{\mu}_{\boldsymbol{i}}^{\boldsymbol{n}}(f, [0,1]^d)$ for each $\boldsymbol{i} \in \{0, \ldots, n\}^d$, a more efficient way generates all the entries of $\boldsymbol{F}^l$ each time based on $\boldsymbol{F}^{l-1}$ in succession for each $l = 1, \ldots, d$. This approach is described as the sum factorization procedure [65]. As for calculating the value of the basis at the quadrature points, one option is pre-computing and storing the values using the de Casteljau algorithm. Another approach, which also achieves the optimal complexity but does not require pre-computing, is shown in the following algorithms. Let $\mathcal{I}_k = \{0, 1, \ldots, n_k\}$ and $\mathcal{J} = \{0, 1, \ldots, q\}$. The following algorithms evaluate all the moments $\hat{\mu}_{\boldsymbol{i}}^{\boldsymbol{n}}(f, [0,1]^d)$ for each $0 \leq \boldsymbol{i} \leq \boldsymbol{n}$.

```
Algorithm 1: MOMENT($\boldsymbol{F}^0$,$\boldsymbol{n}$,$q$)
  Data: The array $\boldsymbol{F}^0$ stores the function values at the quadrature points
        with $\boldsymbol{F}^0_{\boldsymbol{j}} = f(\varsigma_{j_1}, \varsigma_{j_2}, \ldots, \varsigma_{j_d})$, $\boldsymbol{n}$ is the degree of the Bernstein
        polynomial, and $q$ is the order of Gaussian quadrature.
  Result: The array $\boldsymbol{F}^d$ stores the value of moments with $\boldsymbol{F}^d_{\boldsymbol{i}} = \hat{\mu}_{\boldsymbol{i}}^{\boldsymbol{n}}(f, [0,1]^d)$.
  for $l = 1, \ldots, d$ do
    $\boldsymbol{F}^l$ = MOMENTSTEP($\boldsymbol{F}^{l-1}, q, l$)
```

Theorem 5.1. *Given the values of function f at quadrature points and number of quadrature points is $q = n+1$, Algorithm 1 evaluates the moments $\hat{\mu}_{\boldsymbol{i}}^{\boldsymbol{n}}(f, [0,1]^d)$*

Algorithm 2: MomentStep($\boldsymbol{F}^{in}, q, l$)

```
1 for $j_l \in \mathcal{J}$ do
      $w = w_{j_l}, \varsigma = \varsigma_{j_l}, s = 1 - \varsigma, r = \varsigma/s, w = w * s^{n_l}$
2     for $i_l \in \mathcal{I}_l$ do
          for $(i_1, \ldots, i_{l-1}, j_{l+1}, \ldots, j_d) \in (\mathcal{I}_1, \ldots, \mathcal{I}_{l-1}, \mathcal{J}, \ldots, \mathcal{J})$ do
3             $\boldsymbol{F}^{out}_{i_1,\ldots,i_l,j_{l+1},\ldots,j_d} += w * \boldsymbol{F}^{in}_{i_1,\ldots,i_{l-1},j_l,\ldots,j_d}$
          $w* = (r * (n_l - i_l)/(n_l + 1))$
```

in $\mathcal{O}(n^{d+1})$ operations. When f is a polynomial of degree $\leq 2q - 1 - n$ in each variable, the moment $\hat{\mu}^{\boldsymbol{n}}_{\boldsymbol{i}}(f, [0,1]^d) = \mu^{\boldsymbol{n}}_{\boldsymbol{i}}(f, [0,1]^d)$.

Proof. Since the function MomentStep is invoked in Algorithm 1, we first investigate its cost of operations in Algorithm 2.

The statement causing the most complexity is in Line 3, since it is within the innermost loops. Compared with it the other statements' costs are negligible when n is large enough. Within the $d+1$ layers of loops conditioned by Line 1 and 2, the statement repeats totally $(n_1 + 1) \ldots (n_r + 1)q^{d-r+1}$ times in Algorithm 2, and so $\sum_{r=1}^{d} (n_1+1) \ldots (n_r+1)q^{d-r+1}$ in Algorithm 1. As $n = \max\{n_r\}_{r=1,\ldots,d}$ and $q = \mathcal{O}(n)$, the total operations for the statement are of order $\mathcal{O}(n^{d+1})$ in Algorithm 1.

□

5.2.2 Load Vector

The load vector $\boldsymbol{L}$ on $K \subset \mathbb{R}^d$ has entries

$$\boldsymbol{L}_{\boldsymbol{i}} = \int_K \psi^{n,K}_{\boldsymbol{i}}(\boldsymbol{x})\eta(\boldsymbol{x})\mathrm{d}\boldsymbol{x}.$$

Recall that K is a quadrilateral when $d = 2$ and a hexahedron when $d = 3$. Substituting the definition of the shape functions and changing the variable with

$\boldsymbol{x} = \phi_K(\boldsymbol{t})$, it follows that

$$\begin{aligned} \boldsymbol{L_i} &= \textstyle\int_K B_{\boldsymbol{i}}^{\boldsymbol{n}}(\phi_K^{-1}(\boldsymbol{x}))\eta(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &= \textstyle\int_{[0,1]^d} B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{t})\eta(\phi_K(\boldsymbol{t}))\det(J_{\phi_K}(\boldsymbol{t}))d\boldsymbol{t} \\ &= \mu_{\boldsymbol{i}}^{\boldsymbol{n}}(\eta(\phi_K(\boldsymbol{t}))\det(J_{\phi_K}(\boldsymbol{t})),[0,1]^d), \end{aligned} \tag{5.2.3}$$

where $J_{\phi_K}(\boldsymbol{t})$ is the Jacobi matrix of the function $\phi_K(\boldsymbol{t})$.

Hence, Algorithm 1 with Moment$(\boldsymbol{F}^K, \boldsymbol{n}, q)$ evaluates the load vectors, where $\boldsymbol{F}_{\boldsymbol{j}}^K = (\eta(\phi_K(\boldsymbol{t}))\det(J_{\phi_K}))|_{(\varsigma_{j_1},\ldots,\varsigma_{j_d})}$. Then the cost for constructing the load vector $\boldsymbol{L}$ is the same as for the Bernstein polynomial moments in $[0,1]^d$, which is $\mathcal{O}(n^{d+1})$.

5.2.3 Mass Matrix

The mass matrix on K has entries

$$\boldsymbol{M_{i,j}} = \int_K \psi_{\boldsymbol{i}}^{n,K}(\boldsymbol{x})\psi_{\boldsymbol{j}}^{n,K}(\boldsymbol{x})b(\boldsymbol{x})\mathrm{d}\boldsymbol{x}.$$

Substituting the definition of the shape function, changing the variable with $\boldsymbol{x} = \phi_K(\boldsymbol{t})$ and using property (5.1.5), it follows that

$$\begin{aligned} \boldsymbol{M_{i,j}} &= \textstyle\int_K B_{\boldsymbol{i}}^{\boldsymbol{n}}(\phi_K^{-1}(\boldsymbol{x}))B_{\boldsymbol{j}}^{\boldsymbol{n}}(\phi_K^{-1}(\boldsymbol{x}))b(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &= \textstyle\int_{[0,1]^d} B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{t})B_{\boldsymbol{j}}^{\boldsymbol{n}}(\boldsymbol{t})b(\phi_K(\boldsymbol{t}))\det(J_{\phi_K}(\boldsymbol{t}))\mathrm{d}\boldsymbol{t} \\ &= \binom{\boldsymbol{i}+\boldsymbol{j}}{\boldsymbol{i}}\binom{2\boldsymbol{n}-\boldsymbol{i}-\boldsymbol{j}}{\boldsymbol{n}-\boldsymbol{i}}/\binom{2\boldsymbol{n}}{\boldsymbol{n}}\mu_{\boldsymbol{i}+\boldsymbol{j}}^{2\boldsymbol{n}}(b(\phi_K)\det(J_{\phi_K}(\boldsymbol{t})),[0,1]^d). \end{aligned} \tag{5.2.4}$$

Let $\boldsymbol{\sigma}_{i,j} = \binom{i+j}{i}\binom{2n-i-j}{d-i}/\binom{2n}{n}$. Then we have the expression

$$\boldsymbol{M_{i,j}} = (\prod_{l=1}^{d}\boldsymbol{\sigma}_{i_l,j_l})\mu_{\boldsymbol{i}+\boldsymbol{j}}^{2\boldsymbol{n}}(b(\phi_K)\det(J_{\phi_K}),[0,1]^d).$$

The expression suggests that the evaluation of the binomial coefficients are required in the algorithm. We use the Pascal triangle method to compute the coefficients and store them in matrix $\boldsymbol{\kappa}$ with entry $\boldsymbol{\kappa}_{i,j} = \binom{i+j}{i}$. The following algorithm gives the mass matrix.

Algorithm 3: GetMass($\boldsymbol{F}^K, n, q$)

```
Data: The array $\boldsymbol{F}^K$ stores the values of the function $f^K = b(\phi_K)\det(J_{\phi_K})$
      at the quadrature points with $\boldsymbol{F}^K_{\boldsymbol{j}} = f^K(\varsigma_{j_1}, \varsigma_{j_2}, \ldots, \varsigma_{j_d})$.
Result: The matrix $\boldsymbol{M}$ is the mass matrix.
  for $(i,j) \in \mathcal{I} \times \mathcal{I}$ do
1     $\boldsymbol{\sigma}_{i,j} = \boldsymbol{\kappa}_{i,j} * \boldsymbol{\kappa}_{n-i,n-j} / \boldsymbol{\kappa}_{n,n}$
2 $\boldsymbol{L} = \text{Moment}(\boldsymbol{F}^K, 2\boldsymbol{n}, q)$
  for $(i_1, \ldots, i_d) \in \mathcal{I} \times \ldots \times \mathcal{I}$ do
      for $(j_1, \ldots, j_d) \in \mathcal{I} \times \ldots \times \mathcal{I}$ do
3         $\boldsymbol{M}_{i_1,\ldots,i_d,j_1,\ldots,j_d} = \boldsymbol{\sigma}_{i_1,j_1} \ldots \boldsymbol{\sigma}_{i_d,j_d} \boldsymbol{L}_{i_1+j_1,\ldots,i_d+j_d}$
```

Theorem 5.2. *Given the value of the function $f^K = b(\phi_K)\det(J_{\phi_K})$ at the quadrature nodes, the Algorithm* GetMass *constructs the mass matrix in $\mathcal{O}(n^{2d})$ operations for $d = 2, 3$.*

Proof. As $\mathcal{I} = \{1, \ldots, n\}$, the cost for statements in Line 1 and Line 3 are $\mathcal{O}(n^2)$ and $\mathcal{O}(n^{2d})$, respectively. According to Theorem 5.1, the statement for calculating the moments in Line 2 costs $\mathcal{O}((2n)^{d+1})$ operations. Hence, the total cost is $\mathcal{O}(n^2)+\mathcal{O}(n^{d+1})+\mathcal{O}((2n)^{2d}) = \mathcal{O}(n^{2d})$.

□

Note that the algorithm requires pre-computing and storing the matrix $\boldsymbol{\sigma}$ which takes $\mathcal{O}(n^2)$ storage. But the storage is negligible compared with the $\mathcal{O}(n^{2d})$ storage of matrix $\boldsymbol{M}$.

5.2.4 Stiffness Matrix

The stiffness matrix on $K \subset \mathbb{R}^d$ has entries

$$\boldsymbol{S}_{\boldsymbol{i},\boldsymbol{j}} = \int_K \nabla^t \psi^{n,K}_{\boldsymbol{i}}(\boldsymbol{x}) \boldsymbol{A}(\boldsymbol{x}) \nabla \psi^{n,K}_{\boldsymbol{j}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}.$$

Substituting the definition of the shape function, changing the variable with $\boldsymbol{x} = \phi_K(\boldsymbol{t})$ and converting the product into quadratic form gives

$$\begin{aligned}\boldsymbol{S}_{\boldsymbol{i},\boldsymbol{j}} &= \int_K \nabla^t B_{\boldsymbol{i}}^{\boldsymbol{n}}(\phi_K^{-1}(\boldsymbol{x}))\boldsymbol{A}(\boldsymbol{x})\nabla B_{\boldsymbol{j}}^{\boldsymbol{n}}(\phi_K^{-1}(\boldsymbol{x}))\mathrm{d}\boldsymbol{x}\\ &= \int_{[0,1]^d} \nabla^t B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{t})\boldsymbol{J}_{\phi_K}^{-1}\boldsymbol{A}(\phi_K(\boldsymbol{t}))\boldsymbol{J}_{\phi_K}^{-t}(\boldsymbol{t})\nabla B_{\boldsymbol{j}}^{\boldsymbol{n}}(\boldsymbol{t})\mathrm{d}\phi_K(\boldsymbol{t})\\ &= \int_{[0,1]^d} \nabla^t B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{t})\tilde{\boldsymbol{A}}(\boldsymbol{t})\nabla B_{\boldsymbol{j}}^{\boldsymbol{n}}(\boldsymbol{t})\mathrm{d}\boldsymbol{t}\\ &= \sum_{k=1}^{d}\sum_{l=1}^{d}\int_{[0,1]^d} \frac{\partial}{\partial t_k}B_{\boldsymbol{i}}^{\boldsymbol{n}}(\boldsymbol{t})\frac{\partial}{\partial t_l}B_{\boldsymbol{j}}^{\boldsymbol{n}}(\boldsymbol{t})\tilde{\boldsymbol{A}}_{k,l}(\boldsymbol{t})\mathrm{d}\boldsymbol{t},\end{aligned}$$

where $\tilde{\boldsymbol{A}}(\boldsymbol{t}) = \boldsymbol{J}_{\phi_K}^{-1}(\boldsymbol{t})\boldsymbol{A}(\phi_K(\boldsymbol{t}))\boldsymbol{J}_{\phi_K}^{-t}(\boldsymbol{t})\det(\boldsymbol{J}_{\phi_K}(\boldsymbol{t}))$. Using the property (5.1.5) and (5.1.6), it follows that

$$\begin{aligned}\boldsymbol{S}_{\boldsymbol{i},\boldsymbol{j}} &= d^2\sum_{k=1}^{d}\sum_{l=1}^{d}\int_{[0,1]^d}(-B_{\boldsymbol{i}}^{\boldsymbol{n}-\boldsymbol{e}_k}(\boldsymbol{t}) + B_{\boldsymbol{i}-\boldsymbol{e}_k}^{\boldsymbol{n}-\boldsymbol{e}_k}(\boldsymbol{t}))(-B_{\boldsymbol{j}}^{\boldsymbol{n}-\boldsymbol{e}_l}(\boldsymbol{t}) + B_{\boldsymbol{j}-\boldsymbol{e}_l}^{\boldsymbol{n}-\boldsymbol{e}_l}(\boldsymbol{t}))\tilde{\boldsymbol{A}}_{k,l}(\boldsymbol{t})\mathrm{d}\boldsymbol{t}\\ &= d^2\sum_{k=1}^{d}\sum_{l=1}^{d}\Bigg[\\ &\binom{\boldsymbol{i}-\boldsymbol{e}_k+\boldsymbol{j}-\boldsymbol{e}_l}{\boldsymbol{i}-\boldsymbol{e}_k}\binom{2\boldsymbol{n}-\boldsymbol{i}-\boldsymbol{j}}{\boldsymbol{n}-\boldsymbol{i}}\Big/\binom{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}{\boldsymbol{n}-\boldsymbol{e}_k}\mu_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_k-\boldsymbol{e}_l}^{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}(\tilde{\boldsymbol{A}}_{k,l},[0,1]^d)\\ &-\binom{\boldsymbol{i}-\boldsymbol{e}_k+\boldsymbol{j}}{\boldsymbol{i}-\boldsymbol{e}_k}\binom{2\boldsymbol{n}-\boldsymbol{i}-\boldsymbol{j}-\boldsymbol{e}_l}{\boldsymbol{n}-\boldsymbol{i}}\Big/\binom{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}{\boldsymbol{n}-\boldsymbol{e}_k}\mu_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_k}^{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}(\tilde{\boldsymbol{A}}_{k,l},[0,1]^d)\\ &-\binom{\boldsymbol{i}-\boldsymbol{e}_l+\boldsymbol{j}}{\boldsymbol{i}}\binom{2\boldsymbol{n}-\boldsymbol{i}-\boldsymbol{j}-\boldsymbol{e}_k}{\boldsymbol{n}-\boldsymbol{i}-\boldsymbol{e}_k}\Big/\binom{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}{\boldsymbol{n}-\boldsymbol{e}_k}\mu_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_l}^{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}(\tilde{\boldsymbol{A}}_{k,l},[0,1]^d)\\ &+\binom{\boldsymbol{i}+\boldsymbol{j}}{\boldsymbol{i}}\binom{2\boldsymbol{n}-\boldsymbol{i}-\boldsymbol{j}-\boldsymbol{e}_k-\boldsymbol{e}_l}{\boldsymbol{n}-\boldsymbol{i}-\boldsymbol{e}_k}\Big/\binom{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}{\boldsymbol{n}-\boldsymbol{e}_k}\mu_{\boldsymbol{i}+\boldsymbol{j}}^{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}(\tilde{\boldsymbol{A}}_{k,l},[0,1]^d)\Bigg].\end{aligned}$$

As shown in [3], for the purpose of efficient implementation the expression can be rewritten as

$$\begin{aligned}\boldsymbol{S}_{\boldsymbol{i},\boldsymbol{j}} = 2n\prod_{r=1}^{d}\boldsymbol{\sigma}_{i_r,j_r}\sum_{k=1}^{d}\sum_{l=1}^{d}[\alpha_{k,l}\mu_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_k-\boldsymbol{e}_l}^{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}(\tilde{\boldsymbol{A}}_{k,l},[0,1]^d)\\ -\beta_{k,l}\mu_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_k}^{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}(\tilde{\boldsymbol{A}}_{k,l},[0,1]^d) - \gamma_{k,l}\mu_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_l}^{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}(\tilde{\boldsymbol{A}}_{k,l},[0,1]^d) + \delta_{k,l}\mu_{\boldsymbol{i}+\boldsymbol{j}}^{2\boldsymbol{n}-\boldsymbol{e}_k-\boldsymbol{e}_l}(\tilde{\boldsymbol{A}}_{k,l},[0,1]^d)],\end{aligned} \tag{5.2.5}$$

where the coefficients are given as

$$\begin{aligned}\alpha_{k,l} &= \tfrac{2ni_kj_l}{(i_k+j_k)(i_l+j_l)} & \beta_{k,l} &= \tfrac{2ni_k(n-j_l)}{(i_k+j_k)(2n-i_l-j_l)}\\ \gamma_{k,l} &= \tfrac{2n(n-i_k)j_l}{(2n-i_k-j_k)(i_l+j_l)} & \delta_{k,l} &= \tfrac{2n(n-i_k)(n-j_l)}{(2n-i_k-j_k)(2n-i_l-j_l)},\end{aligned}$$

if $k \neq l$, and

$$\begin{aligned}\alpha_{k,k} &= \tfrac{(2n-1)i_kj_k}{(i_k+j_k)(i_k+j_k-1)} & \beta_{k,k} &= \tfrac{(2n-1)i_k(n-j_k)}{(i_k+j_k)(2n-i_k-j_k)}\\ \gamma_{k,k} &= \tfrac{(2n-1)(n-i_k)j_k}{(2n-i_k-j_k)(i_k+j_k)} & \delta_{k,k} &= \tfrac{(2n-1)(n-i_k)(n-j_k)}{(2n-i_k-j_k)(2n-i_k-j_k-1)}.\end{aligned}$$

The algorithm for evaluating the stiffness matrix is given as follows.

```
Algorithm 4: GetStiff($\boldsymbol{F}^K$)
Data: The array $\boldsymbol{F}^K$ stores the values of the vector valued function $\tilde{\boldsymbol{A}}$ at
      the quadrature points with ${\boldsymbol{F}^K}_{k,l;\boldsymbol{j}} = \tilde{\boldsymbol{A}}_{k,l}(\varsigma_{\boldsymbol{j}})$.
Result: The matrix $\boldsymbol{S}$ is the stiffness matrix.
for $(i,j) \in \mathcal{I} \times \mathcal{I}$ do
1   $\boldsymbol{\sigma}(i,j) = \boldsymbol{\kappa}(i,j) * \boldsymbol{\kappa}(n-i,n-j)/\boldsymbol{\kappa}(n,n)$
for $(k,l) \in \{1,\dots,d\} \times \{1,\dots,d\}$ do
2   $\boldsymbol{L} = \text{Moment}(\boldsymbol{F}^K_{k,l}, 2\boldsymbol{n} - \boldsymbol{e}_k - \boldsymbol{e}_l, q)$
    if $k \neq l$ then
        for $\boldsymbol{i} = (i_1,\dots,i_d) \in \mathcal{I} \times \dots \times \mathcal{I}$ do
            for $\boldsymbol{j} = (j_1,\dots,j_d) \in \mathcal{I} \times \dots \times \mathcal{I}$ do
                $\alpha = 2 * n * i_k * j_l/(i_k + j_k)/(i_l + j_l)$
                $\beta = 2 * n * i_k * (n - j_l)/(i_k + j_k)/(2n - i_l - j_l)$
                $\gamma = 2 * n * (n - i_k) j_l/(2n - i_k - j_k)/(i_l + j_l)$
                $\delta = 2 * n * (n - i_k) * (n - j_l)/(2n - i_k - j_k)/(2n - i_l - j_l)$
                $\boldsymbol{S}_{\boldsymbol{i},\boldsymbol{j}} = \boldsymbol{S}_{\boldsymbol{i},\boldsymbol{j}} + 2 * n * \boldsymbol{\sigma}_{i_1,j_1} \dots \boldsymbol{\sigma}_{i_d,j_d} * [\alpha * \boldsymbol{L}_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_k-\boldsymbol{e}_l} - \beta *$
                $\boldsymbol{L}_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_k} - \gamma * \boldsymbol{L}_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_l} + \delta * \boldsymbol{L}_{\boldsymbol{i}+\boldsymbol{j}}]$
    else
        for $(i_1,\dots,i_d) \in \mathcal{I} \times \dots \times \mathcal{I}$ do
            for $(j_1,\dots,j_d) \in \mathcal{I} \times \dots \times \mathcal{I}$ do
                $\alpha = (2 * n - 1) * i_k * j_l/(i_k + j_k)/(i_l + j_l - 1)$
                $\beta = (2 * n - 1) * i_k * (n - j_l)/(i_k + j_k)/(2n - i_l - j_l)$
                $\gamma = (2 * n - 1) * (n - i_k) j_l/(2n - i_k - j_k)/(i_l + j_l)$
                $\delta = (2 * n - 1) * (n - i_k) * (n - j_l)/(2n - i_k - j_k)/(2n - i_l - j_l - 1)$
                $\boldsymbol{S}_{\boldsymbol{i},\boldsymbol{j}} = \boldsymbol{S}_{\boldsymbol{i},\boldsymbol{j}} + 2 * n * \boldsymbol{\sigma}_{i_1,j_1} \dots \boldsymbol{\sigma}_{i_d,j_d} * [\alpha * \boldsymbol{L}_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_k-\boldsymbol{e}_l} - \beta *$
                $\boldsymbol{L}_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_k} - \gamma * \boldsymbol{L}_{\boldsymbol{i}+\boldsymbol{j}-\boldsymbol{e}_l} + \delta * \boldsymbol{L}_{\boldsymbol{i}+\boldsymbol{j}}]$
```

Theorem 5.3. *Given the value of the vector-valued function* $\tilde{\boldsymbol{A}}$ *at quadrature* *points, Algorithm 4 generate the stiffness matrix in $\mathcal{O}(n^{2d})$ operations for $d = 2, 3$.*

Proof. The statements in Line 1 and 2 cost $\mathcal{O}(n^2)$ and $\mathcal{O}(n^{d+1})$ operations, respectively. For the case when $k \neq l$, there are totally q^{2d} layers of loops and each of the operations in the innermost loops cost $\mathcal{O}(1)$. Since $q = \mathcal{O}(n)$, the cost is $\mathcal{O}(n^{2d})$. For the case when $k = l$, a similar analysis gives the same estimation. Hence, the overall cost of operations is $\mathcal{O}(n^{2d})$.

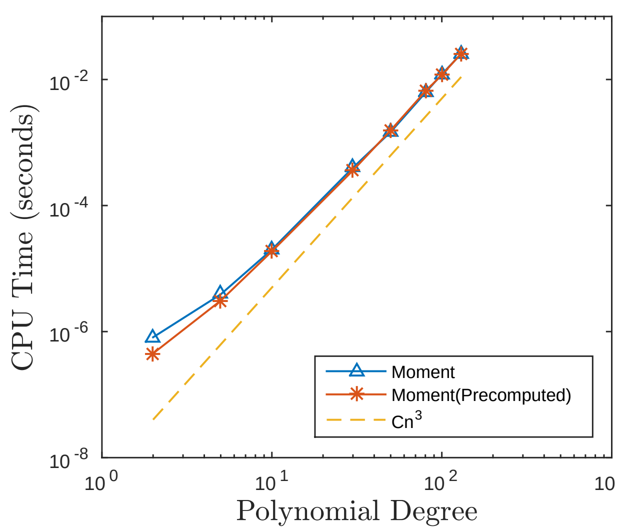
□

5.3 Experiments Results

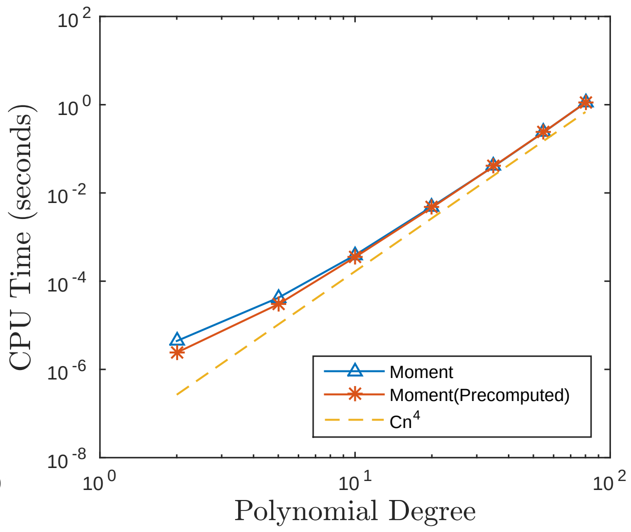
5.3.1 Computation Times

In this part, we test the cost of the algorithms by comparing their CPU computation times. The algorithms are coded in C++, which follow the ideas of the Library BBFEM [1] and are contributed as a part of the Library. Samples of the code are given in Appendix A.2. The computations are performed on a Dell Optiplex 790 workstation with Intel Core i5-2400 3.10GHz processor and 7.6 GB RAM using C++ and the gcc compiler. Each of the following tests is conducted for both quadrilateral and hexahedron cases.

5.3.1.1 With/Without Precomputed Data



(a) Quadrilateral



(b) Hexahedron

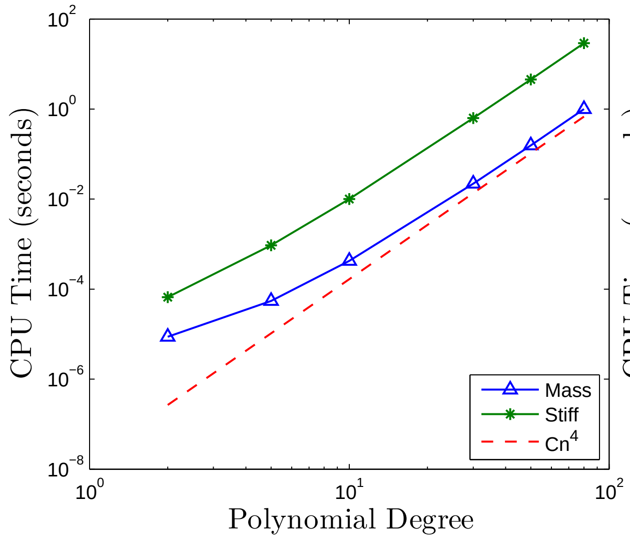
Figure 5.1: CPU computation time for constructing the moments with or without precomputed values of the basis at quadrature nodes.

The test compares the of cost of Algorithm 1 and the algorithm using the precomputed basis values at the quadrature points as discussed in Section 5.2.1.

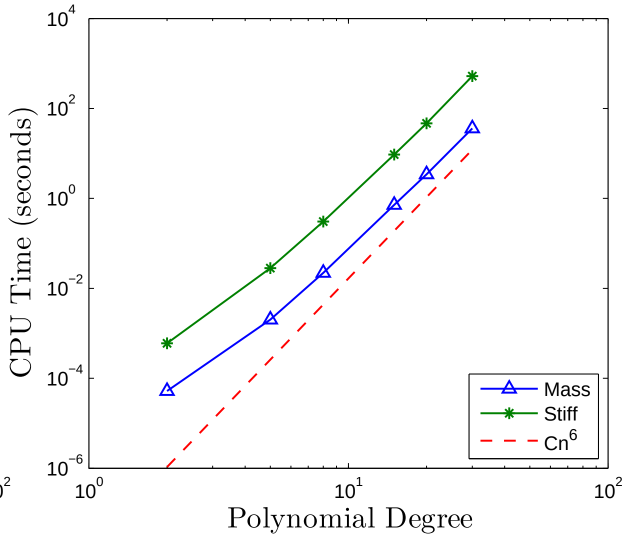
We do the log-log plot of the CPU computation times against the degrees of Bernstein polynomials n for the two algorithms. A curve of Cn^{d+1} is also included for comparison, where $d = 2$ corresponds to quadrilateral case and $d = 3$ to hexahedron case.

Figure 5.1 shows the results for both quadrilateral and hexahedron cases. For small n, the algorithm without precomputed values takes lower computation times, but when n is large the two lines overlap and increase at rate of Cn^{d+1}, which means the difference between costs is negligible for high degree Bernstein moments.

5.3.1.2 Mass and Stiffness Matrices



(a) Quadrilateral



(b) Hexahedron

Figure 5.2: CPU computation time for constructing mass and stiffness matrix.

The test aims at checking the cost of generating mass and stiffness matrices using Algorithm 3 and 4, respectively.

Analogously to the previous test, the CPU computation time is plotted against Bernstein polynomial degree n. As shown in Figure 5.2, with the growth of the degree n, the two computation times increase at the same rate of Cn^{2d}, although more time for the stiffness matrix case is required than the mass case. Hence, algorithms for both stiffness and mass matrices achieve the optimality complexity $\mathcal{O}(n^{2d})$.

5.3.2 Errors of BBFEM Approximation

These experiments check whether the load vector, mass matrix and stiffness matrix in Algorithm 1, 3, and 4, are correctly constructed, by solving specific problems of the form (5.1.8) using Bernstein-Bézier Spline FEM, and observing the behaviour of the error with the growth of the polynomial degree n. The relative error is

Table 5.1: Error of FEM on quadrilateral

	Quadrilateral		Ordinary Hexahedron		Non-Ordinary Hex.	
n	Error	Cond. No.	Error	Cond. No.	Error	Cond. No.
2	8.8×10^{-1}	1	2.1	1	5.5×10^{-2}	1
3	6.1×10^{-1}	1×10	3.6×10^{-1}	3.6×10	2.4×10^{-2}	8.7×10
4	2.4×10^{-1}	1.4×10^{2}	6.1×10^{-2}	3.2×10^{3}	6.2×10^{-3}	6.1×10^{4}
5	2.4×10^{-2}	1.1×10^{3}	2.0×10^{-3}	3.7×10^{5}	1.0×10^{-4}	3.9×10^{6}
6	3.2×10^{-3}	1.6×10^{4}	9.1×10^{-4}	2.9×10^{7}	2.1×10^{-4}	2.6×10^{8}
7	9.3×10^{-4}	2.4×10^{5}	7.3×10^{-6}	2.5×10^{9}	1.1×10^{-6}	1.6×10^{10}
8	9.5×10^{-5}	1.1×10^{7}	2.3×10^{-7}	1.4×10^{11}	8.8×10^{-7}	1.0×10^{12}
9	3.2×10^{-6}	8.6×10^{7}	6.3×10^{-9}	1.2×10^{13}	9.6×10^{-9}	5.9×10^{13}
10	4.0×10^{-8}	5.3×10^{9}	8.7×10^{-10}	7.7×10^{14}	1.6×10^{-9}	5.3×10^{15}
11	3.3×10^{-9}	2.9×10^{10}	1.0×10^{-8}	4.2×10^{16}	3.1×10^{-9}	2.0×10^{17}
12	3.1×10^{-9}	1.4×10^{12}	2.7×10^{-8}	4.8×10^{17}	7.1×10^{-10}	3.3×10^{17}
13	4.5×10^{-8}	8.2×10^{11}	5.4×10^{-8}	1.4×10^{17}	3.0×10^{-8}	2.1×10^{18}
14	3.1×10^{-7}	7.1×10^{13}	1.2×10^{-8}	9.8×10^{17}	3.9×10^{-9}	3.1×10^{18}
15	7.2×10^{-7}	1.9×10^{15}	1.1×10^{-5}	1.4×10^{19}	3.2×10^{-8}	8.1×10^{18}

given by $\|\vec{u} - \vec{u}_h\|_2/\|\vec{u}\|_2$, where $\vec{u}$ is the value of true solution at certain points and $\vec{u}_h$ is the vector of the approximations values at the corresponding points. For simplicity, the problem is studied on a single element domain with Dirichlet boundary conditions. When $d = 2$, we choose the domain to be a quadrilateral, and when $d = 3$, we are interested in two cases where the domain is ordinary and non-ordinary hexahedrons, where non-ordinary hexahedrons are hexahedrons with non-flat faces as explained in Section 5.1.1.3. The linear systems arising in FEM are solved using LU factorization method in C++ package GMM.

Table 5.2: Vertices of the domains

Quadrilateral	Ordinary Hexahedron	Non-Ordinary Hex.
(2,2)	$(0, 0, 0), (\frac{10}{20}, \frac{40}{20}, \frac{-70}{20})$	(0, 0, 0), (1, 0, 0)
(5,1)	$(\frac{13}{20}, \frac{39}{20}, \frac{-61}{20}), (\frac{3}{20}, \frac{-1}{20}, \frac{9}{20})$	(1, 1, 0), (0, 1, 0)
(3,4)	$(\frac{-3}{20}, \frac{6}{20}, \frac{9}{20}), (\frac{-2}{20}, \frac{10}{20}, \frac{2}{20})$	(0, 0, 5), (1, 0, 1)
(4,5)	$(\frac{1}{20}, \frac{9}{20}, \frac{11}{20}), (0, \frac{5}{20}, \frac{18}{20})$	(1,1, 1), (0, 1, 1)

Table 5.2 shows the vertices of the three domains. The relative errors of the approximation using the methods with various degrees of the splines are listed in Table 5.1. The condition numbers of the linear system involved in the FEM for each case are also included. It is observed from the table that the error converges when the degree n is lower than 10, but as n grows higher, the error starts to diverge. On the other hand, the condition numbers are very large at this phase and keeps increasing as n raises. This suggests the divergence of the error is caused by the inaccuracy when solving the linear system in FEM caused by high condition numbers. Hence, the error behaviours are consistent with the assumption that the element matrices are constructed correctly.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

We began by considering B-spline interpolation of smooth functions. To study the performance of the approximation as the degree and smoothness grow, we derived an error estimate measured in a semi-norm which is explicit in the mesh size h, degree p and order of semi-norm k. Making use of the result of [34], which is an estimate in supremum norm of cardinal spline interpolation, we derived an estimate for periodic functions in the L_2 norm in Theorem 2.4. Using the Kolmogorov inequality and the best approximation property derived in Lemma 2.5, the estimation was extended to the semi-norm case in Theorem 2.6. This result then implies an estimate for tensor product B-spline interpolation of multivariate functions in Theorem 2.14. Finally, in Theorem 2.17 and Theorem 2.19, we removed the periodic constraint, and generated estimates for smooth functions in both univariate and multivariate cases. As shown in the Corollary 2.15 and 2.20, the error converges algebraically in term of the mesh size h, and for function satisfying the assumptions of the corollary the error converges exponentially. To confirm the sharpness of the convergence rate in our estimates, we checked the convergence numerically.

Equipped with the results for B-spline interpolation, we studied B-spline FEM approximation in the setting of heat equations and wave equations with each of the two kinds of boundary conditions (periodic and Dirichlet). As in our study of interpolation, we were interested in accuracy when the degree of the B-spline increases, which is referred to as k-refinement in some literature [45]. Following the standard approach of error analysis of Galerkin methods in [87, 56], we derived error estimates of the semi-discrete schemes for the heat equation in Theorem 3.5 and 3.14, and estimates of the full scheme in Theorem 3.11 and 3.17 for periodic and Dirichlet cases, respectively. For the wave equation, we gave the estimates of the semi-discrete schemes in Theorem 4.3 and 4.7, and the full scheme in Theorem 4.5 and 4.9 for periodic and Dirichlet cases, respectively. Under certain assumptions about the equation, as the degree of B-spline increases, the approximation error of spatial variable is shown to converge exponentially in Corollary 3.6 and 3.15 for the heat equation, and in Corollary 4.4 and 4.8 for the wave equation, respectively. In all the cases, numerical results were found to agree with the theory.

Furthermore, in the periodic case, where periodic B-splines are employed as a basis, we showed that the scheme can be implemented efficiently using the fast Fourier transform for each time step in $\mathcal{O}(dN^d log(N))$. We studied the stability of the schemes in this case, which led us to prove a Markov-type inequality for periodic B-spline in Theorem 3.7. The inequality gives the sharpest coefficient when there are even mesh numbers, as shown in Lemma 3.8. In the odd case, we give an expression which is conjectured to be the sharpest coefficient. Numerical experiments supported this conjuncture. Our analysis implies that as the degree increases, the coefficient decreases and converges to a constant. We conclude that when the degree increases the stability does not deteriorate, but gets slightly better.

We developed algorithms for constructing the element matrices in the finite element space of Bernstein-Bézier splines on quadrilaterals and hexahedrons in the optimal complexity $\mathcal{O}(n^{2d})$. Algorithms for evaluating Bernstein polynomial moments are developed in Algorithm 1 by making use of the sum factorization procedure based on the tensor product structure of the basis, which costs $\mathcal{O}(n^{d+1})$ operations. Relying on the relation of the element matrices with the moment, the algorithms for evaluating the element matrices $\boldsymbol{M}$, $\boldsymbol{S}$, and $\boldsymbol{L}$ in the Bernstein-Bézier spline finite element method were developed in Algorithm 3 and 4. The cost of these two algorithms achieves the optimal complexity $\mathcal{O}(n^{2d})$. The load vector is evaluated directly by Algorithm 1 with the cost of $\mathcal{O}(n^{d+1})$. The algorithms were programmed in C++ where the code become a part of the C++ Library bbfem [1]. The result of numerical simulation in benchmark problems agreed with the predicted accuracy of the finite elements. The tests considering the cost of the algorithm also showed that optimality is achieved.

6.2 Future Work

Some further works are list as follows:

- All the estimates for the B-spline approximation require the degree to be odd. The even degree case is still due to be studied.
- The B-splines considered are defined on a uniform mesh. It would be worthwhile to generalize to non-uniform meshes.
- We considered the spline of maximum smoothness, that is, the B-spine of degree p and with smoothness C^{p-1}. It would be interesting to explore the case where the smoothness is $C^{p-s}, 1 \le s \le p$, although the case when $2s \le p$ is already studied in [22].
- We only investigated the periodic case of the stability of the scheme, but the non-periodic case is also interesting to try.
- For the algorithms of Bernstein-Bézier splines finite elements methods, as shown in Table 5.1, the numerical method gets less accurate for high degrees because of the high condition number of the linear system. To fully make

use of the optimal ratio for a high degree, it would be beneficial to study preconditioning techniques for solvers of the system.

Appendix A

Programming Codes

The appendix lists a small sample of the codes that were used in the computational experiments.

A.1 B-spline FEM for the Heat Equation(Periodic Case)

In this section we include some sample codes in Matlab for B-spline FEM.

A.1.1 Value Evaluation of Spline Function

Given a degree, knots vector and coefficients of a spline function express as a B-spline, the following function evaluates its value at a point.

```
function [opt] =curvepoint(p,knots,cof,x,nok)
loc=locspan(x,knots,nok);
N=allBasis(x,p,knots,loc);
opt=0;
for i=0:p,
    opt=opt+N(i+1)*cof(loc-p+i);
end
```

```
end
```

The following function finds the interval where the point of interest is located.

```
function [pos] = locspan(x,knots,Nok)
high=Nok;
low=1;
mid=floor((high+low)/2);
while (x<knots(mid)||x>=knots(mid+1))
    if x<knots(mid),
        high=mid;
    else
        low=mid+1;
    end
    mid=floor((high+low)/2);
end
pos=mid;
end
```

The following function gives values of all the B-splines whose supports include the sub-interval.

```
function [N] = allBasis(x,p,knots,loc )
N=1:(p+1);
left=1:(p+1);
right=1:(p+1);
N(1+0)=1;
for j=1:p,
     left(j)=x-knots(loc-j+1);
     right(j)=knots(loc+j)-x;
     saved=0;
     for r=0:j-1,
     temp=N(r+1)/(right(r+1)+left(j-r));
     N(r+1)=saved+right(r+1)*temp;
     saved=left(j-r)*temp;
     end
     N(j+1)=saved;
end
end
```

The following function evaluates the value of a periodic spline.

```
function [Y]= PerSplnVec(p,Nod,cof,X)
%Evaluate values of periodic spline.
NoK=length(Nod);
Lft=Nod(1);
Rht=Nod(end);
cofhat=zeros(1,NoK+p-1);
for i=1:p,
      cofhat(i)=cof(NoK-p+i-1);
end
for i=1:NoK-1,
      cofhat(i+p)=cof(i);
end
msize=(Rht-Lft)/(NoK-1);
Nodhat=zeros(1,NoK+2*p);
Nodhat(p+1:p+NoK)=Nod;
for i=1:p,
   Nodhat(i)=Lft-p*msize+(i-1)*msize;
   Nodhat(NoK+p+i)=Rht+i*msize;
end
Nodhat(p+1:p+NoK)=Nod;
```

```
NoK=NoK+2*p;
Y=zeros(size(X));
for i=1:length(X)
    x=X(i);
    if(X(i)==Rht)
        x=Lft;
    end
    Y(i)=curvepoint(p,Nodhat,cofhat,x,NoK);
end
end
```

A.1.2 Interpolation

The following function gives the coefficients for interpolation.

```
function [ teA,Vcoef,IntPnt ] = GivePerIntp1D...
    ( p,lenth,msize,Nod,Fun)
%%The function give the coef. of Periodic inteprolation
%% The value of interpolated function.
IntPnt=zeros(lenth,1);
Pnt=zeros(lenth,1);
if mod(p+1,2)==0,
    for i=1:lenth,
        IntPnt(i)=Nod(i);
    end
else
    for i=1:lenth,
        IntPnt(i)=Nod(i)+msize/2;
    end
end
```

```
for i=1:lenth,
    Pnt(i)=Fun(IntPnt(i));
end
%% The matrix.
Cof=zeros(1,lenth);
Cof(1)=1;
Vcoef=PerSplnVec(p,Nod,Cof,IntPnt);
%% Solve the linear system
teA=ifft(fft(Pnt)./fft(Vcoef));
end
```

A.1.3 Mass Matrix

This function returns the mass matrix of the periodic B-spline scheme.

```
function [ Vmass ] = MassMtxPer1D( p, Nod, msize, lenth )
%Give the mass matrix.
Cof=zeros(1,lenth);
Cof(lenth-p)=1;
IntPnt=Nod(1:end-1);
IntPnt=reshape(IntPnt, [length(IntPnt),1]);
Vmass=msize*PerSplnVec(2*p+1, Nod, Cof, IntPnt);
end
```

A.1.4 Stiffness Matrix

This function returns the stiffness matrix of the periodic B-spline scheme.

```
function [ Vstiff ]=StiffMtxPer1D( p, Nod, msize,lenth )
%Give the stiffness matrix
Cof=zeros(1,lenth);
```

```
Cof(lenth-p+1)=1;
IntPnt=Nod(1:end-1);
IntPnt=reshape(IntPnt, [length(IntPnt),1]);
Vstiff=2*PerSplnVec(2*p-1, Nod, Cof, IntPnt);
Cof=zeros(1,lenth);
Cof(lenth-p)=1;
Vstiff=Vstiff-PerSplnVec(2*p-1, Nod, Cof, IntPnt);
Cof=zeros(1,lenth);
Cof(lenth-p+2)=1;
Vstiff=Vstiff-PerSplnVec(2*p-1, Nod, Cof, IntPnt);
Vstiff=Vstiff/msize;
end
```

A.1.5 Full Scheme

This following routine implements the B-spline finite element method for solving the heat equation.

```
clear
clc
Lft=0;Rht=1;
p=5;NoK=61;NoE=NoK-1;
TimeEnd=0.01;TimeNo=800;
theta=0.9; %The theta in theta scheme.
FunInt=@(x) FunTruSoln(x, 0);
%%FunInt is the function of initial value .
%FunTruSoln is the function of true solution.
Frhs=@FRhsHt;
%%FRhsHt is the function of Nonhomogeneous term.
%% Give nodes
```

```
MeshSize=(Rht-Lft)/NoE;
Nod=zeros(1,NoK);
for i=1:NoK,
    Nod(i)=Lft+(i-1)*MeshSize;
end
%% Construct the matrices
[teA,Vcoef,IntPnt]=GivePerIntp1D( p,NoE,MeshSize,Nod,FunInt);
Vmass=MassMtxPer1D( p, Nod, MeshSize,NoE);
Vstiff=StiffMtxPer1D( p,Nod, MeshSize, NoE );
%% Apply the theta scheme
Alph=ThetеSchemePer( theta, teA,  TimeEnd, TimeNo, ...
    Vmass, Vstiff, Vcoef, IntPnt, Frhs );
%% Do plots and calculate error
Npts=7513;
Px=linspace(Lft,Rht,Npts);
Py=PerSplnVec(p,Nod,Alph,Px);
Pz=arrayfun(@(x)FunTruSoln(x,TimeEnd),Px);
plot(Px,Py,Px,Pz,'--r')
Err=norm(Py-Pz,2)
RelErr = Err/norm(Pz)*100
```

A.2 Bernstein-Bézier Spline Finite Elements (Hexahedron case)

The section gives examples of C++ codes for evaluating Bernstein-Bézier element matrices on hexahedrons, as described in Section 5.2. For other functions in the Library, the reader may see [1].

A.2.1 Moments

The following function gives the Bernstein polynomial moments stored in `MLoad`. `Degx`, `Degy`, and `Degz` specify the degree of the polynomial. `Num_Abs`, `Abs`, and `Wgt` give the number of quadrature nodes, the quadrature nodes and the corresponding weights. `Val_Abs` assigns the value of the function f in Section 5.2.1 at the quadrature nodes.

```
void
Get_TP_moment(int Degx, int Degy, int Degz, double ***MLoad,
int Num_Abs,  double ***Val_Abs, double *Abs, double *Wgt)
{
double  ***MF, ***MS, TmpAbs, TmpWgt, TmpRat;
// Create a 3D array.
MF=crt_ary3(Num_Abs+1, Num_Abs+1, Degz+1);
for (int Jx = 1; Jx<= Num_Abs; Jx++)
{
for (int Jy = 1; Jy<= Num_Abs; Jy++)
{
for (int Iz = 0; Iz<= Degz; Iz++)
{
MF[Jx][Jy][Iz]=0;
}
}
}
//Evaluate F^1 in (5.2.3)
for (int Jz = 1; Jz<= Num_Abs; Jz++)
{
TmpAbs=Abs[Jz];
TmpWgt=Wgt[Jz];
TmpWgt=TmpWgt*pow(1-TmpAbs,Degz);
TmpRat=TmpAbs/(1-TmpAbs);
for (int Iz = 0; Iz<= Degz; Iz++)
{
for (int Jx = 1; Jx<= Num_Abs; Jx++)
{
for (int Jy = 1; Jy<= Num_Abs; Jy++)
{
MF[Jx][Jy][Iz]+=(Val_Abs[Jx][Jy][Jz]*TmpWgt);
}
}
TmpWgt*=(TmpRat*(Degz-Iz)/(Iz+1));
}
}

MS=crt_ary3(Num_Abs+1, Degy+1, Degz+1);
for (int Iy = 0; Iy<= Degy; Iy++)
{
for (int Jx = 1; Jx<= Num_Abs; Jx++)
{
for (int Iz = 0; Iz<= Degz; Iz++)
{
MS[Jx][Iy][Iz]=0;
}
}
}

//Evaluate F^2 in (5.2.3)
for (int Jy = 1; Jy<= Num_Abs; Jy++)
```

```
{
TmpAbs=Abs[Jy];
TmpWgt=Wgt[Jy];
TmpWgt=TmpWgt*pow(1-TmpAbs,Degy);
TmpRat=TmpAbs/(1-TmpAbs);
for (int Iy = 0; Iy<= Degy; Iy++)
{
for (int Jx = 1; Jx<= Num_Abs; Jx++)
{
for (int Iz = 0; Iz<= Degz; Iz++)
{
MS[Jx][Iy][Iz]+=(MF[Jx][Jy][Iz]*TmpWgt);
}
}
TmpWgt*=(TmpRat*(Degy-Iy)/(Iy+1));
}
}
for (int Ix = 0; Ix<= Degx; Ix++)
{
for (int Iy = 0; Iy<= Degy; Iy++)
{
for (int Iz = 0; Iz<= Degz; Iz++)
{
MLoad[Ix][Iy][Iz]=0;
}
}
}

//Evaluate the moments, which are stored in the array MLoad.
```

```
for (int Jx = 1; Jx<= Num_Abs; Jx++)
{
TmpAbs=Abs[Jx];
TmpWgt=Wgt[Jx];
TmpWgt=TmpWgt*pow(1-TmpAbs,Degx);
TmpRat=TmpAbs/(1-TmpAbs);
for (int Ix = 0; Ix<= Degx; Ix++)
{
for (int Iy = 0; Iy<= Degy; Iy++)
{
for (int Iz = 0; Iz<= Degz; Iz++)
{
MLoad[Ix][Iy][Iz]+=(MS[Jx][Iy][Iz]*TmpWgt);
}
}
TmpWgt*=(TmpRat*(Degx-Ix)/(Ix+1));
}
}
}
```

A.2.2 Load Vectors

The following function returns the load vector, which is stored in `MLoad`. Vertices of the hexahedron are specified by `Vertex`. `Degx`, `Degy`, and `Degz` assign the degree of the shape function. `Num_Abs` gives the number of the quadrature nodes.

```
double
LodVecHex(double ****Vertex, int Degx, int Degy, int Degz,
double ***MLoad, int Num_Abs)
```

```
{
double ***Val_Abs, *Abs, *Wgt;
Abs=new double [Num_Abs+1];
Wgt=new double [Num_Abs+1];
// Create 3d array
Val_Abs=crt_ary3(Num_Abs+1, Num_Abs+1, Num_Abs+1);
// Calculate quadrature points
GauJac_0_1(Num_Abs,0,0,Abs,Wgt);
for (int i= 1; i <=Num_Abs; i++)
{
for (int j= 1; j <=Num_Abs; j++)
{
for (int k= 1; k <=Num_Abs; k++)
{
//Evaluate values at quadrature points of
//transformed function in (5.2.3)
Val_Abs[i][j][k]=FunLodTran(Vertex, Abs[i],Abs[j],Abs[k]);
}
}
}
//Calculate the load vector using (5.2.3)
Get_TP_moment(Degx, Degy, Degz, MLoad,
Num_Abs, Val_Abs, Abs, Wgt);
}
```

A.2.3 Mass Matrix

The following function returns the mass matrix stored in `Mat_mass`. The roles of the other parameters are the same as in the function for the load vector, LodVecHex.

```
void
MassMtrHex(double ****Vertex, int Degx, int Degy,
int Degz, double **Mat_mass, int Num_Abs)
{
double pdt, **binomialMatx, **binomialMaty,
**binomialMatz, ***MLoad, *Abs, *Wgt, ***Val_Abs;
Abs=new double [Num_Abs+1];
Wgt=new double [Num_Abs+1];
MLoad=crt_ary3(2*Degx+1, 2*Degy+1, 2*Degz+1);
Val_Abs=crt_ary3(Num_Abs+1, Num_Abs+1, Num_Abs+1);
//Create 2d arrays.
binomialMatx=create_BinomialMat(2*Degx+1);
binomialMaty=create_BinomialMat(2*Degy+1);
binomialMatz=create_BinomialMat(2*Degz+1);
//Evaluate binomial coefficients which are stored in the arrays
computeBinomials(binomialMatx, 2*Degx+1);
computeBinomials(binomialMaty, 2*Degy+1);
computeBinomials(binomialMatz, 2*Degz+1);
GauJac_0_1( Num_Abs,0,0,Abs,Wgt);
for (int i= 1; i <=Num_Abs; i++)
{
for (int j= 1; j <=Num_Abs; j++)
{
for (int k= 1; k <=Num_Abs; k++)
{
// Evaluate values at the quadrature points of the transformed
// function in (5.2.4)
Val_Abs[i][j][k]=FunMassTran(Vertex, Abs[i], Abs[j], Abs[k]);
```

```
}
}
}
//Calculate the moments of the transformed function
Get_TP_moment(2*Degx, 2*Degy, 2*Degz, MLoad,
Num_Abs, Val_Abs, Abs, Wgt);
//Evaluate the mass matrix using (5.2.4)
for (int I_1 = 0; I_1<= Degx; I_1++)
{
for (int J_1 = 0; J_1<= Degy; J_1++)
{
for (int K_1 = 0; K_1<= Degz; K_1++)
{
for (int I_2 = 0; I_2<= Degx; I_2++)
{
for (int J_2 = 0; J_2<= Degy; J_2++)
{
for (int K_2 = 0; K_2<= Degz; K_2++)
{
pdt=MLoad[I_1+I_2][J_1+J_2][K_1+K_2];
pdt*=binomialMatx[I_1][I_2];
pdt*=binomialMatx[Degx-I_2][Degx-I_1];
pdt/=binomialMatx[Degx][Degx];
pdt*=binomialMaty[J_1][J_2];
pdt*=binomialMaty[Degy-J_2][Degy-J_1];
pdt/=binomialMaty[Degy][Degy];
pdt*=binomialMatz[K_1][K_2];
pdt*=binomialMatz[Degz-K_2][Degz-K_1];
pdt/=binomialMatz[Degz][Degz];
Mat_mass[I_1*(Degy+1)*(Degz+1)+J_1*
(Degz+1)+K_1][I_2*(Degy+1)*(Degz+1)+J_2*(Degz+1)+K_2]=pdt;
}
```

A.2.4 Stiffness Matrix

The following function returns the mass matrix stored in `Mat_stiff`. The roles of the other parameters are the same as in the function for the load vector, LodVecHex.

```
void
StiffMtrHex(double ****Vertex,int Degx, int Degy, int Degz,
double **Mat_stiff, int Num_Abs)
{
for (int I_1 = 0; I_1<= Degx; I_1++)
{
for (int I_2 = 0; I_2<= Degy; I_2++)
{
for (int I_3 = 0; I_3<= Degz; I_3++)
{
for (int J_1 = 0; J_1<= Degx; J_1++)
{
for (int J_2 = 0; J_2<= Degy; J_2++)
{
for (int J_3 = 0; J_3<= Degz; J_3++)
{
Mat_stiff[I_1*(Degy+1)*(Degz+1)+I_2*
(Degz+1)+I_3][J_1*(Degy+1)*(Degz+1)+J_2*(Degz+1)+J_3]=0;
}
}
}
}
}
}
int d[2][3][3]={{{0,0,0},{0,0,0},{0,0,0}},{{1,0,0},
{0,1,0},{0,0,1}}}, c[3];
c[0]=Degx; c[1]=Degy; c[2]=Degz;
double **binomialMatx, **binomialMaty, **binomialMatz;
//Evaluate binomial coef.
binomialMatx=create_BinomialMat(2*Degx+3);
binomialMaty=create_BinomialMat(2*Degy+3);
binomialMatz=create_BinomialMat(2*Degz+3);
double ***Tem, ***Mom, *Abs, *Wgt,***Val_Abs,
****MStifAbs;
Tem=crt_ary3(2*Degx+5, 2*Degy+5, 2*Degz+5);
Mom=crt_ary3(2*Degx+1, 2*Degy+1, 2*Degz+1);
Val_Abs=crt_ary3(Num_Abs+1, Num_Abs+1, Num_Abs+1);
//4d array
MStifAbs=crt_ary4(9, Num_Abs+1, Num_Abs+1, Num_Abs+1);
Abs=new double [Num_Abs+1];
Wgt=new double [Num_Abs+1];
//Evaluate quadrature points
GauJac_0_1( Num_Abs,0,0,Abs,Wgt);
//Find values of the transformed functions
//in Section (5.2.4) at the quadrature points
FstifTran(Abs, MStifAbs, Vertex, Num_Abs);
int mt, nt, rt, mb, nb, rb, It_1, It_2, It_3,
Jt_1, Jt_2, Jt_3;
for(int i=0;i<3;i++)
```

```
{
mt=Degx-d[1][i][0];
nt=Degy-d[1][i][1];
rt=Degz-d[1][i][2];
for(int j=0;j<3;j++)
{
mb=Degx-d[1][j][0];
nb=Degy-d[1][j][1];
rb=Degz-d[1][j][2];
for (int v= 0; v <2*Degx+3; v++)
{
for (int w= 0; w <2*Degx+3; w++)
{
binomialMatx[v][w]=0;
}
}
for (int v= 0; v <2*Degy+3; v++)
{
for (int w= 0; w <2*Degy+3; w++)
{
binomialMaty[v][w]=0;
}
}
for (int v= 0; v <2*Degz+3; v++)
{
for (int w= 0; w <2*Degz+3; w++)
{
binomialMatz[v][w]=0;
}
```

```
}
computeBinomials(binomialMatx, mt+mb+1);
computeBinomials(binomialMaty, nt+nb+1);
computeBinomials(binomialMatz, rt+rb+1);
//Transform matrix so that it gives 0
//when entries with negative index called
Rearng_binomial(1, 1, binomialMatx, mt+mb+1,
mt+mb+1, 2*Degx+3, 2*Degx+3);
Rearng_binomial(1, 1, binomialMaty, nt+nb+1,
nt+nb+1, 2*Degy+3, 2*Degy+3);
Rearng_binomial(1, 1, binomialMatz, rt+rb+1,
rt+rb+1, 2*Degz+3, 2*Degz+3);
for (int u= 1; u <=Num_Abs; u++)
{
for (int v= 1; v <=Num_Abs; v++)
{
for (int w= 1; w <=Num_Abs; w++)
{
Val_Abs[u][v][w]=MStifAbs[3*i+j][u][v][w];
}
}
}
for (int u= 0; u <2*Degx+1; u++)
{
for (int v= 0; v <2*Degy+1; v++)
{
for (int w= 0; w <2*Degz+1; w++)
{
Mom[u][v][w]=0;
}
}
}
// Evaluate each moments needed in
// the formula in Section (5.2.5)
Get_TP_moment(mt+mb, nt+nb, rt+rb, Mom,
Num_Abs, Val_Abs, Abs, Wgt);
for(int u=0;u<2*Degx+5;u++)
{
for(int v=0;v<2*Degy+5;v++)
{
for(int w=0;w<2*Degz+5;w++)
{
Tem[u][v][w]=0;
}
}
}
for(int u=0;u<2*Degx+1;u++)
{
for(int v=0;v<2*Degy+1;v++)
{
for(int w=0;w<2*Degz+1;w++)
{
Tem[u+2][v+2][w+2]=Mom[u][v][w];
}
}
}
double sgn=1;
for(int k=0;k<2;k++)
```

```
{
for(int l=0;l<2;l++)
{
sgn=pow(-1,k+l);
for (int I_1 = 0; I_1<= Degx; I_1++)
{
It_1=I_1-d[k][i][0];
for (int I_2 = 0; I_2<= Degy; I_2++)
{
It_2=I_2-d[k][i][1];
for (int I_3 = 0; I_3<= Degz; I_3++)
{
It_3=I_3-d[k][i][2];
for (int J_1 = 0; J_1<= Degx; J_1++)
{
Jt_1=J_1-d[l][j][0];
for (int J_2 = 0; J_2<= Degy; J_2++)
{
Jt_2=J_2-d[l][j][1];
for (int J_3 = 0; J_3<= Degz; J_3++)
{
Jt_3=J_3-d[l][j][2];

//Add in each summand in (5.2.5)
double Bio=1;
Bio*=binomialMatx[1+mt-It_1][1+mb-Jt_1];
Bio*=binomialMatx[1+It_1][1+Jt_1];
Bio/=binomialMatx[1+mt][1+mb];
Bio*=binomialMaty[1+nt-It_2][1+nb-Jt_2];
Bio*=binomialMaty[1+It_2][1+Jt_2];
Bio/=binomialMaty[1+nt][1+nb];
Bio*=binomialMatz[1+rt-It_3][1+rb-Jt_3];
Bio*=binomialMatz[1+It_3][1+Jt_3];
Bio/=binomialMatz[1+rt][1+rb];
Mat_stiff[I_1*(Degy+1)*(Degz+1)+I_2*
(Degz+1)+I_3][J_1*(Degy+1)*(Degz+1)+J_2*
(Degz+1)+J_3]+=(sgn*c[i]*c[j]*
Tem[It_1+Jt_1+2][It_2+Jt_2+2][It_3+Jt_3+2]*Bio);
}
}
}
}
}
}
}
}
}
}
}
```

Bibliography

[1] M. Ainsworth, G. Andriamaro, and O. Davydov. BBFEM: Bernstein-Bézier Finite Elements, C++ Software Library. `http://www.bbfem.de/`.

[2] M. Ainsworth, G. Andriamaro, and O. Davydov. Bernstein-Bézier finite elements of Arbitrary Order and Optimal Assembly Procedures. *SIAM Journal on Scientific Computing*, 33(6):3087–3109, 2011.

[3] M. Ainsworth, O. Davydov, and L. Schumaker. Bernstein-Bézier Finite Elements in Tetrahedral-hexahedral-pyramidal Partitions. *Preprint available at https://www.staff.uni-giessen.de/odavydov/pyramids.html*, 2015.

[4] I. Akkerman, Y. Bazilevs, V. Calo, T. Hughes, and S. Hulshoff. The Role of Continuity in Residual-based Variational Multiscale Modeling of Turbulence. *Computational Mechanics*, 41(3):371–378, 2008.

[5] P. Antolin, A. Buffa, F. Calabro, 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*, 285:817–828, 2015.

[6] F. Auricchio, F. Calabro, T. Hughes, A. Reali, and G. Sangalli. A Simple Algorithm for Obtaining Nearly Optimal Quadrature Rules for Nurbs-based Isogeometric Analysis. *Computer Methods in Applied Mechanics and Engineering*, 249:15–27, 2012.

[7] I. Babuška and M. Suri. The p and h-p Versions of the Finite Element Method, Basic Principles and Properties. *SIAM Review*, 36(4):578–632, 1994.

[8] Y. Bazilevs, L. Beirao da Veiga, J. Cottrell, T. Hughes, and G. Sangalli. Isogeometric Analysis: Approximation, Stability and Error Estimates for h-refined Meshes. *Mathematical Models and Methods in Applied Sciences*, 16(07):1031–1090, 2006.

[9] Y. Bazilevs, V. Calo, J. Cottrell, T. Hughes, A. Reali, and G. Scovazzi. Variational Multiscale Residual-Based Turbulence Modeling for Large Eddy Simulation of Incompressible Flows. *Computer Methods in Applied Mechanics and Engineering*, 197(1):173–201, 2007.

[10] M. I. Bhatti and P. Bracken. Solutions of Differential Equations in a Bernstein Polynomial Basis. *Journal of Computational and Applied Mathematics*, 205(1):272–280, 2007.

[11] B. D. Bojanov, H. Hakopian, and B. Sahakian. *Spline Functions and Multivariate Interpolations*, volume 248. Springer, 2013.

[12] P. Bornemann and F. Cirak. A Subdivision-Based Implementation of the Hierarchical B-spline Finite Element Method. *Computer Methods in Applied Mechanics and Engineering*, 253:584–598, 2013.

[13] S. C. Brenner and R. Scott. *The Mathematical Theory of Finite Element Methods (3rd Edition)*. New York, NY : Springer, 2008.

[14] R. Brownlee, E. H. Georgoulis, and J. Levesley. Extending the Range of Error Estimates for Radial Approximation in Euclidean Space and on Spheres. *SIAM Journal on Mathematical Analysis*, 39(2):554–564, 2007.

[15] M. D. Buhmann. Radial Basis Functions: Theory and Implementations.

[16] C. K. Chui. *Multivariate Splines.* Society for Industrial & Applied Mathematics,U.S., 1987.

[17] J. Cottrell, T. Hughes, and A. Reali. Studies of Refinement And Continuity in Isogeometric Structural Analysis. *Computer Methods in Applied Mechanics and Engineering*, 196(41):4160–4183, 2007.

[18] J. Cottrell, A. Reali, Y. Bazilevs, and T. Hughes. Isogeometric Analysis of Structural Vibrations. *Computer Methods in Applied Mechanics and Engineering*, 195(41):5257–5296, 2006.

[19] J. A. Cottrell, T. J. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA.* John Wiley & Sons, 2009.

[20] J. A. Cottrell, T. J. Hughes, A. Reali, and G. Sangalli. Isogeometric Discretizations In Structural Dynamics and Wave Propagation. In *ECOMAS Thematic Conference on Computational Methods in Structural Dynamics and Earthquake Engineering, Crete, Greece*, pages 13–16, 2007.

[21] M. G. Cox. The Numerical Evaluation of B-splines. *IMA Journal of Applied Mathematics*, 10(2):134–149, 1972.

[22] L. B. da Veiga, A. Buffa, J. Rivas, and G. Sangalli. Some Estimates for h–p–k-Refinement in Isogeometric Analysis. *Numerische Mathematik*, 118(2):271–305, 2011.

[23] P. J. Davis. *Interpolation and Approximation.* Courier Corporation, 1975.

[24] C. de Boor. On Calculating with B-Splines. *Journal of Approximation Theory*, 6(1):50–62, 1972.

[25] C. De Boor. A Practical Guide to Splines. *Mathematics of Computation*, 1978.

[26] C. De Boor and R. E. Lynch. On Splines and Their Minimum Properties. *J. Math. Mech*, 15(6):953–969, 1966.

[27] R. A. DeVore and G. G. Lorentz. *Constructive Approximation*, volume 303. Springer, 1993.

[28] T. Eibner and J. M. Melenk. Fast Algorithms for Setting up the Stiffness Matrix in hp-fem: A Comparison. *Computer Mathematics and Its Applications: Advances and Developments*, pages 575–596, 2006.

[29] A. Ern and J.-L. Guermond. *Theory and Practice of Finite Elements*, volume 159. Springer, 2013.

[30] J. A. Evans, Y. Bazilevs, I. Babuška, and T. J. Hughes. N-widths, Sup-infs, and Optimality Ratios for the k-version of the Isogeometric Finite Element Method. *Computer Methods in Applied Mechanics and Engineering*, 198(21):1726–1741, 2009.

[31] L. C. Evans. *Partial Differential Equations*. American Mathematical Society, 1998.

[32] A. Frey, C. Hall, and T. Porsching. Some Results on The Global Inversion of Bilinear and Quadratic Isoparametric Finite Element Transformations. *Mathematics of Computation*, 32(143):725–749, 1978.

[33] H. Gómez, V. M. Calo, Y. Bazilevs, and T. J. Hughes. Isogeometric Analysis of the Cahn- Hilliard Phase-field Model. *Computer Methods in Applied Mechanics and Engineering*, 197(49):4333–4352, 2008.

[34] T. N. T. Goodman and S. L. Lee. A Remainder Formula and Limits of Cardinal Spline Interpolants. *Transactions of The American Mathematical Society*, 271, 1982.

[35] T. N. T. Goodman and S. L. Lee. The Budan-Fourier Theorem and Hermite-Birkhoff Spline Interpolation. *Trans. Amer. Math. Soc.*, 271(2):451–467, 1982.

[36] A. Graham. *Kronecker Products and Matrix Calculus : With Applications.* Chichester : Horwood ; New York : Halsted Press, 1981.

[37] R. M. Gray. *Toeplitz and Circulant Matrices: A Review.* Now Publishers Inc, 2006.

[38] D. F. Griffiths and D. J. Higham. *Numerical Methods for Ordinary Differential Equations: Initial Value Problems.* Springer, 2010.

[39] C. Grossmann, H.-G. Roos, and M. Stynes. *Numerical Treatment of Partial Differential Equations.* Teubner, 2005.

[40] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems.* Springer, 1993.

[41] M. Hazewinkel. *Encyclopaedia of Mathematics: Reaction-Diffusion Equation - Stirling Interpolation Formula.* Encyclopaedia of Mathematics: An Updated and Annotated Translation of the Soviet "Mathematical Encyclopaedia". Reidel.

[42] K. Höllig. *Finite Element Methods with B-Splines.* Society for Industrial Mathematics, 2003.

[43] K. Höllig, U. Reif, and J. Wipper. Weighted Extended B-spline Approximation of Dirichlet Problems. *SIAM Journal on Numerical Analysis*, 39(2):442–462, 2001.

[44] M. H. Holmes. *Introduction to Numerical Methods in Differential Equations*, volume 52. Springer, 2007.

[45] T. J. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry and Mesh Refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39):4135–4195, 2005.

[46] T. J. 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(49):4104–4124, 2008.

[47] T. J. Hughes, A. Reali, and G. Sangalli. Efficient Quadrature for NURBS-based Isogeometric Analysis. *Computer Methods in Applied Mechanics and Engineering*, 199(5):301–313, 2010.

[48] A. Jeffrey and D. Zwillinger. *Table of Integrals, Series, and Products*. Table of Integrals, Series, and Products Series. Elsevier Science, 2007.

[49] W. Jiang and J. E. Dolbow. Adaptive Refinement of Hierarchical B-spline Finite Elements With an Efficient Data Transfer Algorithm. *International Journal for Numerical Methods in Engineering*, 102(3-4):233–256, 2015.

[50] P. Kagan, A. Fischer, and P. Z. Bar-Yoseph. New B-spline Finite Element Approach for Geometrical Design and Mechanical Analysis. *International Journal f*, 41(3):435–458, 1998.

[51] G. Karniadakis and S. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford University Press, 2013.

[52] P. Knabner, S. Korotov, and G. Summ. Conditions for the Invertibility of the Isoparametric Mapping for Hexahedral Finite Elements. *Finite Elements In Analysis and Design*, 40(2):159–172, 2003.

[53] N. Korneichuk. *Exact Constants in Approximation Theory*. Cambridge University Press, 1991.

[54] A. R. Krommer. *Numerical Integration: On Advanced Computer Systems*, volume 848. Springer, 1994.

[55] M.-J. Lai and L. L. Schumaker. *Spline Functions on Triangulations*. Number 110. Cambridge University Press, 2007.

[56] S. Larsson and V. Thomée. *Partial Differential Equations with Numerical Methods*, volume 45. Springer, 2008.

[57] A. Leung and F. Au. Spline Finite Elements for Beam and Plate. *Computers and Structures*, 37(5):717–729, 1990.

[58] J. Levesley, W. Light, D. Ragozin, and X. Sun. A Simple Approach to the Variational Theory for Interpolation on Spheres. In *New Developments in Approximation Theory*, pages 117–143. Springer, 1999.

[59] J. Liu, L. Dedè, J. A. Evans, M. J. Borden, and T. J. Hughes. Isogeometric Analysis of The Advective Cahn–Hilliard Equation: Spinodal Decomposition under Shear Flow. *Journal of Computational Physics*, 242:321–350, 2013.

[60] C. F. V. Loan. *Computational Frameworks for the Fast Fourier Transform*. SIAM, Philadelphia, 1992.

[61] M. Luah and S. Fan. General Free Vibration Analysis of Shells of Revolution Using the Spline Finite Element Method. *Computers and Structures*, 33(5):1153–1162, 1989.

[62] J. M. Melenk, K. Gerdes, and C. Schwab. Fully Discrete hp-Finite Elements: Fast Quadrature. *Computer Methods in Applied Mechanics and Engineering*, 190(32):4339–4364, 2001.

[63] S. Moaveni. *Finite Element Analysis: Theory and Application with ANSYS*. Pearson Education India, 2003.

[64] J. T. Oden and J. N. Reddy. *An Introduction to the Mathematical Theory of Finite Elements*. Courier Corporation, 2012.

[65] S. A. Orszag. Spectral Methods for Problems In Complex Geometries. *Journal of Computational Physics*, 37(1):70–92, 1980.

[66] G. M. Phillips. *Interpolation and Approximation by Polynomials*, volume 14. Springer, 2003.

[67] L. Piegl and W. Tiller. *The NURBS Book*. Springer, 2012.

[68] M. J. D. Powell. *Approximation Theory and Methods*. Cambridge University Press, 1981.

[69] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Springer, 2002.

[70] K. R. Rao, D. N. Kim, and J. J. Hwang. *Fast Fourier Transform-Algorithms and Applications*. Springer, 2011.

[71] S. S. Rao. *The Finite Element Method in Engineering*. Butterworth-Heinemann, 2005.

[72] A. Reali. An Isogeometric Analysis Approach for The Study of Structural Vibrations. *Journal of Earthquake Engineering*, 10(spec01):1–30, 2006.

[73] A. Reali and T. J. Hughes. An Introduction to Isogeometric Collocation Methods. In *Isogeometric Methods for Numerical Simulation*, pages 173–204. Springer, 2015.

[74] M. Renardy and R. C. Rogers. *An Introduction to Partial Differential Equations*, volume 13. Springer, 2006.

[75] D. F. Rogers. *An Introduction to NURBS: with Historical Perspective*. Elsevier, 2000.

[76] D. Schillinger, L. Dede, M. A. Scott, J. A. Evans, M. J. Borden, E. Rank, and T. J. Hughes. An Isogeometric Design-through-analysis Methodology Based on Adaptive Hierarchical Refinement of NURBS, Immersed Boundary Methods, and T-spline CAD Surfaces. *Computer Methods in Applied Mechanics and Engineering*, 249:116–150, 2012.

[77] D. Schillinger, J. A. Evans, A. Reali, M. A. Scott, and T. J. Hughes. Isogeometric Collocation: Cost Comparison with Galerkin Methods and Extension to Adaptive Hierarchical NURBS Discretizations. *Computer Methods in Applied Mechanics and Engineering*, 267:170–232, 2013.

[78] D. Schillinger, S. J. Hossain, and T. J. Hughes. Reduced Bézier Element Quadrature Rules for Quadratic and Cubic Splines in Isogeometric Analysis. *Computer Methods in Applied Mechanics and Engineering*, 277:1–45, 2014.

[79] I. J. Schoenberg. *Cardinal Spline Interpolation*, volume 12. SIAM, 1973.

[80] I. J. Schoenberg. *On The Remainders and the Convergence of Cardinal Spline Interpolation for Almost Periodic Functions.* Springer, 1988.

[81] L. Schumaker. *Spline Functions: Basic Theory.* Cambridge University Press, 2007.

[82] C. Schwab. *p- and hp- Finite Element Methods Theory and Applications in Solid and Fluid Mechanics.* Clarendon Press, 1998.

[83] W. Steeb and Y. Hardy. *Matrix Calculus and Kronecker Product: A Practical Approach to Linear and Multilinear Algebra.* World Scientific, 2011.

[84] K. Subbaraj and M. Dokainish. A Survey of Direct Time-Integration Methods in Computational Structural Dynamics ii. Implicit Methods. *Computers & Structures*, 32(6):1387–1401, 1989.

[85] A. Tagliabue, L. Dedè, and A. Quarteroni. Isogeometric Analysis and Error Estimates for High Order Partial Differential Equations in Fluid Dynamics. *Computers and Fluids*, 102:277–303, 2014.

[86] S. Takacs and T. Takacs. Approximation Error Estimates and Inverse Inequalities for B-splines of Maximum Smoothness. *arXiv preprint arXiv:1502.03733*, 2015.

[87] V. Thomée. *Galerkin Finite Element Methods for Parabolic Problems.* Springer London, Limited, 2006.

[88] L. N. Trefethen. *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations.* Cornell University-Department of Computer Science and Center for Applied Mathematics, 1996.

[89] P. E. Vos, S. J. Sherwin, and R. M. Kirby. From h to p Efficiently: Implementing Finite and Spectral/hp Element Methods to Achieve Optimal Performance for Low-and High-order Discretisations. *Journal of Computational Physics*, 229(13):5161–5181, 2010.

[90] A.-V. Vuong. *Adaptive Hierarchical Isogeometric Finite Element Methods.* Springer, 2012.

[91] W. L. Wood. Practical Time-Stepping Schemes. *International Journal for Numerical Methods in Engineering*, 31(1), 1991.

[92] E. Zeidler. *Nonlinear Functional Analysis and Its Applications, II/A:Linear Monotone Operators, II/B: Nonlinear Monotone Operators.* Springer, New York, 1990.

[93] F. Zhang. *Matrix Theory: Basic Results and Techniques.* Springer, 2011.

[94] S. Zhang. Invertible Jacobian for Hexahedral Finite Elements. Part 1. Bijectivity. *Preprint available at http://www. math. udel. edu/˜ szhang/research/p/bijective1. ps*, 2005.

[95] J. Zhu, R. L. Taylor, and O. C. Zienkiewicz. *The Finite Element Method: Its Basis and Fundamentals*. Butterworth-Heinemann, 2005.

[96] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method*, volume 3. McGraw-Hill London, 1977.