Fast Tensor-Product Solvers for the Numerical Solution of Partial Differential Equations

Application to Deformed Geometries and to Space-Time Domains

Camilla Røvik

Master of Science in Physics and Mathematics
Submission date: June 2010
Supervisor: Einar Rønquist, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Problem Description

The objective with this study is to investigate how to solve partial differential equations accurately and efficiently. To obtain an accurate numerical solution, a spectral method based on high order polynomials will be used. First, the Poisson problem will be studied in non-rectangular geomteries. The potential of using fast tensor-product solvers to solve the resulting system of algebraic equations will be investigated. Next, the unsteady diffusion equation and the unsteady convection-diffusion equation will be considered. A spectral method will be used to approximate the solution in a combined space-time domain. Again, the possibility of using fast tensor-product solvers to solve the resulting system of algebraic equations will be investigated.

Assignment given: 22. January 2010
Supervisor: Einar Rønquist, MATH

Preface

This master thesis was written in the spring semester of 2010 at the Department of Mathematical Sciences at the Norwegian University of Science and Technology (NTNU).

I would like to take this opportunity to thank my fellow students for contributing their time and knowledge to the process of writing this paper.

A special thanks is directed to my advisor Professor Einar Rønquist. He has inspired me with his knowledge of this topic and has assisted me in solving problems throughout the project.

Camilla Røvik
June, 2010
NTNU Gløshaugen

Abstract

Spectral discretization in space and time of the weak formulation of a partial differential equations (PDE) is studied. The exact solution to the PDE, with either Dirichlet or Neumann boundary conditions imposed, is approximated using high order polynomials. This is known as a spectral Galerkin method.

The main focus of this work is the solution algorithm for the arising algebraic system of equations. A direct fast tensor-product solver is presented for the Poisson problem in a rectangular domain. We also explore the possibility of using a similar method in deformed domains, where the geometry of the domain is approximated using high order polynomials. Furthermore, time-dependent PDE's are studied. For the linear convection-diffusion equation in $\mathbb{R}$ we present a tensor-product solver allowing for parallel implementation, solving $\mathcal{O}(N)$ independent systems of equations. Lastly, an iterative tensor-product solver is considered for a nonlinear time-dependent PDE. For most algorithms implemented, the computational cost is $\mathcal{O}(N^{p+1})$ floating point operations and a memory required of $\mathcal{O}(N^p)$ floating point numbers for $\mathcal{O}(N^p)$ unknowns. In this work we only consider $p = 2$, but the theory is easily extended to apply in higher dimensions. Numerical results verify the expected convergence for both the iterative method and the spectral discretization. Exponential convergence is obtained when the solution and domain geometry are infinitely smooth.

Contents

Chapter 1

Introduction

Spectral methods are mainly used to discretize partial differential equations (PDE's) in space [1]. Spectral discretization in space was first introduced in 1944 by Blinova for the purpose of solving large scale computations in fluid dynamics. It was first implemented by Silbermann in 1954. The application was extended to a wider range of problems in the following decades and then thoroughly analyzed in the 1980's.

Spectral discretization in time was first introduced in the 1980's [2,3]. In the past couple of decades the interest in this topic has increased and resulted in further research, e.g. [4–6], but far from all aspects have been studied in detail.

In this master thesis we consider spectral discretization in both time and space based on the weak formulation of the problem; a spectral Galerkin method. These methods are closely related to finite element methods (FEM's). The main difference is that FEM's divide the domain into smaller sub-domains, or elements, and approximate the solution with piece-wise continuous functions, whereas the spectral methods approximate the solution in the entire domain with high order smooth functions. We will also discuss spectral element methods, which are even closer related to the FEM's. The domain is then divided into elements, though larger than those of a FEM, and the solution is approximated with high order polynomials.

The advantage of using a spectral discretization is that the error depends on the regularity of the exact solution and the given data. If the exact solution is infinitely smooth, we can get exponential convergence. On the contrary, the FEM's have a fixed convergence rate [6].

We only discuss spectral methods based on high order polynomials. The reason for this is that polynomials are applicable to a wider range of problems than other function spaces. Fourier methods, for example, are limited to simple geometries and periodic boundary conditions.

The motivation for using spectral methods is clear; the convergence rate is fast for problems with a high degree of regularity. Another important aspect to take into consideration is the computational cost of solving the derived algebraic system of equations. Depending on the solution algorithm, the computational cost varies greatly. Exploiting tensor-product properties and local data structure we find fast solvers: fast tensor-product solvers. The computational cost for these methods can be close to optimal [1]. What we mean by optimal is that the computational cost and storage space is proportional to the degrees of freedom.

Tensor product solvers were introduced in the 60's to solve certain partial differential equations in the simple two dimensional rectangular domain, e.g. the Poisson problem [7]. In this work we consider simple rectangular domains, but we also explore the possibility of finding tensor-product solvers in deformed domains and for time-dependent PDE's.

Chapter 2

Mathematical preliminaries

Before we discuss spectral methods on specific model problems and their solution algorithm, we will introduce relevant mathematical theory and notation that will be used throughout the paper.

2.1 Spaces and norms

The abbreviated notation for the partial derivative is defined as

$$u_x \equiv \frac{\partial u}{\partial x}, \quad \text{and} \quad u_{xx} \equiv \frac{\partial^2 u}{\partial x^2}.$$

The Lebesgue space $L^2(\Omega)$ is defined as

$$L^2(\Omega) = \left\{ v \,\middle|\, \int_\Omega v^2 \,\mathrm{d}x < \infty \right\},$$

with the associated inner-product and $L^2(\Omega)$ norm,

$$\begin{aligned} (u, v)_{L^2(\Omega)} &= \int_\Omega u\, v \,\mathrm{d}x \quad \forall\, u, v \in L^2(\Omega), \\ \|u\|^2_{L^2(\Omega)} &= \int_\Omega u^2 \,\mathrm{d}x \quad \forall\, u \in L^2(\Omega). \end{aligned}$$

The Sobolev space $H^m(\Omega)$ is defined as

$$H^m(\Omega) = \left\{ v \,\middle|\, \sum_{i=0}^{m} \int_\Omega \left(\frac{\mathrm{d}^m v}{\mathrm{d}x^m} \right)^2 \mathrm{d}x < \infty \right\},$$

with the associated inner-product and $H^m(\Omega)$ norm,

$$
\begin{aligned}
(u, v)_{H^m(\Omega)} &= \sum_{i=0}^{m} \int_\Omega \left(\frac{\mathrm{d}^m u}{\mathrm{d}x^m}\right)\left(\frac{\mathrm{d}^m v}{\mathrm{d}x^m}\right) \mathrm{d}x \quad \forall u, v \in H^m(\Omega), \\
\|u\|^2_{H^m(\Omega)} &= \sum_{i=0}^{m} \int_\Omega \left(\frac{\mathrm{d}^m u}{\mathrm{d}x^m}\right)^2 \mathrm{d}x \qquad \forall u, v \in H^m(\Omega).
\end{aligned}
$$

For simplicity the spaces and norms are here introduced in $\mathbb{R}$, but equivalent definitions exist in $\mathbb{R}^N$ [8].

2.2 Gauss-Labatto Legendre quadrature

Gauss Labatto Legendre (GLL) quadrature is a method of evaluating an integral numerically over the domain $\widehat{\Omega} = (-1, 1)$,

$$
\int_{-1}^{1} f(\xi)\,\mathrm{d}\xi \simeq \sum_{\alpha=0}^{N} \rho_\alpha f(\xi_\alpha) \tag{2.1}
$$

where

$\xi_\alpha \in [-1, 1]$ are the GLL quadrature points, where $\xi_0 = -1$ and $\xi_N = 1$,

and $\rho_\alpha \in [0, 1]$ are the GLL qradrature weights, such that $\sum_{\alpha=0}^{N} \rho_\alpha = 2$.

The integral is evaluated exactly if $f(\xi)$ is a polynomial of degree S, $f(\xi) \in \mathbb{P}_S(\Omega)$, and $S \leq 2N - 1$ [6]. The main difference between GLL quadrature and Gaussian quadrature is that GLL quadrature includes the endpoints -1 and 1. This can be beneficial when approximating an unknown function with known boundary conditions. The $L^2(\widehat{\Omega})$ inner product can then be approximated with the GLL quadrature and the discrete inner product is given by

$$
(f, v)_N = \sum_{\alpha=0}^{N} \rho_\alpha f(\xi_\alpha) v(\xi_\alpha). \tag{2.2}
$$

The subscript N indicates that the integral is evaluated with the GLL quadrature, and it is not exact unless $fv \in \mathbb{P}_{2N-1}$. Let $v \in \mathbb{P}_N(\widehat{\Omega})$ and $f \in H^\sigma(\widehat{\Omega})$; then, the quadrature error estimate of (2.2) is given by [1]

$$
|(f, v) - (f, v)_N| \leq C \|f\|_{H^\sigma(\widehat{\Omega})} \|v\|_{L^2(\widehat{\Omega})}.
$$

2.3 Polynomial interpolation

We now consider interpolation in $\mathbb{R}$. All concepts introduced here are easily extended to $\mathbb{R}^N$, as we will see in following chapters. Interpolation is a method of approximating a function u for which we know a discrete set of data points $\{x_i, u(x_i)\}_{i=0}^N$. The interpolated function of $u(x)$, $I_N u(x)$, is exact at the discrete set of points

$$I_N u(x_i) = u(x_i) \quad \text{for } i = 0, 1, ..., N. \tag{2.3}$$

There are different types of interpolation methods, but we will only consider polynomial interpolation, where the function $u(x)$ is approximated by a polynomial,

$$u(x) \approx I_N u(x) \in \mathbb{P}_N(\Omega).$$

In general, $I_N u(x)$ can be written as $I_N u(x) = \sum_{i=0}^N a_i x^i$, where a_i are the basis coefficients and x^i are the basis functions. However, the Lagrange polynomials provide a more powerful basis for constructing higher order polynomial. These polynomials possess the properties

$$\ell_j(x) \in \mathbb{P}_N(\Omega), \tag{2.4}$$
$$\ell_j(x_i) = \delta_{ij}. \tag{2.5}$$

One Lagrange function is plotted in section 3.4, see Figure 3.2. The interpolated function can then be written as

$$u_N(x) \equiv I_N u(x) = \sum_{i=0}^N u_i \ell_i(x), \tag{2.6}$$

where $u_i = u(x_i)$. The Kronecker delta property of $\ell_i(x)$ (2.5) makes $u_N(x)$ exact at all the interpolation points x_i, which is what (2.3) requires.

Consider the function $u(\xi) \in H^\sigma(\widehat{\Omega})$, where $\widehat{\Omega} = (-1, 1)$ and $\sigma \in \mathbb{N}$. The approximated function $u_N(\xi)$ can then be written as (2.6). When we choose the interpolation points to be the Gauss-Labatto Legrende points ξ_i, and if u is smooth enough (in $\mathbb{R}^d$, $\sigma > \frac{d+1}{2}$), then

$$\|u - u_N\|_{L^2(\widehat{\Omega})} \leq cN^{-\sigma}\|u\|_{H^\sigma(\widehat{\Omega})}.$$

It is important to notice that this error bound is with respect to the $L^2(\widehat{\Omega})$ norm, i.e. $u - u_N$ measured in the $H^1(\widehat{\Omega})$ norm satisfies

$$\|u - u_N\|_{H^1(\widehat{\Omega})} \leq cN^{1-\sigma}\|u\|_{H^\sigma(\widehat{\Omega})}.$$

It is of particular interest that we choose the GLL points as interpolation points, as we will see in the following chapters. The GLL interpolation points give a more stable approximation than what an equidistant set of points does. The GLL points are distributed with higher density near the edges of $\widehat{\Omega}$, this gives a more stable solution. This is illustrated in Figure 2.1, where the interpolated solution $I_N f(x)$ of $f(x) = \frac{1}{1+16x^2}$ is shown. $I_N f(x)$ is interpolated with equidistant interpolation points and GLL points for $N = 8$ and $N = 12$. The equidistant points yield an interpolated function with more oscillations near the edges of $\widehat{\Omega}$ as N increases.

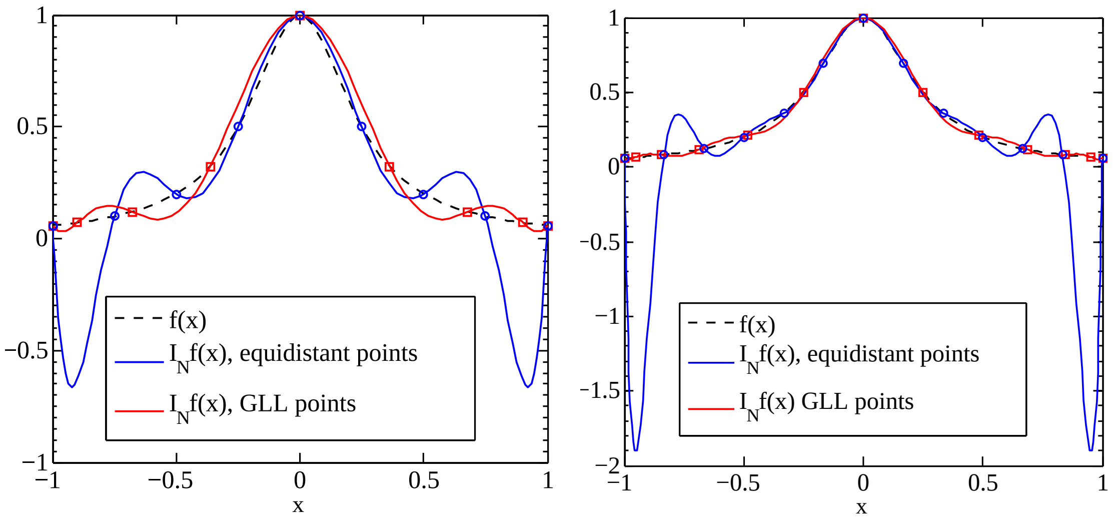


Figure 2.1: The interplant of $f(x)$, $I_N f$, is calculated with equidistant interpolation points, and GLL interpolation points for $N = 8$ (left) and $N = 12$ (right). The black dashed line is the analytical solution, $f(x) = \frac{1}{1+16x^2}$. The blue line is $I_N f$ with equidistant points marked with blue circles. The red line is $I_N f$ with GLL points marked with red circles.

2.4 Floating point operations

In the following chapter we put emphasis on the computational complexity of the algorithms presented. To evaluate the computational complexity we count the number of floating point operations, such as addition, subtraction, multiplication and division. Each such arithmetic operation takes a constant amount of time [9].

Chapter 3

Tensor product solvers in rectangular domains

In this chapter we will introduce tensor-products, and how their properties can be utilized to solve partial differential equations (PDE's) efficiently in rectangular domains.

3.1 Tensor products

First, we introduce some basic properties of tensor-products. A tensor-product is denoted by the symbol $\otimes$. Let $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2}$ and $\mathbf{B} \in \mathbb{R}^{n_3 \times n_4}$, then the tensor-product between the matrices $\mathbf{A}$ and $\mathbf{B}$ is defined as [7]

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{n_1 n_3 \times n_2 n_4},$$

where

$$\mathbf{C} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1n_2}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n_1 1}\mathbf{B} & \dots & \dots & a_{n_1 n_2}\mathbf{B} \end{bmatrix}.$$

The following properties apply for tensor-products [7, 10]:

1. $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$

2. $(\mathbf{A}+\mathbf{B})\otimes\mathbf{C}=\mathbf{A}\otimes\mathbf{C}+\mathbf{B}\otimes\mathbf{C}$

3. If $\mathbf{C}=\mathbf{A}\otimes\mathbf{B}$, then $\mathbf{C}^{-1}=\mathbf{A}^{-1}\otimes\mathbf{B}^{-1}$

4. If $\mathbf{C}=\mathbf{A}\otimes\mathbf{B}$, then $\mathbf{C}=(\mathbf{A}\otimes\mathbf{I})(\mathbf{I}\otimes\mathbf{B})$

5. If $\mathbf{A}$ and $\mathbf{B}$ are diagonal, then $\mathbf{C}=\mathbf{A}\otimes\mathbf{B}$ is diagonal.

For the first two properties we assume that the matrices have proper dimensions. Later we will see how these properties can be applied in clever ways to find fast solvers for the Poisson problem.

3.2 The Poisson problem

The Poisson problem is named after Siméon-Denis Poisson and has a wide range of applications in physics and mathematics. The Poisson problem in $\mathbb{R}^2$ is defined as

$$-\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=f(x,y)\quad\text{in a domain }\Omega,$$

with a suitable set of boundary conditions. The boundary of the domain Ω is denoted by $\partial\Omega$. Different types of boundary conditions where $\mathbf{n}$ is the normal vector are listed below [11].

Proper name	**Boundary condition**
Homogeneous Dirichlet	$u(x,y)\vert_{\partial\Omega}=0$
Nonhomogeneous Dirichlet	$u(x,y)\vert_{\partial\Omega}=f\neq 0$
Homogeneous Neumann	$\frac{\partial u}{\partial n}=\nabla u\cdot\mathbf{n}=0$
Nonhomogeneous Neumann	$\frac{\partial u}{\partial n}=f\neq 0$

Without loss of generality we will study the Poisson problem with homogeneous Dirichlet boundary conditions. The method can easily be extended to handle other boundary conditions.

3.3 The reference domain

In this chapter we will consider the Poisson problem in a simple rectangular domain. To solve this problem with any method numerically we find it useful to

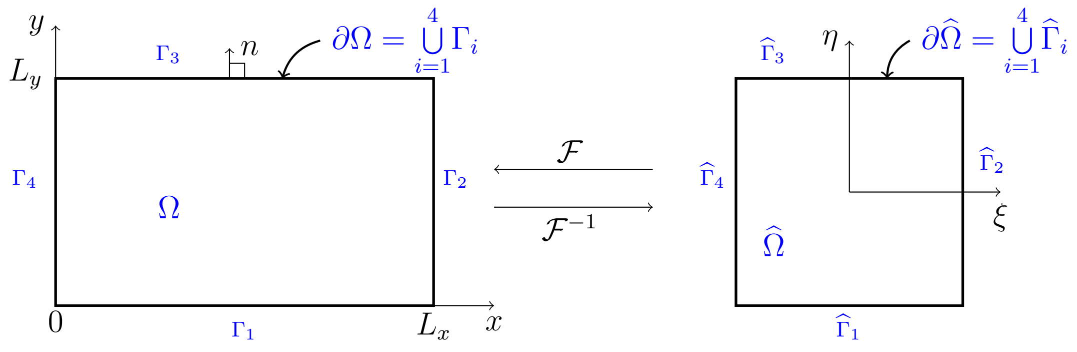


Figure 3.1: Illustration of the mapping between the reference domain $\widehat{\Omega} = (-1,1) \times (-1,1)$ and the rectangular physical domain $\Omega = (0, L_x) \times (0, L_y)$.

introduce a reference domain $\widehat{\Omega} = (-1,1) \times (-1,1)$. The variables in the reference domain are denoted with ξ and η. The physical domain $\Omega = (0, L_x) \times (0, L_y)$ can then be considered as an affine mapping $\mathcal{F}$ of the reference domain [10]

$$(x, y) = \mathcal{F}(\xi, \eta), \qquad \partial\Omega = \mathcal{F}(\partial\widehat{\Omega}).$$

This is illustrated in Figure 3.1. We call it an affine mapping because it is just a translation and stretching of the reference domain. The rectangular domain has the affine mapping

$$\begin{aligned} x = x(\xi) &= \frac{L_x}{2}(\xi + 1), & \frac{\partial x}{\partial \xi} &= \frac{\mathrm{d}x}{\mathrm{d}\xi} = \frac{L_x}{2} \\ y = y(\eta) &= \frac{L_y}{2}(\xi + 1), & \frac{\partial y}{\partial \eta} &= \frac{\mathrm{d}y}{\mathrm{d}\eta} = \frac{L_y}{2}. \end{aligned} \tag{3.1}$$

All coordinates (x, y) in the physical domain can thus be obtained uniquely from the corresponding coordinates (ξ, η) in the reference domain. This allows us to write a function $u(x, y)$ as

$$u(x, y) = u \circ \mathcal{F}(\xi, \eta)$$

or

$$u(x, y) = u(x(\xi), y(\eta)) = \hat{u}(\xi, \eta).$$

$\hat{u}$ indicates that u is a function of ξ and η. The partial derivatives of u can now be evaluated in terms of the reference variables

$$\begin{aligned} \frac{\partial u}{\partial x} &= \frac{\partial \hat{u}}{\partial \xi}\frac{\partial \xi}{\partial x} = \frac{\partial \hat{u}}{\partial \xi}\frac{2}{L_x}, \\ \frac{\partial u}{\partial y} &= \frac{\partial \hat{u}}{\partial \eta}\frac{\partial \eta}{\partial y} = \frac{\partial \hat{u}}{\partial \eta}\frac{2}{L_y}. \end{aligned} \tag{3.2}$$

3.4 The strong and weak formulation

The strong formulation of the Poisson problem in a two dimensional space with homogeneous Dirichlet boundary conditions is stated as: find u such that

$$-\nabla^2 u = f \quad \text{in } \Omega, \tag{3.3}$$

$$u = 0 \quad \text{on } \partial\Omega, \tag{3.4}$$

where ∇^2 is the Laplace operator, $\nabla^2 = \Delta = \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ and $f \in L^2(\Omega)$ is given. Define the function space

$$X = \left\{ v \in H^1(\Omega) \,\middle|\, v(x,y)|_{\partial\Omega} = 0 \right\}.$$

To obtain the weak formulation we multiply both sides of (3.3) by a test function $v \in X$ and integrate over the domain Ω. Green's identity [11] then gives the expression

$$\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x\,\mathrm{d}y - \int_{\partial\Omega} v(\nabla u \cdot \vec{n})\mathrm{d}S = \int_\Omega f\, v \,\mathrm{d}x\,\mathrm{d}y. \tag{3.5}$$

Notice that since $v \in X$, v is zero on the boundary $\partial\Omega$. Hence, the second term on the left hand side of (3.5) is zero and the weak formulation can be stated as: find $u \in X$ such that

$$a(u,v) = (f,v) \quad \forall\, v \in X, \tag{3.6}$$

where

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x\,\mathrm{d}y,$$

$$(f,v) = \int_\Omega f\, v \,\mathrm{d}x\,\mathrm{d}y.$$

Let us now consider these two expressions mapped to the reference domain $\widehat{\Omega}$. With (3.1) and (3.2), we obtain

$$a(u,v) = \int_{\widehat{\Omega}} \left(\frac{L_y}{L_x} \frac{\partial \hat{u}}{\partial \xi} \frac{\partial \hat{v}}{\partial \xi} + \frac{L_x}{L_y} \frac{\partial \hat{u}}{\partial \eta} \frac{\partial \hat{v}}{\partial \eta} \right) \mathrm{d}\xi\,\mathrm{d}\eta \tag{3.7}$$

and

$$(f,v) = \frac{L_x L_y}{4} \int_{\widehat{\Omega}} \hat{f}\, \hat{v} \,\mathrm{d}\xi\,\mathrm{d}\eta. \tag{3.8}$$

The next step is to find an approximate solution u_N of the problem. Let u_N be a polynomial of degree N in two dimensions; $u_N \in \mathbb{P}_N(\Omega)$. The polynomial space in the reference domain is defined as

$$\mathbb{P}_N(\widehat{\Omega}) = \left\{ v(\xi,\eta) |\, v(\xi,\eta^*) \in \mathbb{P}_N((-1,-1)), v(\xi^*,\eta) \in \mathbb{P}_N((-1,1)) \right\},$$

where the notation η^* and ξ^* indicate that these values are fixed. We get a polynomial of degree N in each spatial direction. There are many alternatives to seeking a polynomial solution, i.e. seeking a trigonometric approximation. However, such functions can only be applied to problems with periodic boundary conditions. The best approximation depends on the analytical solution and the method, but a polynomial approximation is well-fitted to solving general problems.

Define the discrete space

$$X_N = \{v(x,y) \in X \mid v \circ \mathcal{F}(\xi,\eta) \in \mathbb{P}_N(\widehat{\Omega})\}. \tag{3.9}$$

Notice that X_N and $\mathbb{P}_N(\Omega)$ have different dimensions. X_N looses two degrees of freedom to the Dirichlet boundary conditions, $\dim(X_N) = (N-1)^2$, while $\dim(\mathbb{P}_N) = (N+1)^2$. Since (3.6) holds for all $v \in X$ and $X_N \subset X$ the discrete problem can be stated as: find $u_N \in X_N$ such that

$$a(u_N, v) = (f, v) \quad \forall\, v \in X_N.$$

It is convenient to choose the bases for the polynomial space $\mathbb{P}_N(\widehat{\Omega})$ and the discrete space X_N to be nodal tensor-product bases of the Lagrange polynomials,

$$\mathbb{P}_N(\widehat{\Omega}) = \text{span}\{\, \ell_m(\xi)\, \ell_n(\eta)\}_{n,m=0}^{N}\,, \tag{3.10}$$

$$X_N(\widehat{\Omega}) = \text{span}\{\, \ell_m(\xi)\, \ell_n(\eta)\,\}_{n,m=1}^{N-1}\,. \tag{3.11}$$

Here $\ell_m(\xi)$ and $\ell_n(\eta)$ are the one-dimensional Lagrange polynomials through the Gauss Lobatto Legendre (GLL) points in each spatial direction. One of these functions is illustrated in Figure 3.2.

Recall that $u(x,y) = \hat{u}(\xi,\eta)$. The numerical solution can now be expressed as

$$\hat{u}_N(\xi,\eta) = \sum_{m=0}^{N}\sum_{n=0}^{N} u_{mn}\, \ell_m(\xi)\ell_n(\eta) = \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} u_{mn}\, \ell_m(\xi)\ell_n(\eta), \tag{3.12}$$

where $u_{mn} = u(\xi_m, \xi_n)$ are the nodal values. $u_{0j} = u_{Nj} = 0$ for $j = 0, ..., N$ and $u_{i0} = u_{iN} = 0$ are imposed by the boundary conditions. We call it a nodal tensor-product basis since the coefficients u_{mn} equal the exact solution at each node in the GLL grid. The GLL grid is illustrated in Figure 3.3.

We now return to our discrete problem $a(u_N, v) = (f, v)$, which holds for all $v \in X_N$. With the given basis functions we can choose $\hat{v} = \ell_i(\xi)\ell_j(\eta)$ for $i = 1, .., N-1$ and $j = 1, .., N-1$. First we consider the bilinear form $a(u_N, v)$, substitute $\hat{v}$ and $\hat{u}_N$ from (3.12) into (3.7), and for simplicity we evaluate the first

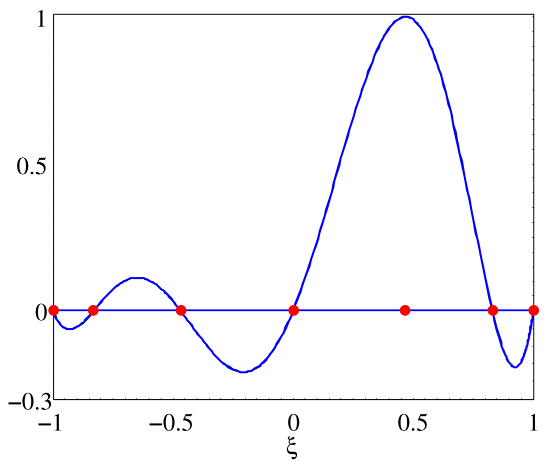


Figure 3.2: The one-dimensional Lagrange polynomial $\ell_4(\xi) \in \mathbb{P}_6((-1,1))$ through the $N+1=7$ GLL points, which are marked with red dots. $\ell_4(\xi)$ is zero at all the GLL points except at ξ_4, where it equals one.

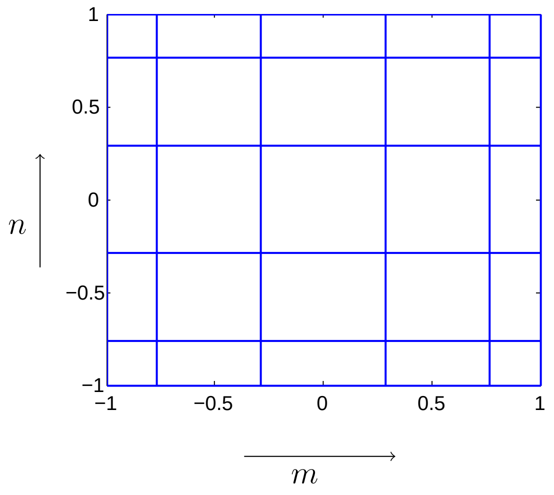


Figure 3.3: The GLL grid on the reference domain $\widehat{\Omega}$. The GLL points ξ_i, $i = 0, .., 5$ are distributed along each spatial directions.

term

$$\begin{aligned}\int_{\widehat{\Omega}} \frac{L_y}{L_x}\frac{\partial \hat{u}_N}{\partial \xi}\frac{\partial \hat{v}}{\partial \xi}\,\mathrm{d}\xi\,\mathrm{d}\eta &= \int_{-1}^{1}\int_{-1}^{1}\frac{L_y}{L_x}\left(\sum_{m=1}^{N-1}\sum_{n=1}^{N-1} u_{mn}\ell_m'(\xi)\ell_n(\eta)\right)\ell_i'(\xi)\ell_j(\eta)\,\mathrm{d}\xi\,\mathrm{d}\eta \\ &= \frac{L_y}{L_x}\sum_{m=1}^{N-1}\sum_{n=1}^{N-1}\underbrace{\int_{-1}^{1}\ell_i'(\xi)\ell_m'(\xi)\mathrm{d}\xi}_{(\ell_i'(\xi),\ell_m'(\xi))^1}\underbrace{\int_{-1}^{1}\ell_j(\eta)\ell_n(\eta)\mathrm{d}\eta\, u_{mn}}_{(\ell_j(\eta),\ell_n(\eta))^1} \\ &= \frac{L_y}{L_x}\sum_{m=1}^{N-1}\sum_{n=1}^{N-1}(\ell_i'(\xi),\ell_m'(\xi))^1(\ell_j(\eta),\ell_n(\eta))^1 u_{mn}\end{aligned}$$

Note that we get two separated one-dimensional integrals. The superscript 1 indicates that the integrals are one-dimensional. The first integral is the matrix elements of the stiffness matrix $\widehat{\mathbf{A}}$ in the one-dimensional reference domain, and the second is the matrix elements of the mass matrix $\widehat{\mathbf{B}}$. Let us now evaluate the integrals numerically using GLL quadrature. Define

$$\widehat{A}_{ij} \equiv (\ell_i'(\xi),\ell_j'(\xi))_N^1 = \sum_{\alpha=0}^{N}\rho_\alpha \ell_i'(\xi_\alpha)\ell_j'(\xi_\alpha), \tag{3.13}$$

$$\widehat{B}_{ij} \equiv (\ell_i(\xi),\ell_j(\xi))_N^1 = \sum_{\alpha=0}^{N}\rho_\alpha \ell_i(\xi_\alpha)\ell_j(\xi_\alpha) = \rho_i\delta_{ij}. \tag{3.14}$$

The subscript N indicates that the integrals are evaluated with GLL quadrature (2.1). If the integrand is a polynomial of degree K, GLL quadrature evaluates it exactly if $K \leq 2N-1$. Hence, $\widehat{A}_{ij}$ is evaluated exactly. We then get

$$\int_{\widehat{\Omega}} \frac{L_y}{L_x}\frac{\partial \hat{u}_N}{\partial \xi}\frac{\partial \hat{v}}{\partial \xi}\,\mathrm{d}\xi\,\mathrm{d}\eta \approx \frac{L_y}{L_x}\sum_{m=1}^{N-1}\sum_{n=1}^{N-1}\widehat{A}_{im}\widehat{B}_{jn}u_{mn}.$$

With a similar evaluation of the second term we get the following expression

$$a(u_N,v)_N = \sum_{m=1}^{N-1}\sum_{n=1}^{N-1}\left(\frac{L_y}{L_x}\widehat{A}_{im}\widehat{B}_{jn} + \frac{L_x}{L_y}\widehat{B}_{im}\widehat{A}_{jn}\right)u_{mn}. \tag{3.15}$$

The linear term can be evaluated in the same way,

$$(f,v)_N = \sum_{m=1}^{N-1}\sum_{n=1}^{N-1}\frac{L_yL_y}{4}\widehat{B}_{im}\widehat{B}_{jn}f_{mn}, \tag{3.16}$$

for $i = 1,...,N-1$ and $j = 1,...,N-1$. Combining (3.15) and (3.16) we finally get

$$\sum_{m=1}^{N-1}\sum_{n=1}^{N-1}\left(\frac{L_y}{L_x}\widehat{A}_{im}\widehat{B}_{jn} + \frac{L_x}{L_y}\widehat{B}_{im}\widehat{A}_{jn}\right)u_{mn} = \sum_{m=1}^{N-1}\sum_{n=1}^{N-1}\frac{L_yL_y}{4}\widehat{B}_{im}\widehat{B}_{jn}f_{mn} \tag{3.17}$$

for $i = 1, ..., N-1$ and $j = 1, ..., N-1$. When N increases the discrete error $\|u - u_N\|$ tends to zero according to the regularity of u. For analytical solutions we expect exponential convergence [1]

$$\|u - u_N\|_{H^1(\Omega)} \propto e^{-\mu N},$$

where μ is a constant depending on the analytical solution. Later we will discuss different methods for solving the algebraic system of equations in (3.17).

3.5 Local and global data representation

The derived algebraic system of equations for the Poisson problem can be solved in several ways. There is a significant difference in the number of operations and storage space required for the different methods. In this section we introduce local and global data representation. The representation is essential for deriving fast solvers.

First, consider

$$w_{ij} = \sum_{m=1}^{N-1} \sum_{n=1}^{N-1} \widehat{A}_{im} \widehat{B}_{jn} u_{mn}$$

or

$$w_{ij} = \sum_{m=1}^{N-1} \sum_{n=1}^{N-1} \widehat{A}_{im} u_{mn} \widehat{B}^T_{nj} \tag{3.18}$$

for $i = 1, .., N-1$ and $j = 1, .., N-1$. The representation and evaluation of this expression can be done in different ways. We can for instance represent u_{mn}, for $m, n = 1, ..., N-1$ in one long vector $\mathbf{u}^x$,

$$\mathbf{u}^x = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{n_y} \end{bmatrix} \in \mathbb{R}^{n_x n_y}, \qquad \text{where} \quad \mathbf{u}_j = \begin{bmatrix} u_{1j} \\ u_{2j} \\ \vdots \\ u_{n_x j} \end{bmatrix} \in \mathbb{R}^{n_x} \quad \text{and} \quad n_x = n_y = N-1.$$

The superscript x indicate that we stack the values of u_{ij} systematically by going through the x direction first. The storage space required is $\mathcal{O}(N^2)$. Making use of this representation, we can apply the tensor-product to evaluate (3.18) [12],

$$\mathbf{w}^x = \left(\widehat{\mathbf{B}} \otimes \widehat{\mathbf{A}}\right)\mathbf{u}^x. \tag{3.19}$$

If we explicitly express $\widehat{\mathbf{B}} \otimes \widehat{\mathbf{A}} \in \mathbb{R}^{(N-1)^2 \times (N-1)^2}$, the storage space required will be $\mathcal{O}(N^4)$ and the evaluation of $\mathbf{w}^x$ requires $\mathcal{O}(N^4)$ floating point operations.

This representation of the data is called *global* data structure. Another way of representing u_{mn} is in a matrix $\mathbf{U}$,

$$\mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n_y} \\ u_{21} & u_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ u_{n_x1} & \dots & \dots & u_{n_xn_y} \end{bmatrix} \in \mathbb{R}^{n_x \times n_y}, \qquad n_x = n_y = N - 1.$$

The storage space required for this *local* data structure is $\mathcal{O}(N^2)$, which is the same as for the global data structure, $\mathbf{u}^x$. When making use of local data structure, (3.18) can be evaluated with two matrix-matrix products

$$\mathbf{W} = \widehat{\mathbf{A}}\mathbf{U}\widehat{\mathbf{B}}^T. \tag{3.20}$$

The operational cost to evaluate $\mathbf{W}$ is $\mathcal{O}(N^3)$ and the storage space is $\mathcal{O}(N^2)$. Hence, the evaluation of (3.18) is much more efficient with local data structure.

3.6 Computational approach

In this section we will derive fast tensor-product solvers for the Poisson problem in the rectangular domain. The algebraic system of equations we want to solve is (3.17). With tensor-product notation we can write the system as

$$\underbrace{\left(\frac{L_y}{L_x}\widehat{\mathbf{B}} \otimes \widehat{\mathbf{A}} + \frac{L_x}{L_y}\widehat{\mathbf{B}} \otimes \widehat{\mathbf{A}}\right)}_{\mathbf{A}^{2D} \in \mathbb{R}^{(N-1)^2 \times (N-1)^2}} \mathbf{u}^x = \underbrace{\frac{L_y L_y}{4}\left(\widehat{\mathbf{B}} \otimes \widehat{\mathbf{B}}\right)\mathbf{f}^x}_{\mathbf{f}^{2D} \in \mathbb{R}^{(N-1)^2}}. \tag{3.21}$$

First, consider the generalized eigenvalue problem to the one dimensional operators [10]

$$\widehat{\mathbf{A}}\mathbf{q}_i = \lambda_i \widehat{\mathbf{B}}\mathbf{q}_i.$$

The two matrices $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{B}}$ are symmetric positive definite (SPD) and we therefore expect the eigenvalues to be real and positive; $\lambda_i \in \mathbb{R}$ and $\lambda_i > 0$. Define the matrices

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & \lambda_{N-1} \end{bmatrix}, \qquad \mathbf{Q} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{q}_1, & \mathbf{q}_1, & \dots, & \mathbf{q}_{N-1} \\ \vdots & \vdots & & \vdots \end{bmatrix}. \tag{3.22}$$

The generalized eigenvalue problem can be written in matrix form as

$$\widehat{\mathbf{A}}\mathbf{Q} = \widehat{\mathbf{B}}\mathbf{Q}\mathbf{\Lambda}.$$

Further, we get

$$\begin{aligned}\mathbf{Q}^T\widehat{\mathbf{A}}\mathbf{Q} &= \underbrace{\mathbf{Q}^T\widehat{\mathbf{B}}\mathbf{Q}}_{c\mathbf{I}}\mathbf{\Lambda}\\ &= c\mathbf{\Lambda}.\end{aligned}$$

Let us assume that the eigenvectors are scaled such that $c = 1$. Then the following expressions are obtained

$$\widehat{\mathbf{A}} = \mathbf{Q}^{-T}\mathbf{\Lambda}\mathbf{Q}^{-1}, \tag{3.23}$$

$$\widehat{\mathbf{B}} = \mathbf{Q}^{-T}\mathbf{Q}^{-1}. \tag{3.24}$$

Consider the expression of $\mathbf{A}^{2D}$ from (3.21). When we replace $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{B}}$ with (3.23) and (3.24), we get

$$\begin{aligned}\mathbf{A}^{2D} &= \left(\frac{L_y}{L_x}\mathbf{Q}^{-T}\mathbf{Q}^{-1}\otimes\mathbf{Q}^{-T}\mathbf{\Lambda}\mathbf{Q}^{-1} + \frac{L_x}{L_y}\mathbf{Q}^{-T}\mathbf{\Lambda}\mathbf{Q}^{-1}\otimes\mathbf{Q}^{-T}\mathbf{Q}^{-1}\right)\\ &= \left(\mathbf{Q}^{-T}\otimes\mathbf{Q}^{-T}\right)\frac{L_y}{L_x}\left(\mathbf{I}\otimes\mathbf{\Lambda}\right)\left(\mathbf{Q}^{-1}\otimes\mathbf{Q}^{-1}\right) + \left(\mathbf{Q}^{-T}\otimes\mathbf{Q}^{-T}\right)\frac{L_x}{L_y}\left(\mathbf{\Lambda}\otimes\mathbf{I}\right)\left(\mathbf{Q}^{-1}\otimes\mathbf{Q}^{-1}\right)\\ &= \left(\mathbf{Q}^{-T}\otimes\mathbf{Q}^{-T}\right)\left(\frac{L_y}{L_x}\mathbf{I}\otimes\mathbf{\Lambda} + \frac{L_x}{L_y}\mathbf{\Lambda}\otimes\mathbf{I}\right)\left(\mathbf{Q}^{-1}\otimes\mathbf{Q}^{-1}\right).\end{aligned}$$

The next step is to construct the inverse of $\mathbf{A}^{2D}$. We know that the inverse of $\mathbf{A} = \mathbf{B}\otimes\mathbf{C}$ is $\mathbf{A}^{-1} = \mathbf{B}^{-1}\otimes\mathbf{C}^{-1}$, and we get

$$\mathbf{A}^{-2D} = \left(\mathbf{Q}\otimes\mathbf{Q}\right)\left(\frac{L_y}{L_x}\mathbf{I}\otimes\mathbf{\Lambda} + \frac{L_x}{L_y}\mathbf{\Lambda}\otimes\mathbf{I}\right)^{-1}\left(\mathbf{Q}^T\otimes\mathbf{Q}^T\right).$$

We have now directly constructed the inverse of $\mathbf{A}^{2D}$ and the solution vector u^x can be expressed as

$$\mathbf{u}^x = \left(\mathbf{Q}\otimes\mathbf{Q}\right)\left(\frac{L_y}{L_x}\mathbf{I}\otimes\mathbf{\Lambda} + \frac{L_x}{L_y}\mathbf{\Lambda}\otimes\mathbf{I}\right)^{-1}\left(\mathbf{Q}^T\otimes\mathbf{Q}^T\right)\underbrace{\left(\frac{L_yL_y}{4}\mathbf{B}\otimes\mathbf{B}\right)\mathbf{f}^x}_{\mathbf{f}^{2D}}. \tag{3.25}$$

The operational cost of evaluating $\mathbf{Q}$, $\mathbf{Q}^T$ and $\mathbf{\Lambda}$ is $\mathcal{O}(N^3)$. After this evaluation we have a direct operator to compute the solution, $\mathbf{u}^x = \mathbf{A}^{-2D}\mathbf{f}^{2D}$. Both the storage space required and the operational cost for the method will be $\mathcal{O}(N^4)$. This is better than using Gaussian elimination on $\mathbf{A}^{2D}\mathbf{u}^x = \mathbf{f}^{2D}$, where the storage space is the same, but the operational cost is $\mathcal{O}(N^6)$.

To reduce the operational cost further, we can convert (3.25) back to local data structure using the relation between (3.19) and (3.20). Let $\mathbf{F}$ be the local representation of $\mathbf{f}^{2D}$, the final fast solution algorithm for the Poisson problem is then stated in **Algorithm 1**.

Algorithm 1 Fast Poisson solver: spectral method with Dirichlet boundary conditions

```
1. $\mathbf{V} = \mathbf{Q}^T\mathbf{F}\mathbf{Q}$
2. $w_{ij} = v_{ij} \,/\, \left(\frac{L_y}{L_x}\lambda_i + \frac{L_x}{L_y}\lambda_j\right)$ for $i, j = 1, .., N-1$
3. $\mathbf{U} = \mathbf{Q}\mathbf{W}\mathbf{Q}^T$
```

The most expensive operation for this algorithm with $\mathcal{O}(N^2)$ unknowns is the matrix-matrix products. Hardware related, the matrix-matrix products are among the most efficient operations and can be evaluated in $\mathcal{O}(N^3)$ operations. The storage space required is $\mathcal{O}(N^2)$. Notice that we have only used the one dimensional operators $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{B}}$ to construct $\mathbf{Q}$ and $\mathbf{\Lambda}$. The cost of evaluating $\mathbf{\Lambda}$ and $\mathbf{Q}$ are $\mathcal{O}(N^3)$.

Algorithm 1 is constructed for the model problem with homogeneous Dirichlet boundary conditions. Problems with mixed boundary conditions, where the dimension of the algebraic system of equations is $n_x \times n_y$ and $n_x \neq n_y$, require that we solve two generalized eigenvalue problems. With the same procedure as above the algorithm for the fast solver for such a problem is stated in **Algorithm 2**, where the subscript 1 and 2, indicate the dimension in the x and y direction, i.e $\mathbf{Q}_1 \in \mathbb{R}^{n_x \times n_x}$ and $\mathbf{Q}_2 \in \mathbb{R}^{n_y \times n_y}$.

Algorithm 2 Fast Poisson solver: spectral method with mixed boundary conditions

```
1. $\mathbf{V} = \mathbf{Q}_1^T\mathbf{F}\mathbf{Q}_2$
2. $w_{ij} = v_{ij} / \left(\frac{L_y}{L_x}\lambda_i + \frac{L_x}{L_y}\lambda_j\right)$    for $i, j = 1, .., N-1$
3. $\mathbf{U} = \mathbf{Q}_1\mathbf{W}\mathbf{Q}_2^T$
```

Now consider the Poisson problem with homogeneous Neumann boundary conditions. This problem does not have a unique solution. If $u^*(x, y)$ solves the Poisson problem

$$\begin{aligned} \nabla^2 u &= f \quad \text{in } \Omega, \\ \frac{\partial u}{\partial n} &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

then so does $u(x, y) = u^*(x, y) + C$, where C is a constant. Solving the generalized eigenvalue problem will give one $\lambda_i = 0$. We can not divide by zero. Therefore the second step of the algorithm must be modified, while steps one and three remain the same, see **Algorithm 3**.

Algorithm 3 Fast Poisson solver: spectral method with Neumann boundary conditions

```
1. $\mathbf{V} = \mathbf{Q}^T F \mathbf{Q}$
2. $w_{ij} = \begin{cases} v_{ij} / \left(\frac{L_y}{L_x}\lambda_i + \frac{L_x}{L_y}\lambda_j\right) & \text{if } \lambda_i \neq 0 \text{ or } \lambda_j \neq 0 \\ c_1 & \text{if } \lambda_i = \lambda_j = 0 \end{cases} \qquad \forall i, j$
3. $\mathbf{U} = \mathbf{Q}\mathbf{W}\mathbf{Q}^T$
```

Let us now explore what c_1 in **Algorithm 3** contributes to the solution. If we first organize $\mathbf{Q}$ and $\mathbf{\Lambda}$ such that $\lambda_0 = 0$, the corresponding eigenvector is a constant vector; $\mathbf{q}_0 = [a, a, .., a]^T$. Recall that we have scaled the eigenvectors such that $\mathbf{Q}^T \widehat{\mathbf{B}} \mathbf{Q} = \mathbf{I}$, which means that $\mathbf{q}_0^T \widehat{\mathbf{B}} \mathbf{q}_0 = 1$. We get

$$\begin{bmatrix} a & \cdots & a \end{bmatrix} \widehat{\mathbf{B}} \begin{bmatrix} a \\ \vdots \\ a \end{bmatrix} = a^2 \underbrace{\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \widehat{\mathbf{B}} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}}_{2} = a^2 2 = 1 \quad \Rightarrow \quad a = \frac{1}{\sqrt{2}}.$$

The second argument comes from

$$\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T \widehat{\mathbf{B}} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \sum_{i=0}^{N} \sum_{j=0}^{N} \rho_i \delta_{ij} = \sum_{i}^{N} \rho_i = \int_{-1}^{1} 1 \mathrm{d}\xi = 2.$$

After applying the second step of **Algorithm 3** we have $w_{00} = c_1$, the last step becomes

$$\begin{bmatrix} a & | & & | \\ \vdots & \mathbf{q_1} & \cdots & \mathbf{q_N} \\ a & | & & | \end{bmatrix} \begin{bmatrix} c_1 & & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} a & \cdots & a \\ \text{---} & \mathbf{q_1} & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{q_N} & \text{---} \end{bmatrix}.$$

This means that the contribution of c_1 is an outer product

$$\mathbf{U} = \mathbf{U}^* + c_1 \mathbf{q_0}\mathbf{q_0}^T = \mathbf{U}^* + c_1 \begin{bmatrix} a^2 & \dots & a^2 \\ \vdots & & \vdots \\ a^2 & \dots & a^2 \end{bmatrix} = \mathbf{U}^* + c_1 a^2 \mathbf{I}.$$

And since $a = \frac{1}{\sqrt{2}}$, we finally get

$$\mathbf{U} = \mathbf{U}^* + \frac{c_1}{2}\mathbf{I}.$$

This means that we raise our solution by the constant $C = \frac{c_1}{2}$. The discrete solution is now given by

$$u_N = u_N^* + \frac{c_1}{2}.$$

3.7 Finite differences

So far we have discussed tensor-product solvers for problems approximated with spectral discretization. Fast tensor-product solvers are not constrained to spectral methods; their applications can be applied on many numerical methods. The more specific and structured the problem is, the easier it is to make fast and accurate solvers. In this section we again solve the Poisson problem with homogeneous Dirichlet boundary conditions

$$\begin{aligned} -\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) &= f(x,y) \quad \text{in} \quad \Omega, \\ u(x,y)|_{\partial\Omega} &= 0, \end{aligned}$$

but now we use finite differences to solve the problem. Consider the central differences

$$\begin{aligned} \frac{\partial^2 u(x_m,y_n)}{\partial x^2} &= \frac{1}{h_x^2}\Big(u(x_m+h_x,y_n) - 2u(x_m,y_n) + u(x_m-h_x,y_n)\Big) + \mathcal{O}(h_x^2), \\ \frac{\partial^2 u(x_m,y_n)}{\partial y^2} &= \frac{1}{h_y^2}\Big(u(x_m,y_n+h_y) - 2u(x_m,y_n) + u(x_m,y_n-h_y)\Big) + \mathcal{O}(h_y^2), \end{aligned}$$

for $m,n = 1,..,N-1$, where $x_0 = y_0 = 0$, $x_N = L_x$, $y_N = L_y$, and the step size in the x and y direction is $h_x = L_x/N$ and $h_y = L_y/N$. Notice that we now use a uniform grid, see Figure 3.4. With central differences we can approximate the

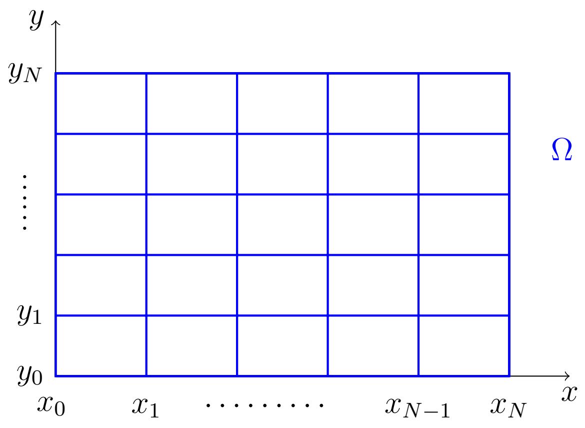


Figure 3.4: Uniform mesh of the domain $\Omega = (0, L_x) \times (0, L_y)$.

Poisson problem by the following system of equations

$$\begin{aligned} &\frac{1}{h_x^2}\Big(-u(x_m + h_x, y_n) + 2u(x_m, y_n) - u(x_m - h_x, y_n)\Big) \\ +&\frac{1}{h_y^2}\Big(-u(x_m, y_n + h_y) + 2u(x_m, y_n) - u(x_m, y_n - h_y)\Big) = f(x_m, y_n), \end{aligned}$$

where $m, n = 1, .., N-1$. In global form we write

$$\left(c_x\widehat{\mathbf{D}} \otimes \mathbf{I} + c_y\mathbf{I} \otimes \widehat{\mathbf{D}}\right)\mathbf{u}^x = \mathbf{f}^x, \tag{3.26}$$

where $c_x = \frac{1}{h_x^2}$, $c_y = \frac{1}{h_y^2}$, $f_{mn} = f(x_m, y_n)$, $u_{mn} = u(x_m, y_n)$, and

$$\widehat{\mathbf{D}} = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & -1 \\ 0 & \dots & 0 & -1 & 2 \end{bmatrix}.$$

To obtain a fast solver we solve the eigenvalue problem

$$\widehat{\mathbf{D}}\mathbf{R} = \mathbf{R}\boldsymbol{\Lambda},$$

where $\mathbf{R}$ is the eigenvector matrix and $\boldsymbol{\Lambda}$ is the eigenvalue matrix. We then obtain

$$\widehat{\mathbf{D}} = \mathbf{R}\boldsymbol{\Lambda}\mathbf{R}^{-1},$$

which we substitute into (3.26), and with the general tensor-product properties the following expressions are obtained

$$\left(\mathbf{R\Lambda R}^{-1} \otimes c_x \mathbf{RR}^{-1} + c_y \mathbf{RR}^{-1} \otimes \mathbf{R\Lambda R}^{-1}\right) \mathbf{u}^x = \mathbf{f}^x$$
$$\underbrace{(\mathbf{R} \otimes \mathbf{R})\,(c_x \mathbf{\Lambda} \otimes \mathbf{I} + \mathbf{I} \otimes c_y \mathbf{\Lambda})\left(\mathbf{R}^{-1} \otimes \mathbf{R}^{-1}\right)}_{\mathbf{A}^{2D}} \mathbf{u}^x = \mathbf{f}^x.$$

Again, with the general tensor-product properties we can find the inverse of $\mathbf{A}^{2D}$

$$\left(\mathbf{A}^{2D}\right)^{-1} = (\mathbf{R} \otimes \mathbf{R})\,(c_x \mathbf{\Lambda} \otimes \mathbf{I} + \mathbf{I} \otimes c_y \mathbf{\Lambda})^{-1}\left(\mathbf{R}^{-1} \otimes \mathbf{R}^{-1}\right),$$

and we get

$$\mathbf{u}^x = (\mathbf{R} \otimes \mathbf{R})\,(c_x \mathbf{\Lambda} \otimes \mathbf{I} + \mathbf{I} \otimes c_y \mathbf{\Lambda})^{-1}\left(\mathbf{R}^{-1} \otimes \mathbf{R}^{-1}\right) \mathbf{f}^x.$$

To explicitly construct this matrix will require $\mathcal{O}(N^4)$ storage space and $\mathcal{O}(N^6)$ operations for finding $\mathbf{u}^x$. If we instead convert the system back to a local data structure we obtain the fast solver that we want. Notice that $\widehat{\mathbf{D}}$ is symmetric positive definite, and therefore $\mathbf{R}^{-1} = \mathbf{R}^T$. The fast algorithm for the Poisson problem with homogeneous Dirichlet boundary conditions is stated in **Algorithm 4**.

Algorithm 4 Fast Poisson solver: finite differences

```
1. $\mathbf{V} = \mathbf{R}^T\mathbf{F}\mathbf{R}$
2. $w_{ij} = v_{ij}/(c_x\lambda_i + c_y\lambda_j)$    for $i, j = 1, .., N-1$
3. $\mathbf{U} = \mathbf{R}\mathbf{W}\mathbf{R}^T$
```

Notice that the eigenvectors and eigenvalues here are different than those used when solving the Poisson problem with a spectral method. The complexity of finding $\mathbf{R}$ and $\mathbf{\Lambda}$ is $\mathcal{O}(N^3)$. The total cost of finding $\mathbf{U}$ is $\mathcal{O}(N^3)$ floating point operations for $\mathcal{O}(N^2)$ unknowns, similar to the spectral case.

3.8 The discrete L^2 norm and energy norm

Before we proceed with the numerical results we briefly discuss the discrete L^2 norm and the discrete energy norm [10] that will be used. The energy norm is induced by the bilinear form in (3.6). Notice that the bilinear form $a(\,\cdot\,,\,\cdot\,)$ of the

Poisson problem is symmetric

$$\begin{aligned} a(u,v) &= \int_0^{L_x}\int_0^{L_y} (u_x v_x + u_y v_y)\,\mathrm{d}x\,\mathrm{d}y \\ &= \int_0^{L_x}\int_0^{L_y} (v_x u_x + v_y u_y)\,\mathrm{d}x\,\mathrm{d}y \\ &= a(v,u) \qquad \forall\, u,v \in X \end{aligned}$$

and positive definite

$$a(w,w) = \int_0^{L_x}\int_0^{L_y} \left(w_x^2 + w_y^2\right)\mathrm{d}x\,\mathrm{d}y \ > 0 \qquad \forall w \in X,\ w \neq 0.$$

Given that a bilinear form $a(\,\cdot\,,\,\cdot\,)$ is symmetric positive definite (SPD), we can use it to define a norm $|\!|\!|\cdot|\!|\!|$ by

$$|\!|\!|w|\!|\!|^2 = a(w,w) \quad \forall w\, \in X,$$

and this is what we call the energy norm on X. The bilinear form is problem dependent. The discrete space X is not the same for different sets of boundary conditions, and clearly $a(u,v)$ is defined differently for different PDE's. Accordingly, the energy norm is also problem dependent.

Define the vector $\mathbf{x} = [x\ \ y]$. Consider the following expression of the error with respect to any norm

$$\begin{aligned} \|u_N(\mathbf{x}) - u(\mathbf{x})\| &= \|u_N(\mathbf{x}) - I_M u(\mathbf{x}) + I_M u(\mathbf{x}) - u(\mathbf{x})\| \\ &\le \|u_N(\mathbf{x}) - I_M u(\mathbf{x})\| + \|I_M u(\mathbf{x}) - u(\mathbf{x})\|. \end{aligned}$$

Here, $I_M u(\mathbf{x})$ is the interpolated function of $u(\mathbf{x})$ based on a finer grid than the grid used for u_N; $N \le M$. Later, in the numerical results, we choose $M = 3N$. It is reasonable to assume that when $M \gg N$, the interpolation error of the exact solution is much smaller than the discrete error:

$$\|I_M u(\mathbf{x}) - u(\mathbf{x})\| \ll \|u_N(\mathbf{x}) - u(\mathbf{x})\|.$$

The error with respect to a norm can thus be approximated by

$$\|u_N(\mathbf{x}) - u(\mathbf{x})\| \approx \|u_N(\mathbf{x}) - I_M u(\mathbf{x})\|.$$

Define u_{ij}^M to be the discrete solution at the finer grid

$$u_{ij}^M = \sum_{m=0}^{N}\sum_{n=0}^{N} u_{mn}\ell(\xi_i^M)\ell(\xi_j^M)$$

for $i, j = 0, .., M$, where the superscript M indicates the order of the finer grid. Define $u^M_{e_{ij}}$ to be the exact solution on the finer grid

$$u^M_{e_{ij}} = \hat{u}(\xi^M_i, \xi^M_j) = u(x(\xi^M_i), y(\xi^M_j)).$$

Then the discrete error at each point in the finer grid becomes

$$e^M_{ij} = u^M_{ij} - u^M_{e_{ij}}.$$

For simplicity, let $L_x = L_y = 2$. We then get the following expressions for the discrete L^2 norm error

$$\begin{aligned}
\|u_N(\mathbf{x}) - I_M u(\mathbf{x})\|^2_{L^2(\Omega)} &= \int_{-1}^{1}\int_{-1}^{1} (\hat{u}_N(\xi,\eta) - I_M\hat{u}(\xi,\eta))^2 \,\mathrm{d}\xi\,\mathrm{d}\eta \\
&= \int_{-1}^{1}\int_{-1}^{1} \left(\sum_{i=0}^{M}\sum_{j=0}^{M} e^M_{ij}\ell_i(\xi)\ell_j(\eta)\right)\left(\sum_{m=0}^{M}\sum_{n=0}^{M} e^M_{mn}\ell_m(\xi)\ell_n(\eta)\right)\mathrm{d}\xi \\
&= \sum_{i=0}^{M}\sum_{j=0}^{M}\sum_{m=0}^{M}\sum_{n=0}^{M} e^M_{ij} \underbrace{\left(\int_{-1}^{1}\ell_i(\xi)\ell_m(\xi)\,\mathrm{d}\xi\right)}_{(\ell_i(\xi),\ell_m(\xi))^1}\underbrace{\left(\int_{-1}^{1}\ell_j(\eta)\ell_n(\eta)\,\mathrm{d}\eta\right)}_{(\ell_j(\eta),\ell_n(\eta))^1} e^M_{mn} \\
&= \sum_{i=0}^{M}\sum_{j=0}^{M}\sum_{m=0}^{M}\sum_{n=0}^{M} e^M_{ij}\,(\ell_i(\xi),\ell_m(\xi))^1\,(\ell_j(\eta),\ell_n(\eta))^1\, e^M_{mn}.
\end{aligned}$$

Evaluating the inner products with GLL quadrature and applying (3.14) we obtain the discrete L^2 norm error

$$\|u_N(\mathbf{x}) - u(\mathbf{x})\|^2_{L^2_N(\widehat{\Omega})} = \sum_{i=0}^{M}\sum_{j=0}^{M}\sum_{m=0}^{M}\sum_{n=0}^{M} e^M_{ij}\underbrace{\left(\widehat{B}_{im}\widehat{B}_{jn}\right)}_{\widehat{B}^{2D}_{(ij)(mn)}} e^M_{mn}.$$

The reason for evaluating the error at a finer grid is to make sure that the quadrature error does not have a significant contribution [10]. Using global data representation we can express the discrete L^2 norm error in matrix form

$$\|u_N(\mathbf{x}) - u(\mathbf{x})\|^2_{L^2_N(\Omega)} = \left(\mathbf{u}^M - \mathbf{u}^M_e\right)^T \widehat{\mathbf{B}}^{2D}\left(\mathbf{u}^M - \mathbf{u}^M_e\right).$$

The superscript $2D$ indicates that the mass matrix is two-dimensional. A similar evaluation of the energy norm error gives

$$\|\!|u_N(\mathbf{x}) - u(\mathbf{x})|\!\|^2_N = \left(\mathbf{u}^M - \mathbf{u}^M_e\right)^T \widehat{\mathbf{A}}^{2D}\left(\mathbf{u}^M - \mathbf{u}^M_e\right).$$

The cost of evaluating these two expressions in global forms is $\mathcal{O}(N^4)$ for $\mathcal{O}(N^2)$ unknowns and the storage space required is $\mathcal{O}(N^4)$. If we instead use a local data representation the storage space required is $\mathcal{O}(N^2)$ and the operational cost is $\mathcal{O}(N^3)$. Again, using a local data representation is much more efficient.

3.9 Numerical results

In the following problems we will discuss both spectral discretization and finite differences solved with fast tensor-product solvers. First, let us consider spectral discretization for the following Poisson problem

$$\begin{aligned}
-\nabla^2 u &= \left(\frac{2\pi}{L_x}\right)^2 \cos\left(\frac{2\pi x}{L_x}\right)\left(1-\frac{y}{Ly}\right) && \text{in } \Omega,\\
\frac{\partial u}{\partial n} &= 0 && \text{on } \Gamma_2, \Gamma_4,\\
u(x,0) &= \cos\left(\frac{2\pi}{L_x}\right) && (\Gamma_1)\,,\\
u(x,L_x) &= 0 && (\Gamma_3)\,.
\end{aligned}$$

Here, Γ_i for $i = 1, ..., 4$ are the four edges of the deformed domain, see Figure 3.1. This is a problem with mixed boundary conditions, homogeneous Neumann, non-homogeneous and homogeneous Dirichlet. In the model problem we mainly discussed the simplest case, homogeneous Dirichlet boundary conditions. In this case the discrete space X_N will be

$$X_N = \left\{ v \circ \mathcal{F}(\xi,\eta) \in \mathbb{P}_N(\widehat{\Omega}), v = 0 \text{ on } \widehat{\Gamma}_1, \widehat{\Gamma}_3 \right\}.$$

The dimension of the algebraic system of equations is $N+1$ by $N-1$. In the linear form, we obtain an extra term since the numerical solution is nonzero at one of the boundaries with Dirichlet condition,

$$\begin{aligned}
\hat{u}_N(\xi,\eta) &= \sum_{m=0}^{N}\sum_{n=0}^{N-1} u_{mn}\,\ell_m(\xi)\,\ell_n(\eta)\\
&= \sum_{m=0}^{N}\sum_{n=1}^{N-1} u_{mn}\,\ell_m(\xi)\,\ell_n(\eta) + \underbrace{\sum_{m=0}^{N} u_{m0}\,\ell_m(\xi)\,\ell_n(\eta)}_{\text{given}}\,.
\end{aligned}$$

The bilinear and linear form can be expressed as

$$a(u_N, v)_N = \sum_{m=0}^{N}\sum_{n=1}^{N-1}\left(\frac{L_y}{L_x}\widehat{A}_{im}\widehat{B}_{jn} + \frac{L_x}{L_y}\widehat{B}_{im}\widehat{A}_{jn}\right) u_{mn}$$

and

$$(f,v)_N = \sum_{m=0}^{N}\sum_{n=1}^{N-1}\frac{L_yL_y}{4}\widehat{B}_{im}\widehat{B}_{jn} f_{mn} + \sum_{m=0}^{N}\left(\frac{L_y}{L_x}\widehat{A}_{im}\widehat{B}_{jn} + \frac{L_x}{L_y}\widehat{B}_{im}\widehat{A}_{jn}\right) u_{m0},$$

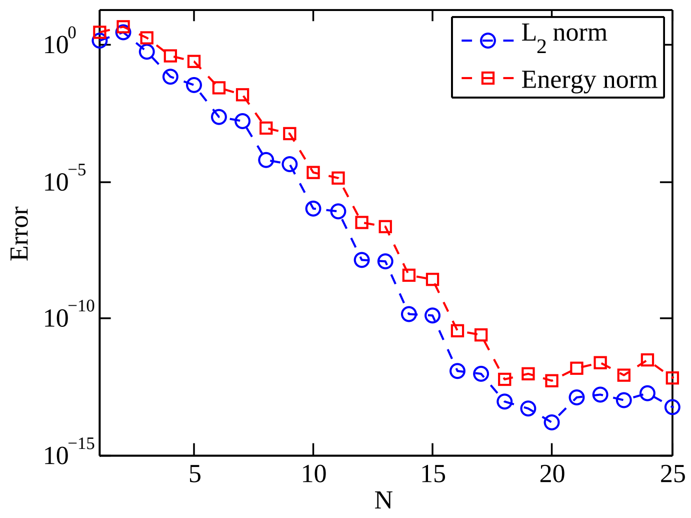


Figure 3.5: The discretization error $u(x) - u_N(x)$ measured in the discrete L^2 norm $\|\cdot\|_{L^2_N(\Omega)}$, and the discrete energy norm $|\!|\!|\cdot|\!|\!|_N$, as a function of the polynomial degree N. $u_N(x)$ is the discrete solution to the Poisson problem, where $f = \left(\frac{2\pi}{3}\right)^2 \cos\left(\frac{2\pi x}{3}\right)\left(1 - \frac{y}{2}\right)$ and u is subject to mixed boundary conditions in the domain $\Omega = (0,3) \times (0,2)$. The error is plotted on a logarithmic scale.

for $i = 0, ..., N+1$ and $j = 1, ..., N-1$, where the last term is given by the nonhomogeneous Dirichlet condition.

The system of equations, $a(u_N, v)_N = (f, v)_N$, are solved with **Algorithm 2**; the fast tensor-product solver with two generalized eigenvalue problems. Figure 3.6 graphs the analytical and the discrete solution to the problem in the physical domain $\Omega = (0,3) \times (0,2)$. Here, the polynomial degree of the discrete solution is $N = 10$, $u_N \in \mathbb{P}_{10}(\Omega)$. The discrete solution seems to be a good approximation of the analytical solution; they look the same.

In Figure 3.5 the error $u - u_N$ is plotted with respect to the discrete L^2 norm and the discrete energy norm as a function of the polynomial degree N. As expected, we obtain exponential convergence, since the exact solution u is infinitely smooth.

Next, we consider the following Poisson problem with homogeneous Dirichlet boundary conditions

$$\begin{aligned} -\nabla^2 u &= \frac{10\pi^2}{9}\sin(\pi x)\sin\left(\frac{\pi y}{3}\right) && \text{in } \Omega = (0,2) \times (0,3), \\ u &= 0 && \text{on } \partial\Omega. \end{aligned}$$

To solve this problem we used **Algorithm 4**, the fast solver for finite differences.

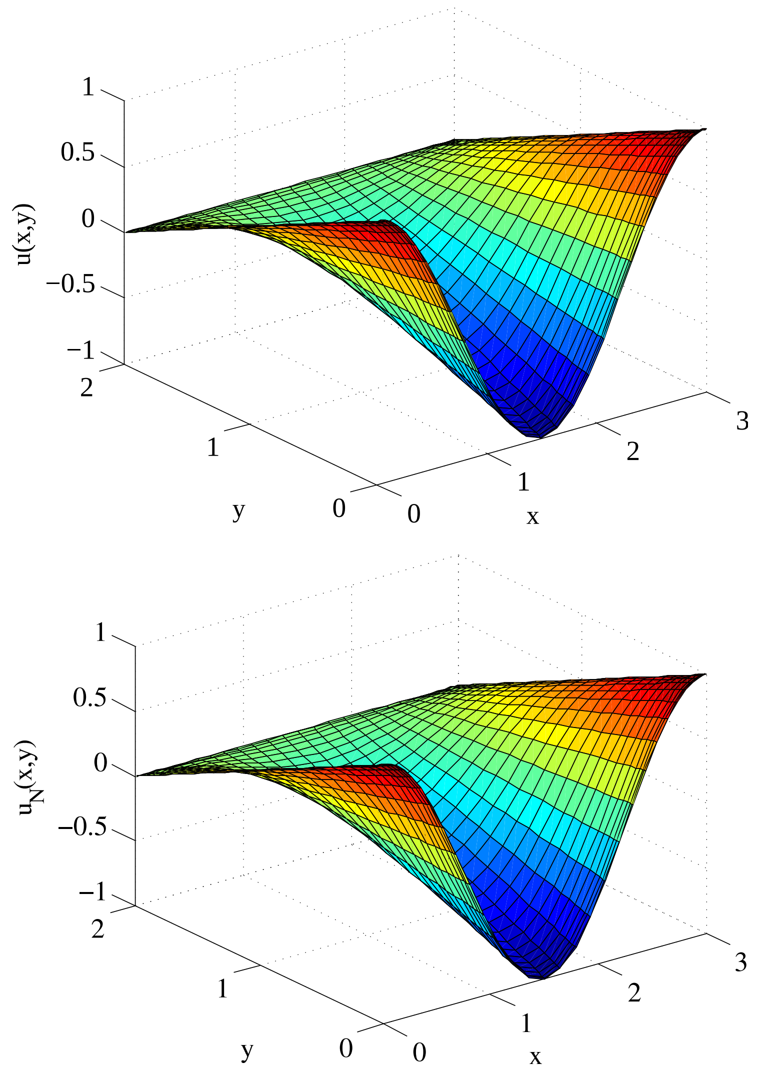


Figure 3.6: The analytical solution $u(x, y)$ at the top and the discrete solution $u_N(x, y)$ at the bottom of the Poisson problem, where $f = \left(\frac{2\pi}{3}\right)^2 \cos\left(\frac{2\pi x}{3}\right)\left(1 - \frac{y}{2}\right)$ and u is subject to mixed boundary conditions. The domain is $\Omega = (0, 3) \times (0, 2)$ and the polynomial degree of the discrete solution is 10, $u_N \in \mathbb{P}_{10}(\Omega)$.

Figure 3.7 graphs the discrete solution to the problem. There are $N + 1 = 11$ uniform grid points in each spatial direction. In addition, we solved the problem with a fast tensor-product solver for spectral discretization, **Algorithm 1**. In Figure 3.8 we compare the two methods by plotting the error $u - u_N$ with respect to the maximum norm as a function of N. Recall that in both methods there are $N + 1$ grid points in each spatial direction. Recall that in the spectral case the grid points are not uniformly spaced, and u_N is a polynomial of degree N. As expected, we obtain an algebraic convergence rate for the finite difference method

$$\max_{i,j} |\hat{u}_N(\xi_i, \xi_j) - \hat{u}(\xi_i, \xi_j)| \sim \mathcal{O}(N^{-2}).$$

On the other hand, we obtain an exponential convergence rate for the spectral method. For a fixed N (number of degrees of freedom) the cost of solving the system of algebraic equations is the same for the spectral method and the finite difference method. However, for a fixed N the error is smaller using a spectral method if the solution is smooth.

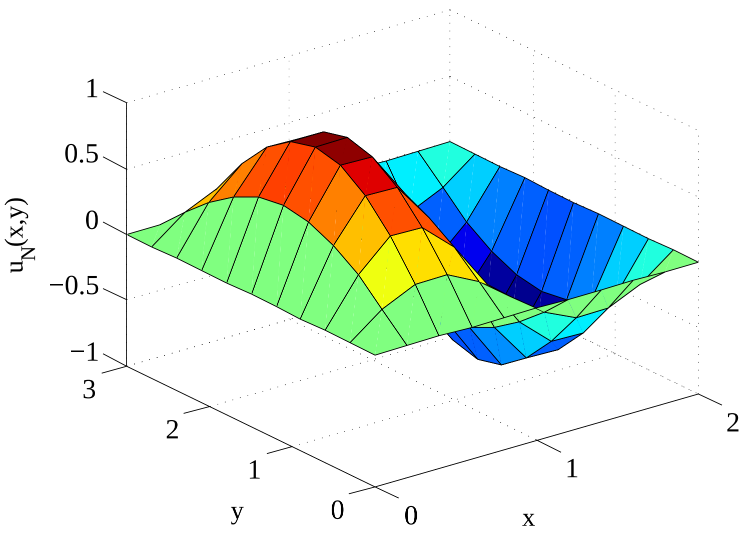


Figure 3.7: The discrete solution $u_N(x, y)$ of the Poisson problem with homogeneous Dirichlet boundary conditions and $f = \frac{10\pi^2}{9} \sin(2\pi x) \sin\left(\frac{\pi x}{3}\right)$. The problem is solved using central differences with a uniform 11×11-grid in the domain $\Omega = (0, 2) \times (0, 3)$.

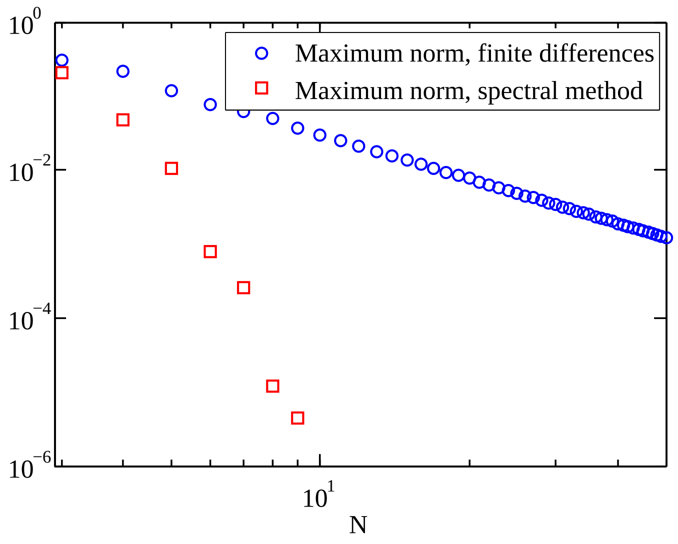


Figure 3.8: The discretization error $u(x)-u_N(x)$ for both the spectral method and the finite difference method measured in the maximum norm as a function of the degrees of freedom N. $u_N(x)$ is the discrete solution to the Poisson problem, where $f = \frac{10\pi^2}{9}\sin(2\pi x)\sin\left(\frac{\pi x}{3}\right)$ subject to homogenous Dirichlet boundary conditions in the domain $\Omega = (0,2)\times(0,3)$. Both axes has a logarithmic scale.

Chapter 4

Exploring tensor-product solvers in deformed domains

So far we have only discussed fast tensor-product solvers in rectangular domains. A naturally arising question is: can one extend this theory to deformed domains? In this chapter we explore the possibility of exploiting fast tensor-product solvers in a two dimensional deformed domain. First, we consider the mapping of the reference domain into the deformed domain, and then the spectral discretization of the Poisson problem. We will shortly discuss the conjugate gradient methods as a way of solving the derived system of equations. Again, we observe that the local data representation is essential for deriving a fast solver.

4.1 The reference domain

The mapping of the reference domain is no longer just a stretching and translation, see figure 4.1. Now x and y are functions of both ξ and η,

$$\begin{aligned} x &= x(\xi,\eta), & \mathrm{d}x &= \frac{\partial x}{\partial \xi}\mathrm{d}\xi + \frac{\partial x}{\partial \eta}\mathrm{d}\eta, \\ y &= y(\xi,\eta), & \mathrm{d}y &= \frac{\partial y}{\partial \xi}\mathrm{d}\xi + \frac{\partial y}{\partial \eta}\mathrm{d}\eta. \end{aligned}$$

In matrix form this can be written as

$$\begin{pmatrix} \mathrm{d}x \\ \mathrm{d}y \end{pmatrix} = \underbrace{\begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix}}_{\mathbf{J}} \begin{pmatrix} \mathrm{d}\xi \\ \mathrm{d}\eta \end{pmatrix}, \quad J = \det(\mathbf{J}) = x_\eta y_\eta - x_\eta y_\xi, \tag{4.1}$$

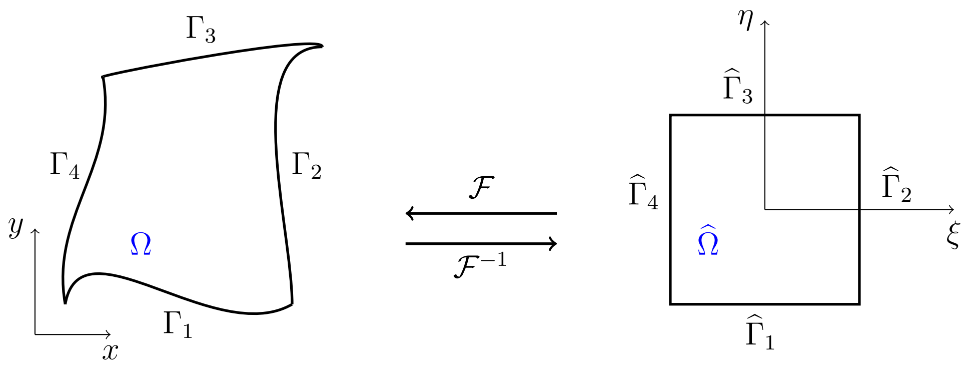


Figure 4.1: The mapping of the reference domain $\widehat{\Omega} = (-1, 1) \times (-1, 1)$ to the deformed physical domain Ω, $\mathcal{F}(\xi, \eta) = (x, y)$

where $\mathbf{J}$ is the Jacobian matrix and J is the determinant of $\mathbf{J}$. Similarly, ξ and η are functions of x and y, and in matrix form we write

$$\begin{pmatrix} \mathrm{d}\xi \\ \mathrm{d}\eta \end{pmatrix} = \begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix} \begin{pmatrix} \mathrm{d}x \\ \mathrm{d}y \end{pmatrix}. \tag{4.2}$$

Alternatively, (4.1) allow us to express these variables as

$$\begin{pmatrix} \mathrm{d}\xi \\ \mathrm{d}\eta \end{pmatrix} = \mathbf{J}^{-1} \begin{pmatrix} \mathrm{d}x \\ \mathrm{d}y \end{pmatrix} = \frac{1}{J} \begin{pmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{pmatrix} \begin{pmatrix} \mathrm{d}x \\ \mathrm{d}y \end{pmatrix}. \tag{4.3}$$

Combining (4.2) and (4.3) we get that

$$\begin{aligned} \xi_x &= \frac{1}{J} y_\eta, \qquad & \xi_y &= -\frac{1}{J} x_\eta, \\ \eta_x &= -\frac{1}{J} y_\xi, \qquad & \eta_y &= \frac{1}{J} x_\xi. \end{aligned} \tag{4.4}$$

This relations will turn out to be of great significance. Another important relation is the following property of the determinant of the Jacobian matrix $\mathbf{J}$ [10]

$$J = \det(\mathbf{J}) = \frac{\mathrm{d}\Omega}{\mathrm{d}\widehat{\Omega}}. \tag{4.5}$$

In the deformed domain, J will change throughout the whole domain, contrary to the rectangular case where $J = \frac{L_x L_y}{4}$ is a constant.

4.2 The weak formulation of the Poisson problem

In the section concerning rectangular domains we found the following weak formulation of our model Poisson problem with homogeneous Dirichlet boundary conditions: find $u \in X$ such that

$$\underbrace{\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x \, \mathrm{d}y}_{a(u,v)} = \underbrace{\int_\Omega f \, v \, \mathrm{d}x \, \mathrm{d}y}_{(f,v)} \quad \forall v \in X.$$

Let us first consider the gradient in the bilinear form

$$\nabla u = \begin{pmatrix} u_x \\ u_y \end{pmatrix} = \begin{pmatrix} \hat{u}_\xi \xi_x + \hat{u}_\eta \eta_x \\ \hat{u}_\xi \xi_y + \hat{u}_\eta \eta_y \end{pmatrix} = \underbrace{\begin{pmatrix} \xi_x & \eta_x \\ \xi_y & \eta_y \end{pmatrix}}_{\mathbf{G}_\nabla} \underbrace{\begin{pmatrix} \hat{u}_\xi \\ \hat{u}_\eta \end{pmatrix}}_{\widehat{\nabla} \hat{u}} = \mathbf{G}_\nabla \widehat{\nabla} \hat{u}. \tag{4.6}$$

Here, ∇u is the gradient with respect to the physical variables and $\widehat{\nabla} \hat{u}$ is the gradient with respect to the reference variables. Making use of the derived expressions in (4.4), we can express the matrix in (4.6) as

$$\mathbf{G}_\nabla = \frac{1}{J} \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix}. \tag{4.7}$$

With the relations in (4.5) and (4.6) the bilinear form can be expressed as

$$\begin{aligned} \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x \, \mathrm{d}y &= \int_\Omega (\nabla v)^T \nabla u \, \mathrm{d}x \, \mathrm{d}y \\ &= \int_{\widehat{\Omega}} (\mathbf{G}_\nabla \widehat{\nabla} \hat{v})^T (\mathbf{G}_\nabla \widehat{\nabla} \hat{u}) J \, \mathrm{d}\xi \, \mathrm{d}\eta \\ &= \int_{\widehat{\Omega}} (\widehat{\nabla} \hat{v})^T \underbrace{(\mathbf{G}_\nabla^T \mathbf{G}_\nabla)}_{\mathbf{G}} \widehat{\nabla} \hat{u} \, J \, \mathrm{d}\xi \, \mathrm{d}\eta \\ &= \int_{\widehat{\Omega}} (\widehat{\nabla} \hat{v})^T \mathbf{G} \widehat{\nabla} \hat{u} \, J \, \mathrm{d}\xi \, \mathrm{d}\eta. \end{aligned}$$

Note that the matrix $\mathbf{G}_\nabla$ is not symmetric. However, $\mathbf{G}$ is a symmetric matrix:

$$\mathbf{G} = \mathbf{G}_\nabla^T \mathbf{G}_\nabla.$$

We defining $\widetilde{\mathbf{G}} \equiv J\mathbf{G} = J\mathbf{G}_\nabla^T \mathbf{G}_\nabla$, where $\mathbf{G}_\nabla$ is expressed in (4.7). The matrix $\widetilde{\mathbf{G}}$ has the following matrix elements

$$\begin{aligned} \tilde{g}_{11} &= \frac{1}{J}(y_\eta^2 + x_\eta^2) \\ \tilde{g}_{12} = \tilde{g}_{21} &= -\frac{1}{J}(y_\xi y_\eta + x_\eta x_\xi) \\ \tilde{g}_{22} &= \frac{1}{J}(y_\xi^2 + x_\xi^2). \end{aligned} \tag{4.8}$$

The weak formulation can now be stated as: find $u \in X$ such that

$$\int_{\widehat{\Omega}} (\widehat{\nabla} \hat{v})^T \widetilde{\mathbf{G}} \widehat{\nabla} \hat{u} \;\, \mathrm{d}\xi \, \mathrm{d}\eta = \int_{\widehat{\Omega}} \hat{f} \, \hat{v} \, J \;\, \mathrm{d}\xi \, \mathrm{d}\eta \qquad \forall v \in X. \tag{4.9}$$

4.3 Isoparametric mapping

Let us now consider the mapping from the reference domain $\widehat{\Omega}$ to the deformed domain Ω using an isoparametric mapping

$$x_N(\xi,\eta) = \sum_{i=0}^{N}\sum_{j=0}^{N} x_{ij}\, \ell_i(\xi)\, \ell_j(\eta) \in \mathbb{P}_N(\widehat{\Omega}),$$
$$y_N(\xi,\eta) = \sum_{i=0}^{N}\sum_{j=0}^{N} y_{ij}\, \ell_i(\xi)\, \ell_j(\eta) \in \mathbb{P}_N(\widehat{\Omega}),$$

where $x_{ij} = x(\xi_i,\xi_j)$ and $y_{ij} = y(\xi_i,\xi_j)$. We call this an isoparametric mapping since the geometry is approximated in the same way as the solution. In most cases, the solution is more complex than the geometry. Therefore, using an isoparametric mapping will not contribute significantly to the error in the numerical solution.

4.4 Discretization and computational approach

In this section we will discuss different aspects of the implementation to solve the discrete Poisson problem [10]. According to the isoparametric mapping, the partial derivative of $x_N(\xi_\alpha,\xi_\beta)$ can be expressed as

$$\begin{aligned}\left.\frac{\partial x_N}{\partial \xi}\right|_{(\xi_\alpha,\xi_\beta)} &= \sum_{i=0}^{N}\sum_{j=0}^{N} x_{ij}\, \ell_i'(\xi_\alpha)\, \ell_j(\xi_\beta)\\ &= \sum_{i=0}^{N+1}\sum_{j=0}^{N+1} \ell_i'(\xi_\alpha)\, x_{ij}\, \delta_{\beta j}\\ &= \sum_{i=0}^{N+1} \ell_i'(\xi_\alpha)\, x_{i\beta}.\end{aligned}$$

Define the following matrices

D with matrix elements, $D_{ij} = l_j'(\xi_i)\ ,\quad i,j = 0,...,N.$

X with matrix elements, $x_{ij} = x(\xi_i,\xi_j),\quad i,j = 0,...,N.$

$\mathbf{X}_{,\xi}$ with matrix elements, $(X_{,\xi})_{ij} = \left.\frac{\partial x_N}{\partial \xi}\right|_{(\xi_i,\xi_j)},\quad i,j = 0,...,N.$

The above expression can then be written as

$$(X_{,\xi})_{\alpha\beta} = \sum_{i=0}^{N+1} D_{\alpha i} x_{i\beta},$$

or in matrix form as

$$\mathbf{X}_{,\xi} = \mathbf{D}\mathbf{X}. \tag{4.10}$$

Thus, the partial derivative with respect to ξ for all points $i, j = 0, ..., N$ is evaluated by one matrix-matrix multiplication using local data structure. The computational cost is $\mathcal{O}(N^3)$. Define $\mathbf{X}_{,\eta}$, $\mathbf{Y}$, $\mathbf{Y}_{,\xi}$, $\mathbf{Y}_{,\eta}$, $\mathbf{U}$, $\mathbf{U}_{,\xi}$, and $\mathbf{U}_{,\eta}$ in the same way, and the following results are obtained:

$$\mathbf{X}_{,\eta} = \mathbf{X}\mathbf{D}^T, \tag{4.11}$$
$$\mathbf{Y}_{,\xi} = \mathbf{D}\mathbf{Y}, \tag{4.12}$$
$$\mathbf{Y}_{,\eta} = \mathbf{Y}\mathbf{D}^T, \tag{4.13}$$
$$\mathbf{U}_{,\xi} = \mathbf{D}\mathbf{U}, \tag{4.14}$$
$$\mathbf{U}_{,\eta} = \mathbf{U}\mathbf{D}^T. \tag{4.15}$$

From the results above we can evaluate the Jacobian J in (4.1), and the matrix $\widetilde{\mathbf{G}}$ at the GLL grid points

$$J_{ij} = (X_{,\xi})_{ij}(Y_{,\eta})_{ij} - (X_{,\eta})_{ij}(Y_{,\xi})_{ij},$$

$$\widetilde{\mathbf{G}}_{ij} = \begin{pmatrix} \frac{1}{J_{ij}}\left((Y_{,\eta})_{ij}^2 + (X_{,\eta})_{ij}^2\right) & -\frac{1}{J_{ij}}\left((Y_{,\xi})_{ij}(Y_{,\eta})_{ij} + (X_{,\eta})_{ij}(X_{,\xi})_{ij}\right) \\ -\frac{1}{J_{ij}}\left((Y_{,\xi})_{ij}(Y_{,\eta})_{ij} + (X_{,\eta})_{ij}(X_{,\xi})_{ij}\right) & \frac{1}{J_{ij}}\left((Y_{,\xi})_{ij}^2 + (X_{,\xi})_{ij}^2\right) \end{pmatrix},$$

for $i, j = 1, ..., N-1$. Let us now consider the discrete formulation of (4.9): find $u_N \in X_N$ such that

$$\int_{\widehat{\Omega}} (\widehat{\nabla}\hat{v})^T \widetilde{\mathbf{G}} \widehat{\nabla}\hat{u}_N \, \mathrm{d}\xi \, \mathrm{d}\eta = \int_{\widehat{\Omega}} \hat{f}\, \hat{v} J \, \mathrm{d}\xi \, \mathrm{d}\eta \qquad \forall v \in X_N, \tag{4.16}$$

where X_N is defined in (3.9). We are claiming that (4.16) holds for all $v \in X_N$, so with the same tensor-product bases as before we let

$$\hat{v}(\xi,\eta) = \ell_i(\xi)\ell_j(\eta), \qquad 1 \le i, j \le N-1$$

and

$$\hat{u}_N(\xi,\eta) = \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} u_{mn}\, \ell_m(\xi)\, \ell_n(\eta).$$

With GLL quadrature we obtain

$$\begin{aligned}\int_{\widehat{\Omega}} \hat{f}\,\hat{v}\,J\,\mathrm{d}\xi\,\mathrm{d}\eta &\simeq \sum_{\alpha=0}^{N}\sum_{\beta=0}^{N} \rho_\alpha\,\rho_\beta\,f_{\alpha\beta}\,\delta_{i\alpha}\,\delta_{j\beta}\,J_{\alpha\beta}\\ &= \rho_i\,\rho_j\,J_{ij}\,f_{ij}\end{aligned}$$

and

$$\int_{\widehat{\Omega}} (\widehat{\nabla}\hat{v})^T\widetilde{\mathbf{G}}\widehat{\nabla}\hat{u}_N\,\mathrm{d}\xi\mathrm{d}\eta \simeq \sum_{\alpha=0}^{N}\sum_{\beta=0}^{N}\rho_\alpha\rho_\beta\begin{pmatrix}D_{\alpha i}\delta_{\beta j} & \delta_{\alpha i}D_{\beta j}\end{pmatrix}\begin{pmatrix}\tilde{g}_{11} & \tilde{g}_{12}\\ \tilde{g}_{21} & \tilde{g}_{22}\end{pmatrix}_{\alpha\beta}\begin{pmatrix}U_{,\xi}\\ U_{,\eta}\end{pmatrix}_{\alpha\beta}.$$

Recall that $(U_{,\xi})_{\alpha\beta} = \frac{\partial u_N}{\partial \xi}\Big|_{(\xi_\alpha,\xi_\beta)}$ and $(U_{,\eta})_{\alpha\beta} = \frac{\partial u_N}{\partial \eta}\Big|_{(\xi_\alpha,\xi_\beta)}$. We have now found an expression for the discrete problem. In the rectangular domain we used a fast tensor-product solver to solve the system of equations. We could do this because the Jacobian was constant. Now however, the Jacobian changes in every point in space. The system above can be written in the form

$$\mathbf{A}^{2D}\mathbf{u}^x = \mathbf{f}^{2D}. \tag{4.17}$$

Iterative methods are typical approaches for solving this system. The stiffness matrix is symmetric, allowing for the conjugate gradient method (CGM) [13] to be used for solving (4.17). The most costly operation in the CGM is the operation $\mathbf{A}^{2D}\mathbf{u}^x$. The key here is to take advantage of the local data structure. If we constructed $\mathbf{A}^{2D}$, $\mathcal{O}(N^4)$ storage space and $\mathcal{O}(N^4)$ floating point operations are required to evaluate the expression. If we instead exploit local data structure, this can be evaluated in $\mathcal{O}(N^3)$ operations and the storage space required is only $\mathcal{O}(N^2)$. The algorithm for evaluating $\mathbf{A}^{2D}\mathbf{u}^x$ with local data structure is:

1. Preprocessing: Compute $\widetilde{\mathbf{G}}_{\alpha\beta}$ and $\mathbf{D}$ (Only needs to be evaluated once)

2. $\left(U_{,\xi}\right)_{\alpha\beta} = \sum_{m=0}^{N} D_{\alpha m} u_{m\beta}$, one matrix-matrix product $\mathbf{U}_{,\xi} = \mathbf{DU}$

   $\left(U_{,\eta}\right)_{\alpha\beta} = \sum_{n=0}^{N} u_{\alpha n} D_{\beta n}$, one matrix-matrix product $\mathbf{U}_{,\eta} = \mathbf{UD}^T$

3. $(s_1)_{\alpha\beta} = (\tilde{g}_{11})_{\alpha\beta}\ \left(U_{,\xi}\right)_{\alpha\beta} + (\tilde{g}_{12})_{\alpha\beta}\ \left(U_{,\eta}\right)_{\alpha\beta}$
   $(s_2)_{\alpha\beta} = (\tilde{g}_{21})_{\alpha\beta}\ \left(U_{,\xi}\right)_{\alpha\beta} + (\tilde{g}_{22})_{\alpha\beta}\ \left(U_{,\eta}\right)_{\alpha\beta}$

4. $(t_1)_{\alpha\beta} = (s_1)_{\alpha\beta}\, \rho_\alpha \rho_\beta$

   $(t_2)_{\alpha\beta} = (s_2)_{\alpha\beta}\, \rho_\alpha \rho_\beta$

5. $w_{ij} = \sum_{\alpha=0}^{N} D_{\alpha i}\,(t_1)_{\alpha j} + \sum_{\beta=0}^{N}(t_2)_{i\beta} D_{\beta j}$, $\underbrace{\text{two matrix-matrix products}}_{\mathbf{W} = \mathbf{D}^T\,\mathbf{T}_1 + \mathbf{T}_2\,\mathbf{D}}$

Step 2 and 5 require $\mathcal{O}(N^3)$ floating point operations, while steps 3 and 4 are evaluated with $\mathcal{O}(N^2)$ floating point operations. The total work for the algorithm is $\mathcal{O}(N^3)$ if we exclude step 1. Step 1 only needs to be evaluated once.

If the geometry is not too complex a preconditioned conjugate gradient method can be a good choice for solving the system. The preconditioner would then be the solution of the problem solved in a rectangular domain that is similar to the deformed domain. A fast tensor-product solver can then be used to find the preconditioned solution in $\mathcal{O}(N^3)$ operations and the CGM will converge faster.

4.5 Gordon-Hall algorithm

In the previous discussions we assumed x_{ij} and y_{ij} to be known in the isoparametric mappings

$$\begin{aligned} x_N &= \sum_{i=0}^{N} x_{ij}\, \ell_i(\xi)\, \ell_j(\eta), \\ y_N &= \sum_{i=0}^{N} y_{ij}\, \ell_i(\xi)\, \ell_j(\eta). \end{aligned} \tag{4.18}$$

Thus, we need a mapping from the reference domain $\widehat{\Omega}$ to the physical domain Ω. We use the Gordon Hall-algorithm to find this mapping [14]. First, we define the

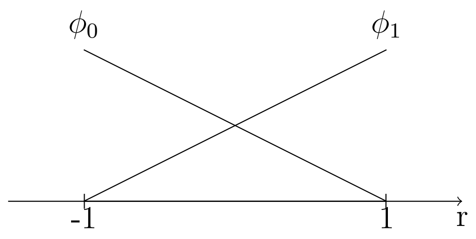


Figure 4.2: The linear Lagrange interpolation functions through $r_0 = -1$ and $r_1 = 1$

linear shape functions

$$\phi_0(r) = \frac{1-r}{2},$$
$$\phi_1(r) = \frac{1+r}{2}.$$

Let $r_0 = -1$ and $r_1 = 1$. The shape functions then have the following properties: $\phi_i(r_j) = \delta_{ij}$ for $i, j = 0, 1$. This is in fact the linear Lagrange interpolation functions through r_0 and r_1, see Figure 4.2. Let the edges of the physical domain be denoted by

$$\begin{aligned} &\Gamma_1 : \mathbf{x}(\xi, -1),\\ &\Gamma_2 : \mathbf{x}(1, \eta),\\ &\Gamma_3 : \mathbf{x}(\xi, 1),\\ &\Gamma_4 : \mathbf{x}(-1, \eta), \end{aligned} \tag{4.19}$$

where $\mathbf{x}(\xi,\eta) = \begin{bmatrix} x(\xi,\eta) & y(\xi,\eta) \end{bmatrix}$. Consider the following four mappings:

1. $\mathcal{F}_\xi(\xi,\eta) = \phi_0(\xi)\mathbf{x}(-1,\eta) + \phi_1(\xi)\mathbf{x}(1,\eta)$.

2. $\mathcal{F}_\eta(\xi,\eta) = \phi_0(\xi)\mathbf{x}(\xi,-1) + \phi_1(\xi)\mathbf{x}(\xi,1)$.

3. $\mathcal{F}_{\xi\eta}(\xi,\eta) = \sum_{i=0}^{1}\sum_{j=0}^{1}\phi_i(\xi)\,\phi_j(\eta)\mathbf{x}(r_i,r_j)$.

4. $\mathcal{F}^{GH} = \mathcal{F}_\xi + \mathcal{F}_\eta - \mathcal{F}_{\xi\eta}$.

The three first mappings are illustrated in Figure 4.3. The mapping $\mathcal{F}_\xi$ conserves the two edges Γ_1 and Γ_3, while the mapping $\mathcal{F}_\eta$ conserves Γ_2 and Γ_4. $\mathcal{F}_\eta$ and $\mathcal{F}_\xi$ are linear interpolations between two edges. Further, $\mathcal{F}_{\xi\eta}$ conserves the four corner points.

The final step $\mathcal{F}^{GH}$ preserves all four edges and maps the interior of the reference domain $\widehat{\Omega}$ to the interior of the physical domain Ω. This is known as the Gordon-

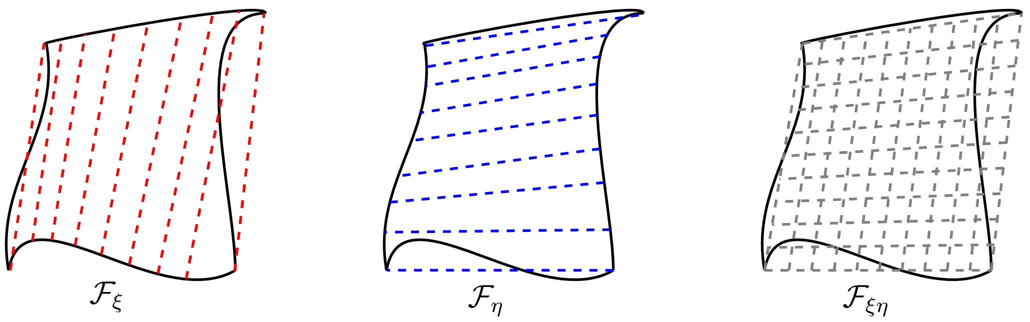


Figure 4.3: Illustration of the three mappings $\mathcal{F}_\xi$, $\mathcal{F}_\eta$ and $\mathcal{F}_{\xi\eta}$ that are used in the Gordon-Hall algorithm $\mathcal{F}^{GH} = \mathcal{F}_\xi + \mathcal{F}_\eta - \mathcal{F}_{\xi\eta}$.

Hall algorithm.

We now return to our original problem, namely finding $\mathbf{x}_{ij} = [x_{ij} \quad y_{ij}]$ in (4.18). We distribute $N+1$ points along each edge of the domain, similar to the GLL points in the rectangular domain. There are several ways to do this. One option is to draw a straight line between the four corners and distribute the GLL points along each edge. The GLL points on the deformed edge are obtained by drawing a cord from the GLL points on the straight line, as illustrated in Figure 4.4. We

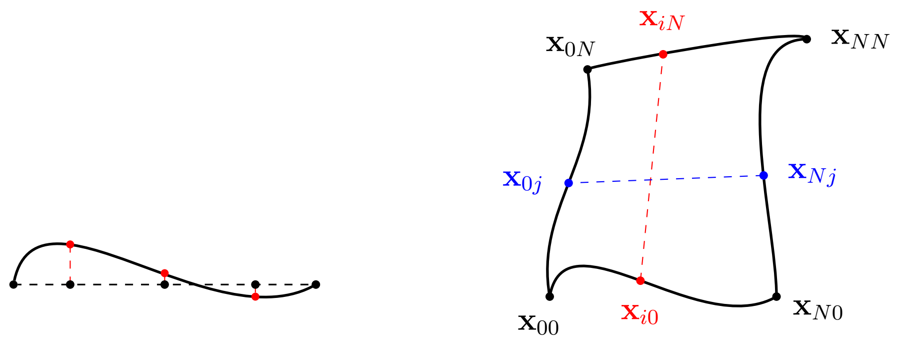


Figure 4.4: The figure to the left illustrates how the GLL points are obtained on one of the deformed edges. The GLL points are distributed along the straight line and we obtain the GLL points on the deformed edge by drawing a cord to the edge. The figure to the right illustrates the numbering of the GLL grid points.

then have sufficient information to compute $\mathbf{x}_{ij}$ for $i, j = 0, ...N$ using the following algorithm:

1. $\mathbf{x}_{ij}^A = \phi_0(\xi_i)\mathbf{x}_{0j} + \phi_1(\xi_i)\mathbf{x}_{Nj}$.
2. $\mathbf{x}_{ij}^B = \phi_0(\xi_j)\mathbf{x}_{i0} + \phi_1(\xi_j)_{iN}$.
3. $\mathbf{x}_{ij}^C = \phi_0(\xi_i)\phi_0(\xi_j)\mathbf{x}_{00} + \phi_0(\xi_i)\phi_1(\xi_j)\mathbf{x}_{0N} + \phi_1(\xi_i)\phi_0(\xi_j)\mathbf{x}_{N0} + \phi_1(\xi_i)\phi_1(\xi_j)\mathbf{x}_{NN}$.
4. $\mathbf{x}_{ij} = \mathbf{x}_{ij}^A + \mathbf{x}_{ij}^B - \mathbf{x}_{ij}^C$.

Making use of this information, the isoparametric mapping

$$\mathbf{x}_N(\xi,\eta) = \sum_{i=0}^{N}\sum_{j=0}^{N} \mathbf{x}_{ij}\,\ell_i(\xi)\,\ell_j(\eta)$$

approximates the edges of the domain with high order polynomials of the same degree as the approximated solution

$$\hat{u}_N(\xi,\eta) = \sum_{i=0}^{N}\sum_{j=0}^{N} u_{ij}\,\ell_i(\xi)\,\ell_j(\eta).$$

4.6 An idea for a tensor-product solver

So far we have only discussed the conjugate gradient method as a way of solving the algebraic system of equations. We will now explore the possibility of deriving a fast tensor-product solver similar to the ones we discussed for the rectangular domain. The motivation of deriving such a method is obvious: it would reduce the computational costs.

Recall the weak form of the Poisson problem with homogeneous Dirichlet boundary conditions in a deformed domain: find $u \in X$ such that

$$\int_{\widehat{\Omega}} (\widehat{\nabla}\hat{v})^T \widetilde{\mathbf{G}}\,\widehat{\nabla}\hat{u}\,\mathrm{d}\xi\,\mathrm{d}\eta = \int_{\widehat{\Omega}} \hat{f}\,\hat{v} J\,\mathrm{d}\xi\,\mathrm{d}\eta \qquad \forall\, v \in X, \tag{4.20}$$

where X is defined in (3.4) and $\widetilde{\mathbf{G}} = J\,\mathbf{G}_\nabla^T\mathbf{G}_\nabla$. First define the matrix

$$\mathbf{G}_\nabla^* = \sqrt{J}\,\mathbf{G}_\nabla, \tag{4.21}$$

and the two transformations

$$\widehat{\nabla}\hat{u}^* = \mathbf{G}_\nabla^*\widehat{\nabla}\hat{u} \tag{4.22}$$

and

$$\widehat{\nabla}\hat{v}^* = \mathbf{G}_\nabla^*\widehat{\nabla}\hat{v}. \tag{4.23}$$

Then consider the following equation

$$\begin{aligned}\int_{\widehat{\Omega}} (\widehat{\nabla}\hat{v}^*)^T\,\widehat{\nabla}\hat{u}^*\,\mathrm{d}\xi\,\mathrm{d}\eta &= \int_{\widehat{\Omega}} (\mathbf{G}_\nabla^*\widehat{\nabla}\hat{v})^T(\mathbf{G}_\nabla^*\widehat{\nabla}\hat{u})\,\mathrm{d}\xi\,\mathrm{d}\eta \\ &= \int_{\widehat{\Omega}} (\widehat{\nabla}\hat{v})^T\,\underbrace{\mathbf{G}_\nabla^{*\,T}\mathbf{G}_\nabla^*}_{J\mathbf{G}_\nabla^T\mathbf{G}_\nabla=\widetilde{\mathbf{G}}}\,\widehat{\nabla}\hat{u}\,\mathrm{d}\xi\,\mathrm{d}\eta \\ &= \int_{\widehat{\Omega}} (\widehat{\nabla}\hat{v})^T\widetilde{\mathbf{G}}\widehat{\nabla}\hat{u}\,\mathrm{d}\xi\,\mathrm{d}\eta.\end{aligned}$$

The last expression is equivalent to the left side of (4.20). We can thus consider the generalized weak formulation: find $u^* \in X$ such that

$$\int_{\widehat{\Omega}} (\widehat{\nabla}\hat{v}^*)^T\widehat{\nabla}\hat{u}^*\,\mathrm{d}\xi\,\mathrm{d}\eta = \int_{\widehat{\Omega}} \hat{f}\hat{v}J\,\mathrm{d}\xi\,\mathrm{d}\eta \qquad \forall v^* \in X. \tag{4.24}$$

The bilinear form is equivalent to the bilinear form in the rectangular domain, except that $\hat{u}^*$ is considered instead of $\hat{u}$. The idea is to discretize this weak formulation similarly to how it was done in the rectangular domain, and apply a fast tensor-product solver to find $\hat{u}_N^*$. If possible, we then find the solution $\hat{u}_N$ from the transformation (4.22).

We choose $\hat{v}^* = \ell_i(\xi)\ell_j(\eta)$ for $i, j = 1, ..., N-1$ and approximate $\hat{u}^*$ as before,

$$\hat{u}_N^*(\xi,\eta) = \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} u_{mn}^*\,\ell_m(\xi)\,\ell_n(\eta)$$

The system of equations can in global form be written as

$$(\widehat{\mathbf{A}} \otimes \widehat{\mathbf{B}} + \widehat{\mathbf{B}} \otimes \widehat{\mathbf{A}})\mathbf{u}^* = \mathbf{b}^*. \tag{4.25}$$

This system can be solved with a fast tensor-product solver if $\mathbf{b}^*$ is known. Assume $\mathbf{b}^*$ is known. The possible fast solver can then be broken down to four steps, as described in **Algorithm 5**.

Algorithm 5 An idea for a fast Poisson solver in deformed domains

```
1. Find $\mathbf{U}^*$ in (4.25) with the fast Poisson solver for Dirichlet conditions.
2. Find the partial derivatives $\mathbf{U}^*_{,\xi}$ and $\mathbf{U}^*_{,\eta}$, using the information from $\mathbf{U}^*$.
3. Find one of the partial derivatives $\mathbf{U}_{,\xi}$ and $\mathbf{U}_{,\eta}$, using the relation (4.22)
   combined with $\mathbf{U}^*_{,\xi}$ and $\mathbf{U}^*_{,\eta}$.
4. Find $\mathbf{U}$, using the information from $\mathbf{U}_{,\xi}$ or $\mathbf{U}_{,\eta}$.
```

We will go through all the four steps and see if we can realize a fast solver for the Poisson problem in a deformed domain.

4.6.1 Step 4 of Algorithm 5

To succeed at finding a fast algorithm in a deformed domain all steps are equally important. We start by consider the fourth step, since this step can be tested numerically without any information from the previous steps. We know the value of the derivative at the $(N+1)$ GLL points and want to find the function value at the same points. For simplicity let us consider a simple example in $\mathbb{R}$:

$$\begin{aligned} v(x) &= 1 - x^2 \qquad \text{in } \Omega = (-1,1), \\ v(-1) &= 0, \\ v(1) &= 0. \end{aligned} \tag{4.26}$$

The derivative of this function is $v_x = 2x$. The function is approximated (or exactly represented) as a polynomial of degree N in the one dimensional reference domain $\widehat{\Omega} = \Omega = (-1,1)$

$$v_N(x(\xi)) = \hat{v}_N(\xi) = \sum_{m=0}^{N} v_m \ell_m(\xi), \tag{4.27}$$

where v_m are the nodal values and $\ell_m(\xi)$ the basis functions for $m = 0, ..., N$. The nodal values are unknown, except at the boundary, while the derivatives at all the $N+1$ points are known. The derivative of v_N at the $N+1$ GLL points is

$$\begin{aligned} \left.\frac{\partial \hat{v}_N}{\partial \xi}\right|_{\xi_i} &= \sum_{m=0}^{N} v_m \underbrace{\ell'_m(\xi_i)}_{D_{im}} \\ &= \sum_{m=0}^{N} v_m D_{im} \end{aligned} \tag{4.28}$$

for $i = 0, ..., N$. In matrix form we write

$$\mathbf{v}_{,\xi} = \mathbf{D}\mathbf{v}.$$

The matrix $\mathbf{D}$ is singular and has dimension $(N+1) \times (N+1)$. The unknown vector $\mathbf{v}$ can thus not be found by taking the inverse of $\mathbf{D}$. We want to find an algorithm that returns a unique vector $\mathbf{v}$. First we consider a simple example that illustrates the importance of using the right information from the derivatives and boundary conditions to obtain a unique solution. Let us approximate the function v in (4.26) with a polynomial of degree two, $N = 2$. The three GLL points will then be $\xi_0 = -1$, $\xi_1 = 0$ and $\xi_2 = 1$. The derivatives at these points are:

$$\begin{aligned} \hat{v}_\xi(-1) &= 2, \\ \hat{v}_\xi(0) &= 0, \\ \hat{v}_\xi(1) &= -2. \end{aligned}$$

If we only use the derivatives, the solution will not be unique. The choice $v(x) = c - x^2$, where c is a constant, will satisfy all the derivative requirements and the matrix $\mathbf{D} \in \mathbb{R}^{(N+1)\times(N+1)}$ will be singular.

In a second attempt of finding a unique solution we use the boundary conditions and the derivative at ξ_1:

$$\begin{aligned}\hat{v}(-1) &= 0,\\ \hat{v}_\xi(0) &= 0,\\ \hat{v}(1) &= 0.\end{aligned}$$

There is no unique solution to this problem either. For example $v(x) = 1 - x^2$, $v(x) = -1 + x^2$ and $v(x) = 1-x^4$ all satisfy the above conditions. If we instead use the boundary condition at one point and the derivatives at the reminding points we get a third set of equations we get a third set of equations:

$$\begin{aligned}\hat{v}(-1) &= 0,\\ \hat{v}_\xi(0) &= 0,\\ \hat{v}_\xi(1) &= -2.\end{aligned}$$

The solution to this problem is unique. Hence, to find the unique nodal values we use the boundary condition at one of the boundaries and the derivative at the remaining points. We can now write equation (4.28) as

$$\begin{aligned}\hat{v}_\xi(\xi_i) &= v_0 D_{i0} + \sum_{m=1}^{N} v_m D_{im}\\ \hat{v}_\xi(\xi_i) - v_0 D_{i0} &= \sum_{m=1}^{N} D_{im} v_m,\end{aligned}$$

for $i = 1, .., N$. In matrix form we write

$$\mathbf{v}_{,\xi} - v_0 \mathbf{d}_0 = \mathbf{D}_1 \mathbf{v},$$

where $\mathbf{d}_0 = \Big[D_{10}, \dots, D_{N0}\Big]^T$ and $\mathbf{D}_1$ has dimension $N \times N$. Again, the subscript 1 denotes the dimension in the x-direction. We loose one degree of freedom to one of the boundary conditions. Notice that $\mathbf{v}_{,\xi}$ and $\mathbf{v}$ have changed; $\mathbf{v}_{,\xi}, \mathbf{v} \in \mathbb{R}^N$. The matrix and the vectors on the left hand side are known and we find $\mathbf{v}$ by computing

$$\mathbf{v} = \mathbf{D_1}^{-1}(\mathbf{v}_{,\xi} - u_0 \mathbf{d}_0), \tag{4.29}$$

When we have found the nodal values we can find the function values at any point in the domain using (4.27).

This result can be extended to two dimensions, where the discrete function is

$$\hat{u}_N(\xi,\eta) = \sum_{m=0}^{N}\sum_{n=0}^{N} u_{mn}\,\ell_m(\xi)\,\ell_n(\eta) \tag{4.30}$$

and the derivatives at the GLL grid points are

$$\begin{aligned} \left.\frac{\partial \hat{u}_N}{\partial \xi}\right|_{(\xi_i,\xi_j)} &= \sum_{m=0}^{N}\sum_{n=0}^{N} u_{mn}\,\underbrace{\ell_m'(\xi_i)}_{D_{im}}\,\underbrace{\ell_n(\xi_j)}_{\delta_{nj}} \\ &= D_{i0}u_{0j} + \sum_{m=1}^{N} D_{im}u_{mj}. \end{aligned} \tag{4.31}$$

If we define the vector $\mathbf{u}^0 = \left[u_{01},\ldots,u_{0N}\right]$ and exploit local data representation, the above equation can be written as

$$\mathbf{U}_{,\xi} - \mathbf{d}_0\mathbf{u}^0 = \mathbf{D}_1\mathbf{U}. \tag{4.32}$$

Note that $\mathbf{U}_{,\xi},\mathbf{U},\mathbf{D}_1 \in \mathbb{R}^{N\times N}$, $\mathbf{d}_0 \in \mathbb{R}^{N\times 1}$ and $\mathbf{u}^0 \in \mathbb{R}^{1\times N}$. The notation $\mathbf{u}^0$ and $\mathbf{d}_0$ denote the first row and column vector of $\mathbf{U}$ and $\mathbf{D}$ respectively. Finally, step four can be written as:

$$\boxed{\mathbf{U} = \mathbf{D}_1^{-1}\left(\mathbf{U}_{,\xi} - \mathbf{d}_0\mathbf{u}^0\right).} \tag{4.33}$$

Using a similar procedure for finding $\mathbf{U}$ when the partial derivative with respect to η is known, gives the following result

$$\mathbf{U}_{,\eta} - \mathbf{u}_0\mathbf{d}^0 = \mathbf{U}\mathbf{D}_1^T,$$

where $\mathbf{u}_0 = \left[u_{10},\ldots,u_{N0}\right]^T$ and $\mathbf{d}^0 = \left[D_{01},\ldots,D_{0N}\right]$. We then get that

$$\boxed{\mathbf{U} = \left(\mathbf{U}_{,\eta} - \mathbf{u}_0\mathbf{d}^0\right)\mathbf{D}_1^{-T}.} \tag{4.34}$$

If the matrix $\mathbf{D}_1^{-1}$ is known, we can compute $\mathbf{U}$ in $\mathcal{O}(N^3)$ operations. If the function we approximate is infinitely smooth, we expect exponential convergence as N increases using either (4.33) or (4.34) ($\|u(x) - u_N(x)\| \sim \mathcal{O}(e^{-\mu N})$). If the function is a polynomial of degree P we expect an exact solution to the problem when $N \geq P$.

Numerical results in $\mathbb{R}$

The first problem we consider in $\mathbb{R}$ is

$$\begin{aligned}\frac{\mathrm{d}u}{\mathrm{d}x} =& 15x^4 + 8x^3 \quad \text{in } \Omega = (-1,1),\\ u(-1) =& 2,\\ u(1) =& 8.\end{aligned} \tag{4.35}$$

The exact solution is a polynomial of degree 5, $u \in \mathbb{P}_5(\Omega)$. We know the boundary conditions at both sides. However, we only impose the boundary condition at one side to obtain a unique solution. On the contrary, when solving a hyperbolic problem, such as $-u_{xx} = f$, two boundary conditions must be imposed to obtain a unique solution. Figure 4.5 shows the exact solution to (4.35) and the numerical solutions when $N = 3$ and $N = 4$. In the figure to the left, the boundary condition at $x = -1$ is imposed, while the boundary condition at $x = 1$ is imposed in the figure to the right. Both methods gives the exact solution (to machine precision) when $N \geq 5$. Figure 4.6 shows the error with respect to the discrete L^2 and energy norm.

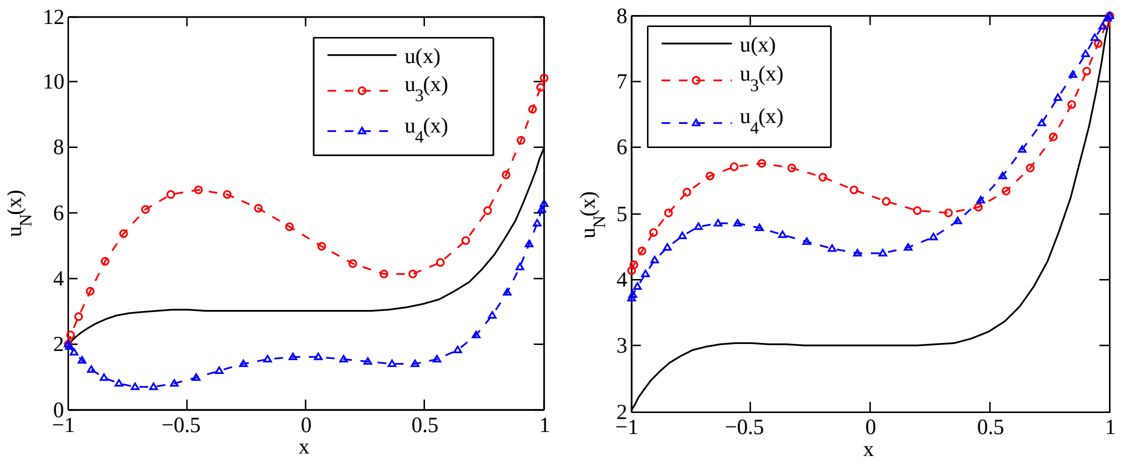


Figure 4.5: The discrete solution $u_N(x) \in \mathbb{P}_N(\Omega)$ and the exact solution of the problem $\frac{\mathrm{d}u}{\mathrm{d}x} = 15x^4 + 8x^3$ in the domain $\Omega = (-1,1)$. In the figure to the left we have only used the boundary condition at $x = -1$ and the derivatives at the remaining GLL points to compute $u_N(x)$. In the figure to the right we have imposed the boundary condition at $x = 1$. The numerical function values at the GLL points are marked with circles and triangles.

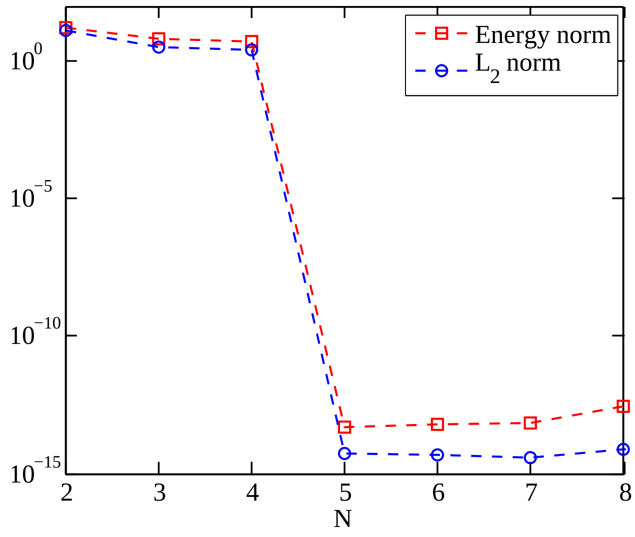


Figure 4.6: The discretization error $u(x) - u_N(x)$ measured in the discrete L^2 norm $\| \cdot \|_{L^2_N(\Omega)}$, and the discrete energy norm $\| \cdot \|_N$, as a function of the polynomial degree N. $u_N \in \mathbb{P}_N(\Omega)$ is the discrete solution to the problem $\frac{du}{dx} = 15x^4 + 8x^3$ in the domain $\Omega = (-1, 1)$ with nonhomogeneous Dirichlet boundary conditions.

Next, we consider a problem where the solution is infinitely smooth, but the solution is not a polynomial,

$$\begin{aligned} \frac{\mathrm{d}u}{\mathrm{d}x} =& 2\pi\cos(2\pi x) \quad \text{in } \Omega = (-1, 1), \\ u(-1) =& 0, \\ u(1) =& 0. \end{aligned}$$

We expect exponential convergence as N increases

$$\|u - u_N\| \propto e^{-\mu N}$$

or

$$\log(\|u - u_N\|) \propto -\mu N.$$

If we plot the error on a logarithmic scale, we expect to see a linear function with a slope μ. Accordingly, Figure 4.8 shows exponential convergence of the discrete error. Both the discrete energy norm and L^2 norm of the error are plotted on a logarithmic scale as a function of N. We also notice that the error reduces quicker when N is chosen to be odd. In fact, the error grows when N increases from odd to even. The exact solution and the discrete solutions $u_N(x)$ for $N = 7, 8, 9$ are plotted in Figure 4.7. We clearly see that $u_7(x)$ is a better approximation than $u_8(x)$.

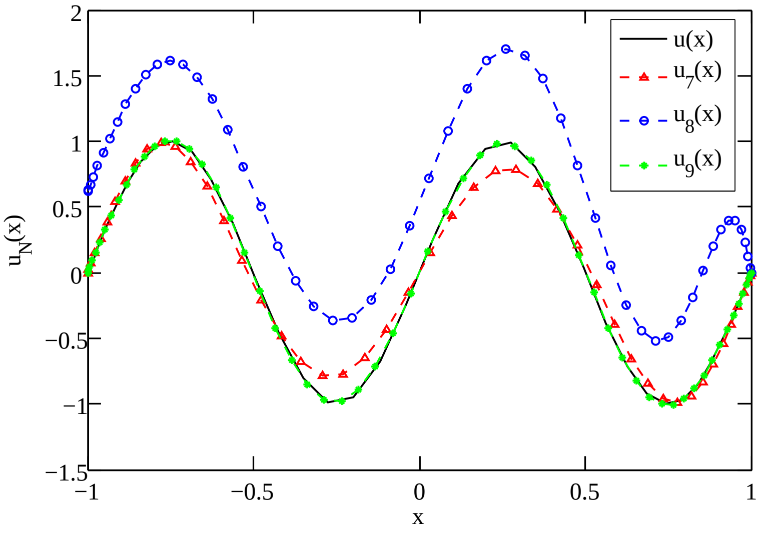


Figure 4.7: The discrete solutions $u_N(x) \in \mathbb{P}_N(\Omega)$ for $N = 7, 8, 9$ and the exact solution of the problem $\frac{\mathrm{d}u}{\mathrm{d}x} = 2\pi\cos(2\pi x)$. The domain is $\Omega = (-1, 1)$ and u is subject to homogeneous Dirichlet boundary conditions. We only impose the boundary condition at $x = 1$. The numerical function values at the GLL points are marked.

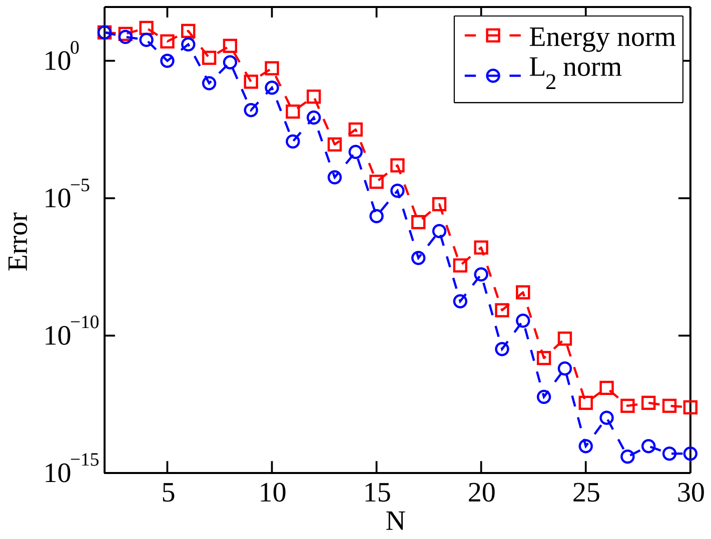


Figure 4.8: The discretization error $u(x) - u_N(x)$ measured in the discrete L^2 norm $\|\cdot\|_{L^2_N(\Omega)}$, and the discrete energy norm $|\!|\!|\cdot|\!|\!|_N$, as a function of the polynomial degree N. $u_N \in \mathbb{P}_N(\Omega)$ is the discrete solution to the problem $\frac{\mathrm{d}u}{\mathrm{d}x} = 2\pi\cos(2\pi x)$ in the domain $\Omega = (-1, 1)$ with homogeneous Dirichlet boundary conditions.

Numerical results in $\mathbb{R}^2$

Let us now consider the following problems in $\mathbb{R}^2$:

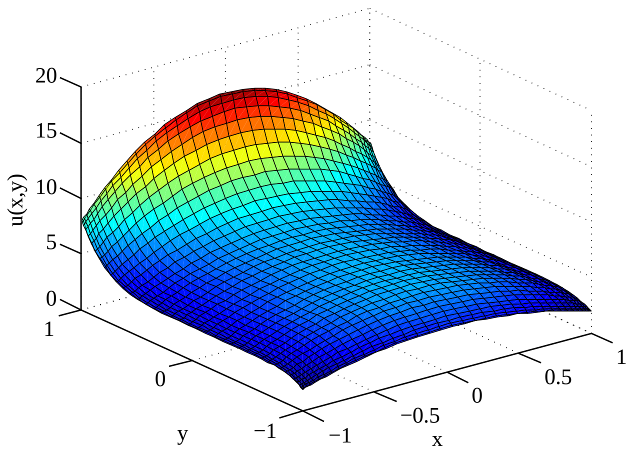
$$\begin{aligned}
\frac{\partial u}{\partial x} &= -2x^2(3y^5 + 2y^4 + 3) \quad \text{in } \Omega = (-1,1) \times (-1,1), \\
u(-1, y) &= 3y^5 + 2y^4 + 3, \\
u(x, -1) &= 2(2 - x^2)
\end{aligned} \tag{4.36}$$

and

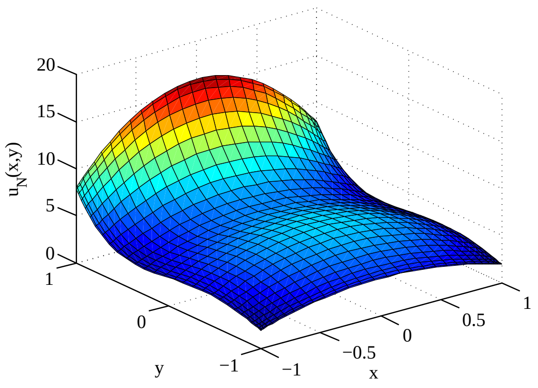
$$\begin{aligned}
\frac{\partial u}{\partial y} &= (2 - x^2)(15y^4 + 8y^3) \quad \text{in } \Omega = (-1,1) \times (-1,1), \\
u(-1, y) &= 3y^5 + 2y^4 + 3, \\
u(x, -1) &= 2(2 - x^2).
\end{aligned} \tag{4.37}$$

Both of these problems have the exact same solution, which is a polynomial of degree 5, $u(x, y) \in \mathbb{P}_5(\Omega)$. In (4.36) we are given the partial derivative with respect to x, and we use (4.33) to solve the problem numerically. In (4.37) we know the partial derivative with respect to y, and we then use (4.34) to solve the problem numerically.

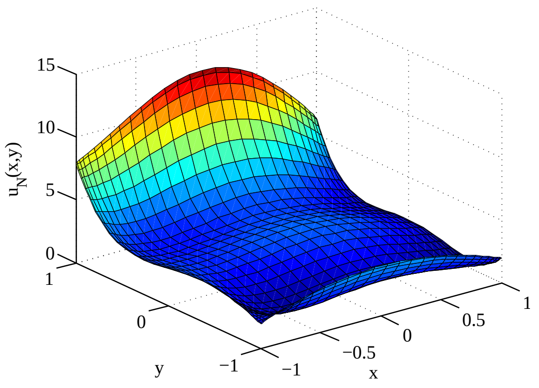
In Figure 4.9 the exact solution and both numerical solutions $u_N(x, y) \in \mathbb{P}_4(\Omega)$ are plotted in the domain Ω. We clearly see that there is a difference in the two numerical solutions. However, both methods give the exact solution (to machine precision) when $N = 5$, witness Figure 4.10.



(a) The exact solution $u(x,y) = (2 - x^2)(3y^5 + 2y^4 + 3) \in \mathbb{P}_5(\Omega)$ of problem (4.36) and (4.37).

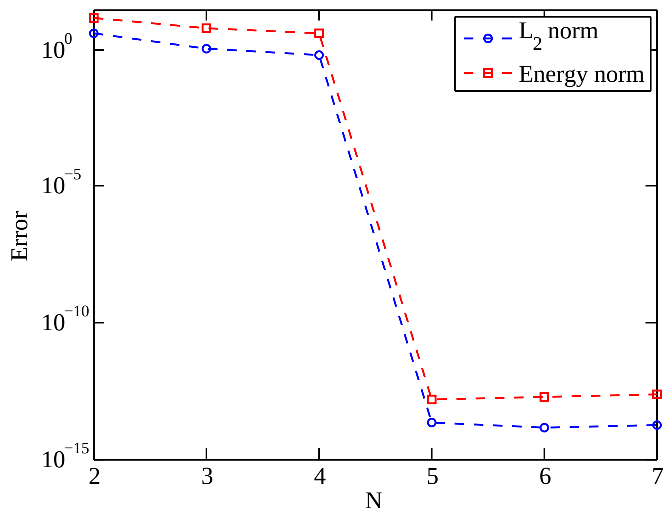


(b) The discrete solution $u_N(x,y) \in \mathbb{P}_4(\Omega)$ of problem (4.36), where the partial derivative with respect to x is given.

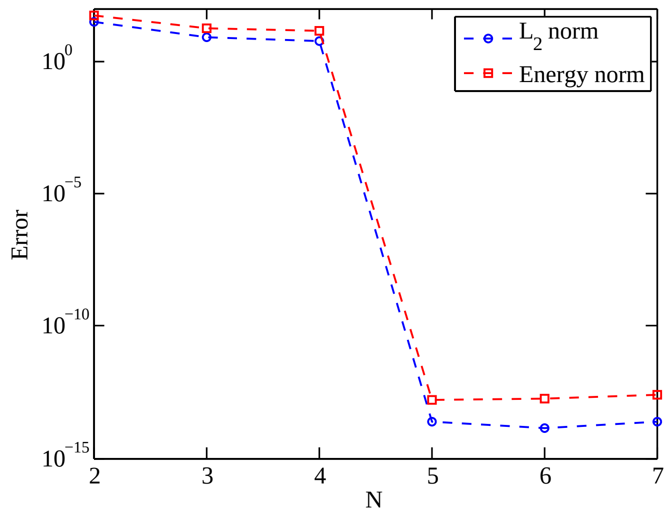


(c) The discrete solution $u_N(x,y) \in \mathbb{P}_4(\Omega)$ of problem (4.37), where the partial derivative with respect to y is given.

Figure 4.9: The exact solution (a) and the numerical solutions (b) and (c) to problems (4.36) and (4.37).



(a) The discretization error of problem (4.36) where the partial derivative with respect to x is given with nonhomogeneous boundary conditions.



(b) The discretization error of problem (4.37) where the partial derivative with respect to y is given with nonhomogeneous boundary conditions.

Figure 4.10: The discretization error $u(x, y) - u_N(x, y)$ measured in the discrete L^2 norm $\| \cdot \|_{L^2_N(\Omega)}$, and the discrete energy norm $\| \cdot \|_N$, as a function of the polynomial degree N, $u_N \in \mathbb{P}_N(\Omega)$ and $\Omega = (-1, 1) \times (-1, 1)$.

4.6.2 Steps 1 and 2 ofAlgorithm 5

We now consider the first two steps of **Algorithm 5**, where we find the solution v^* to the generalized discrete problem, and its derivatives $\mathbf{U}^*_{,\xi}$ and $\mathbf{U}^*_{,\eta}$. Recall that the derived system of equations are written in a global form as

$$(\widehat{\mathbf{A}} \otimes \widehat{\mathbf{B}} + \widehat{\mathbf{B}} \otimes \widehat{\mathbf{A}})\mathbf{u}^* = \mathbf{b}^*.$$

This system is similar to the global form (3.21) in the rectangular case if $L_x = L_y = 2$. Assume we know b^*. We can then use **Algorithm 1** to compute $\mathbf{U}^*$; the fast algorithm for the Poisson problem with Dirichlet boundary conditions. The operational cost is $\mathcal{O}(N^3)$ for $\mathcal{O}(N^2)$ unknowns. When we know $\mathbf{U}^*$ we can use the following relation to find $\mathbf{U}^*_{,\xi}$

$$\begin{aligned}\left.\frac{\partial u_N^*}{\partial \xi}\right|_{(\xi_i,\xi_j)} &= \sum_{m=0}^{N}\sum_{n=0}^{N} u^*_{mn}\underbrace{\ell'_m(\xi_i)}_{D_{im}}\underbrace{\ell_n(\xi_j)}_{\delta_{nj}} \\ &= \sum_{m=0}^{N} D_{im}u^*_{mj}.\end{aligned}$$

We compute $\mathbf{U}^*_{,\xi}$ with a simple matrix-matrix product

$$\boxed{\mathbf{U}^*_{,\xi} = \mathbf{D}\mathbf{U}^*.}$$

Similarly we find the partial derivatives with respect to η by

$$\boxed{\mathbf{U}^*_{,\eta} = \mathbf{U}^*\mathbf{D}^T.}$$

Recall the notation

$$(U_{,\xi})_{ij} = \left.\frac{\partial u_N}{\partial \xi}\right|_{(\xi_i,\xi_j)} \quad \text{and} \quad (U_{,\eta})_{ij} = \left.\frac{\partial u_N}{\partial \eta}\right|_{(\xi_i,\xi_j)}.$$

4.6.3 Step 3 of Algorithm 5

Step 3 is the only step we have not yet discussed. Consider the following relation, which we will use to find the partial derivatives of $\hat{u}_N$ at the GLL grid points:

$$\widehat{\nabla}\hat{u}_N^* = \mathbf{G}^*_{\nabla}\widehat{\nabla}\hat{u}_N.$$

Taking the inverse of the matrix $\mathbf{G}_\nabla^*$ we can find the partial derivates of u_N as

$$\widehat{\nabla}\hat{u}_N = \mathbf{G}_\nabla^{*\,-1}\widehat{\nabla}\hat{u}_N^*. \tag{4.38}$$

We know that $\mathbf{G}_\nabla^* = \sqrt{J}\mathbf{G}_\nabla$, where the elements of $\mathbf{G}_\nabla$ are defined in (4.7) and the Jacobian is defined in (4.1). We thus find an expression for this matrix

$$\mathbf{G}_\nabla^* = \frac{1}{\sqrt{J}}\begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix}, \tag{4.39}$$

and the inverse becomes

$$\mathbf{G}_\nabla^{*\,-1} = \sqrt{J}\begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix}^{-1} = \frac{\sqrt{J}}{\underbrace{y_\eta x_\xi - y_\xi x_\eta}_{J}}\begin{pmatrix} x_\xi & y_\xi \\ x_\eta & y_\eta \end{pmatrix} = \frac{1}{\sqrt{J}}\begin{pmatrix} x_\xi & y_\xi \\ x_\eta & y_\eta \end{pmatrix}.$$

Finally (4.38) can be written as

$$\begin{pmatrix} u_{N\xi} \\ u_{N\eta} \end{pmatrix} = \frac{1}{\sqrt{J}}\begin{pmatrix} x_\xi & y_\xi \\ x_\eta & y_\eta \end{pmatrix}\begin{pmatrix} u^*_{N\xi} \\ u^*_{N\eta} \end{pmatrix}.$$

We can now compute the partial derivatives of $u_N(x,y)$ at the $(N+1)^2$ GLL grid points from

$$\boxed{(U_{,\xi})_{ij} = \frac{1}{\sqrt{J_{ij}}}\Big((X_{,\xi})_{ij}(U^*_{,\xi})_{ij} + (Y_{,\xi})_{ij}(U^*_{,\eta})_{ij}\Big)}$$

and

$$\boxed{(U_{,\eta})_{ij} = \frac{1}{\sqrt{J_{ij}}}\Big((X_{,\eta})_{ij}(U^*_{,\xi})_{ij} + (Y_{,\eta})_{ij}(U^*_{,\eta})_{ij}\Big)}$$

for $i,j = 1,...,N-1$. We can compute $\mathbf{X}_{,\xi}$, $\mathbf{X}_{,\eta}$, $\mathbf{Y}_\xi$ and $\mathbf{Y}_\eta$ from (4.10), (4.11), (4.12) and (4.13) in $\mathcal{O}(N^3)$ operations and we can find J_{ij} from these matrices in $\mathcal{O}(N^2)$ operations. We already know $\mathbf{U}^*_{,\xi}$ and $\mathbf{U}^*_{,\eta}$ from step two. The total cost for step 3 is $\mathcal{O}(N^3)$ floating point operations.

We have now gone through all the four steps to compute $u_N(x,y)$, but after further considerations it is clear that the gradient of u_N^* in the mapping (4.7) is not always a real gradient. A gradient to any scalar field ϕ, $\nabla\phi$, has to fulfill the property [15]

$$\nabla \times (\nabla\phi) = \mathbf{0} \tag{4.40}$$

In words, the curl of the gradient of any scalar field is the zero vector. In the two dimensional case this boils down to the equation

$$\frac{\partial \phi_y}{\partial x} - \frac{\partial \phi_x}{\partial y} = 0.$$

We consider a simple example where the function mapped from the reference domain is

$$\phi(\xi, \eta) = \sin(\pi\xi)\sin(\pi\eta) \quad \text{in } \widehat{\Omega} = (-1,1) \times (-1,1),$$

while the physical domain is the rectangle $\Omega = (0, L_x) \times (0, L_y)$. The gradient with respect to the reference variables becomes

$$\widehat{\nabla}\phi = \begin{pmatrix} \phi_\xi \\ \phi_\eta \end{pmatrix} = \pi \begin{pmatrix} \cos(\pi\xi)\sin(\pi\eta) \\ \sin(\pi\xi)\cos(\pi\eta) \end{pmatrix}.$$

If we then use the information from (3.1), the matrix (4.39) becomes

$$\mathbf{G}^*_\nabla = \frac{1}{\sqrt{\frac{L_x L_y}{4}}} \begin{pmatrix} \frac{L_y}{2} & 0 \\ 0 & \frac{L_x}{2} \end{pmatrix}$$

and the gradient of ϕ^* is given by

$$\widehat{\nabla}\phi^* = \mathbf{G}^*_\nabla \widehat{\nabla}\phi = \frac{\pi}{\sqrt{L_x L_y}} \begin{pmatrix} L_y \cos(\pi\xi)\sin(\pi\eta) \\ L_x \sin(\pi\xi)\cos(\pi\eta) \end{pmatrix}.$$

The gradient to any scalar field ϕ has to satisfy (4.40), meaning we must have

$$\begin{aligned} \widehat{\nabla} \times (\widehat{\nabla}\phi^*) =& \frac{\pi}{\sqrt{L_x L_y}} \left(\pi L_x \cos(\pi\xi)\cos(\pi\eta) - L_y \cos(\pi\xi)\cos(\pi\eta)\right) \\ =& \frac{\pi^2}{\sqrt{L_x L_y}} \cos(\pi\xi)\cos(\pi\eta)\left(L_x - L_y\right) = 0. \end{aligned}$$

This is only true if $L_x = L_y$. The gradient to ϕ^* will therefore not always satisfy (4.40). Hence, the fast tensor-product algorithm for deformed domains given by **Algorithm 5** can not be realized.

Chapter 5

Tensor product solvers for time dependent problems

5.1 The unsteady convection-diffusion equation

The one-dimensional unsteady convection-diffusion equation with homogeneous boundary conditions is defined as

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \kappa \frac{\partial^2 u}{\partial x^2} + f(x,t) \quad \text{in} \quad \omega_x = (0, L), \tag{5.1a}$$

$$u(x = 0, t) = 0, \quad \forall\, t \in \omega_t = [0, T], \tag{5.1b}$$

$$u(x = L, t) = 0, \quad \forall\, t \in \omega_t, \tag{5.1c}$$

with initial condition

$$u(x, t = 0) = s(x). \tag{5.1d}$$

$u(x,t)$ is the temperature (the solution), κ is the thermal diffusivity and $f(x,t)$ is a thermal heat source. In this report we assume that κ is a constant.

5.2 The weak formulation

To find the numerical solution we first convert the strong formulation (5.1) into a weak formulation [4]. We start by defining the spaces

$$X = H^1(\omega_t, L^2(\omega_x)) \cap L^2(\omega_t, H_0^1(\omega_x)) \tag{5.2}$$

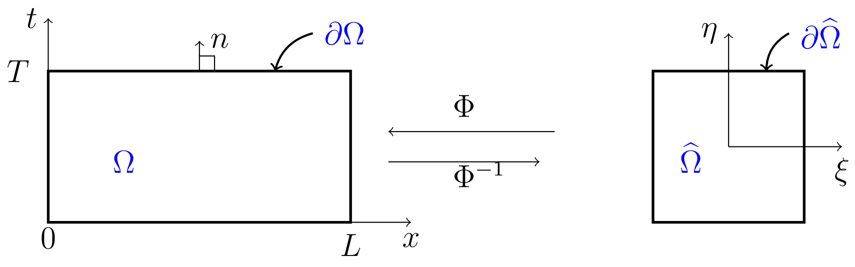


Figure 5.1: Illustration of the mapping between the reference domain $\widehat{\Omega} = (-1, 1) \times (-1, 1)$ and the rectangular physical domain $\Omega = (0, L) \times (0, T)$.

and

$$X^0 = \{w \in X, w(x, 0) = 0\}. \tag{5.3}$$

Let $v \in X^0$, and multiply both sides of (5.1a) with v and integrate over the time and space direction. If we then apply integration by parts on the diffusion term, the weak formulation can be stated as: find $u \in X$ such that

$$\int_0^T \int_0^L \frac{\partial u}{\partial t} v \,\mathrm{d}x\,\mathrm{d}t + \int_0^T \int_0^L \frac{\partial u}{\partial x} v \,\mathrm{d}x\,\mathrm{d}t = -\kappa \int_0^T \int_0^L \frac{\partial u}{\partial x}\frac{\partial v}{\partial x} \mathrm{d}x\,\mathrm{d}t + \int_0^T \int_0^L f\, v \,\mathrm{d}x\,\mathrm{d}t, \quad \forall\, v \in X^0.$$

In the same way we considered the two dimensional domain in space $\Omega = (0, L_x) \times (0, L_y)$ as an affine mapping of the reference domain, we will consider the space time domain

$$\Omega = (0, L) \times (0, T) = \omega_x \times \omega_t,$$

as an affine mapping of the reference domain,

$$\widehat{\Omega} = (-1, 1) \times (-1, 1).$$

We get the mapping

$$\Phi : \widehat{\Omega} \to \Omega,$$

which is illustrated in Figure 5.1. A function can then be expressed in terms of the reference variables,

$$\hat{u} = u \circ \Phi.$$

Recall the notation $\hat{u}$, which means that u is a function of ξ and η; the "*hat*" corresponds to the reference variables and the reference domain.

The weak formulation can be expressed in terms of the reference variables: Find $\hat{u} \in \widehat{X}$ such that

$$\int_{\hat{\Omega}} \frac{\partial \hat{u}}{\partial \eta} \hat{v}\, \mathrm{d}\xi\, \mathrm{d}\eta + \gamma_1 \int_{\hat{\Omega}} \frac{\partial \hat{u}}{\partial \xi} \hat{v}\, \mathrm{d}\xi\, \mathrm{d}\eta + \gamma_2 \int_{\hat{\Omega}} \frac{\partial \hat{u}}{\partial \xi} \frac{\partial \hat{v}}{\partial \xi} \mathrm{d}\xi\, \mathrm{d}\eta = \gamma_3 \int_{\hat{\Omega}} \hat{f}\, \hat{v}\, \mathrm{d}\xi\, \mathrm{d}\eta, \quad \forall\, \hat{v} \in \widehat{X}^0,$$

where

$$\widehat{X} = \left\{ \hat{w}, \quad \hat{w} \circ \Phi^{-1} \in X \right\}, \qquad \widehat{X}^0 = \left\{ \hat{w}, \quad \hat{w} \circ \Phi^{-1} \in X^0 \right\},$$

and

$$\gamma_1 = \frac{T}{L}, \qquad \gamma_2 = 2\kappa \frac{T}{L^2}, \qquad \gamma_3 = \frac{T}{2}.$$

The next step is to discretize the weak formulation. When spectral methods are applied on time-dependent PDE's, the numerical solution is usually approximated by high order polynomials in space. However, the time direction is usually discretized using a multi-stage or a multi-step method. If the solution is smooth, the error in the time direction will then be the dominant contribution to the discrete error, and exponential convergence will not be obtained when we let the time step go to zero.

To find the discrete solution we will now consider spectral discretization in time as well. With this new approach we are hoping to obtain exponential convergence for smooth functions.

5.3 Spectral discretization in space and time

To consider a spectral discretization based on high order polynomials in both space and time we introduce the discrete spaces

$$\widehat{X}_N = \widehat{X} \cap \mathbb{P}_N(\hat{\Omega}), \qquad \widehat{X}_N^0 = \widehat{X}^0 \cap \mathbb{P}_N(\hat{\Omega}).$$

The discrete problem is then stated as: Find $\hat{u}_N \in \widehat{X}_N$ such that

$$\int_{\hat{\Omega}} \frac{\partial \hat{u}_N}{\partial \eta} \hat{v}\, \mathrm{d}\xi\, \mathrm{d}\eta + \gamma_1 \int_{\hat{\Omega}} \frac{\partial \hat{u}_N}{\partial \xi} \hat{v}\, \mathrm{d}\xi\, \mathrm{d}\eta + \gamma_2 \int_{\hat{\Omega}} \frac{\partial \hat{u}_N}{\partial \xi} \frac{\partial \hat{v}}{\partial \xi} \mathrm{d}\xi\, \mathrm{d}\eta = \gamma_3 \int_{\hat{\Omega}} \hat{f}\, \hat{v}\, \mathrm{d}\xi\, \mathrm{d}\eta, \quad \forall \hat{v} \in \widehat{X}_N^0. \tag{5.4}$$

The numerical solution we are seeking is a polynomial of degree N, $\hat{u}_N \in \widehat{X}_N \subset \mathbb{P}_N(\widehat{\Omega})$. Let the tensor-product basis of the one dimensional Lagrange polynomials through the GLL points be the basis for the discrete spaces. $\hat{u}_N \in \widehat{X}_N$ can then be expressed as

$$\hat{u}_N(\xi, \eta) = \sum_{i=0}^{N} \sum_{j=0}^{N} u_{ij}\, \ell_i(\xi)\, \ell_j(\eta), \tag{5.5}$$

where u_{ij} are the nodal values at the $(N+1)^2$ GLL grid points, $u_{ij} = \hat{u}(\xi_i, \xi_j)$. The nodal values u_{0j} and u_{Nj} for $j = 0, .., N$ are given by the boundary conditions (5.1b) and (5.1c); $u_{0j} = u_{Nj} = 0$. The nodal values u_{i0} for $i = 0, ..., N$ are given by the initial condition (5.1d); $u_{i0} = s_m = \hat{s}(\xi_i)$. Hence, the initial condition is approximated by a high order polynomial.

Any function $\hat{v} \in \widehat{X}_N^0$ can be expressed as

$$\hat{v}(\xi, \eta) = \sum_{i=0}^{N} \sum_{j=0}^{N} v_{ij}\, \ell_i(\xi)\, \ell_j(\eta), \tag{5.6}$$

where we know that $u_{0k} = u_{Nk} = u_{k0} = 0$ for $k = 0, ..., N$. If we also approximate $f(\xi, \eta)$ as a high order polynomial and combine (5.4), (5.5) and (5.6), we end up with the following system of equations

$$\begin{aligned}
&\int_{-1}^{1}\int_{-1}^{1} \left(\sum_{m=0}^{N}\sum_{n=0}^{N} u_{mn}\, \ell_i(\xi)\, \ell_j'(\eta) \right) \ell_i(\xi)\, \ell_j(\eta)\, \mathrm{d}\xi\, \mathrm{d}\eta + \\
\gamma_1 &\int_{-1}^{1}\int_{-1}^{1} \left(\sum_{m=0}^{N}\sum_{n=0}^{N} u_{mn}\, \ell_m'(\xi)\, \ell_n(\eta) \right) \ell_i(\xi)\, \ell_j(\eta)\, \mathrm{d}\xi\, \mathrm{d}\eta + \\
\gamma_2 &\int_{-1}^{1}\int_{-1}^{1} \left(\sum_{m=0}^{N}\sum_{n=0}^{N} u_{mn}\, \ell_m'(\xi)\, \ell_n(\eta) \right) \ell_i'(\xi)\, \ell_j(\eta)\, \mathrm{d}\xi\, \mathrm{d}\eta = \\
&\gamma_3 \int_{-1}^{1}\int_{-1}^{1} \left(\sum_{m=0}^{N}\sum_{n=0}^{N} f_{mn}\, \ell_m(\xi)\, \ell_n(\eta) \right) \ell_i(\xi)\, \ell_j(\eta),
\end{aligned}$$

for $i = 1, ..., N-1, \quad j = 1, ..., N$. With the same notation as we used in previous chapters, $(v, w) = \int_{-1}^{1} v(\xi)\, w(\xi)\, \mathrm{d}\xi$, we can rewrite the above equation as

$$\begin{aligned}
&\sum_{m=0}^{N}\sum_{n=0}^{N} (\ell_i, \ell_m)(\ell_j, \ell_n')u_{mn} + \gamma_1 \sum_{m=0}^{N}\sum_{n=0}^{N} (\ell_i, \ell_m')(\ell_j, \ell_n)u_{mn} + \\
\gamma_2 &\sum_{m=0}^{N}\sum_{n=0}^{N} (\ell_i', \ell_m')(\ell_j, \ell_n)u_{mn} = \gamma_3 \sum_{m=0}^{N}\sum_{n=0}^{N} (\ell_i, \ell_m)(\ell_j, \ell_n)f_{mn}
\end{aligned}$$

for $i = 1, ..., N-1, \quad j = 1, ..., N$. The integrals can now be evaluated numerically by GLL quadrature. Two of these integrals are already evaluated in chapter 3:

$$\begin{aligned}
\widehat{A}_{ij} &= (\ell_i', \ell_j')_N = \sum_{i=0}^{N} \rho_i D_{\alpha i} D_{\alpha j}, \\
\widehat{B}_{ij} &= (\ell_i, \ell_j)_N = \rho_i \delta_{ij},
\end{aligned}$$

where $D_{ij} \equiv \ell_j(\xi_i)$. $\widehat{A}_{ij}$ are the matrix elements of the one-dimensional stiffness matrix $\widehat{\mathbf{A}}$ and $\widehat{B}_{ij}$ are the matrix elements of the one-dimensional mass matrix $\widehat{\mathbf{B}}$. The last integral we want to evaluate is

$$\widehat{C}_{ij} = (\ell_i, \ell_j')_N = \rho_i D_{ij}, \tag{5.7}$$

where $\widehat{C}_{ij}$ are the matrix elements of the one-dimensional convection matrix $\widehat{\mathbf{C}}$. Both $\widehat{\mathbf{C}}$ and $\widehat{\mathbf{A}}$ are exactly evaluated by the GLL-quadrature. With these three matrices we end up with the system of equations

$$\begin{aligned}&\sum_{m=1}^{N-1}\sum_{n=1}^{N}\left(\widehat{B}_{im}\widehat{C}_{jn} + \gamma_1\widehat{C}_{im}\widehat{B}_{jn} + \gamma_2\widehat{A}_{im}\widehat{B}_{jn}\right)u_{mn} = \\ &\gamma_3\sum_{m=1}^{N-1}\sum_{n=1}^{N}\widehat{B}_{im}\widehat{B}_{jn}f_{mn} - \sum_{m=1}^{N-1}\delta_{im}\widehat{C}_{j0}s_m, \quad i = 1, ..., N-1, \quad j = 1, ..., N,\end{aligned} \tag{5.8}$$

where $s_m = \hat{s}(\xi_m)$ and $f_{mn} = \hat{f}(\xi_m, \xi_n)$. In the next section we exploit tensor-product properties in a clever way to find a fast solver for (5.8) .

5.4 Tensor product solver

To find a fast tensor-product solver for (5.8), we start by introducing the variables

$$g_{ij} = \gamma_3\sum_{m=1}^{N-1}\sum_{n=1}^{N}\widehat{B}_{im}\widehat{B}_{jn}f_{mn} - \sum_{m=1}^{N-1}\delta_{im}\widehat{C}_{j0}s_m. \tag{5.9}$$

We can then rewrite (5.8) as

$$\sum_{m=1}^{N-1}\sum_{n=1}^{N-1}\left(\widehat{B}_{im}\widehat{C}_{jn} + \gamma_1\widehat{C}_{im}\widehat{B}_{jn} + \gamma_2\widehat{A}_{im}\widehat{B}_{jn}\right)u_{mn} = g_{ij}. \tag{5.10}$$

Exploiting the tensor-product properties, we write the above equation in global form,

$$\left(\widehat{\mathbf{C}}_2 \otimes \widehat{\mathbf{B}}_1 + \gamma_1\widehat{\mathbf{B}}_2 \otimes \widehat{\mathbf{C}}_1 + \gamma_2\widehat{\mathbf{B}}_2 \otimes \widehat{\mathbf{A}}_1\right)\mathbf{u}^x = \mathbf{g}^x. \tag{5.11}$$

The subscript 1 indicates that the matrix corresponds to the space direction ξ, while the subscript 2 indicate that the matrix corresponds to the time direction η. The dimension of $\widehat{\mathbf{C}}_1$ is $(N-1) \times (N-1)$, while the dimension of $\widehat{\mathbf{C}}_2$ is $N \times N$.

The first step towards finding a fast tensor-product solver is to diagonalize the convection matrix $\widehat{\mathbf{C}}_2$ with respect to the mass matrix $\widehat{\mathbf{B}}_2$. Let us consider the eigenvalue problem

$$\widehat{\mathbf{C}}_2\mathbf{S}_2 = \widehat{\mathbf{B}}_2\mathbf{S}_2\mathbf{\Lambda}_2,$$

where $\mathbf{\Lambda}$ is the diagonal eigenvalue matrix with elements $\Lambda_{ii} = \lambda_i$ and $\mathbf{S}_2$ is the eigenvector matrix with the eigenvectors in each column. As opposed to $\widehat{\mathbf{C}}_1$, $\widehat{\mathbf{C}}_2$ is not skew-symmetric because of the boundary condition imposed at $\eta = -1$. $\widehat{\mathbf{C}}_2$ is therefore not normal and we are not able to diagonalize $\widehat{\mathbf{C}}_2$ for every N. However, if $N \leq 32$ we expect that we can diagonalize $\widehat{\mathbf{C}}$ without any significant roundoff errors [1]. We are not going to consider higher values of N in the time direction. The eigenvalues of $\widehat{\mathbf{C}}$ will be complex, which is important to take into account when we implement the method. $\widehat{\mathbf{C}}_2$ and $\widehat{\mathbf{B}}_2$ in (5.11) can now be written as

$$\begin{aligned} \widehat{\mathbf{C}}_2 &= \widehat{\mathbf{B}}_2 \mathbf{S}_2 \mathbf{\Lambda}_2 \mathbf{S}_2^{-1}, \\ \widehat{\mathbf{B}}_2 &= \widehat{\mathbf{B}}_2 \mathbf{S}_2 \mathbf{S}_2^{-1}. \end{aligned} \tag{5.12}$$

We will discuss the generalized eigenvalue problem further in the next section. We now substitute these two expressions into (5.11) and end up with the following tensor-product form

$$\left(\widehat{\mathbf{B}}_2\mathbf{S}_2 \otimes \mathbf{I}_2\right)\left(\mathbf{\Lambda}_2 \otimes \widehat{\mathbf{B}}_1 + \mathbf{I}_2\left(\gamma_1\widehat{\mathbf{C}}_1 + \gamma_2\widehat{\mathbf{A}}_1\right)\right)\left(\mathbf{S}_2^{-1} \otimes \mathbf{I}_1\right)\mathbf{u}^x = \mathbf{g}^x. \tag{5.13}$$

Let us define the variables

$$\tilde{\mathbf{u}}^x = \left(\mathbf{S}_2^{-1} \otimes \mathbf{I}_1\right)\mathbf{u}^x, \tag{5.14}$$

$$\tilde{\mathbf{g}}^x = \left(\mathbf{S}_2^{-1}\widehat{\mathbf{B}}_2^{-1} \otimes \mathbf{I}_1\right)\mathbf{g}^x. \tag{5.15}$$

With these two variables, we can write (5.13) as

$$\left(\mathbf{\Lambda}_2 \otimes \widehat{\mathbf{B}}_1 + \mathbf{I}_2\left(\gamma_1\widehat{\mathbf{C}}_1 + \gamma_2\widehat{\mathbf{A}}_1\right)\right)\tilde{\mathbf{u}}^x = \tilde{\mathbf{g}}^x, \tag{5.16}$$

or

$$\sum_{m=1}^{N-1}\left(\left(\gamma_1\widehat{C}_1 + \gamma_2\widehat{A}_1\right)_{im} + \lambda_j(\widehat{B}_1)_{im}\right)\tilde{u}_{mj} = \tilde{g}_{ij} \tag{5.17}$$

for $i = 1, ..., N-1$, $j = 1, ..., N$. For each value of $j = 1, ..., N$ we define the vectors

$$\tilde{\mathbf{g}}_j = \begin{bmatrix} \tilde{g}_{1j} \\ \tilde{g}_{2j} \\ \vdots \\ \tilde{g}_{N-1j} \end{bmatrix}, \qquad \tilde{\mathbf{u}}_j = \begin{bmatrix} \tilde{u}_{1j} \\ \tilde{u}_{2j} \\ \vdots \\ \tilde{u}_{N-1j} \end{bmatrix}.$$

Exploiting local data structure, (5.17) can be expressed as N independent systems of equations

$$\left(\gamma_1\widehat{\mathbf{C}}_1 + \gamma_2\widehat{\mathbf{A}}_1 + \lambda_j\widehat{\mathbf{B}}_1\right)\tilde{\mathbf{u}}_j = \tilde{\mathbf{g}}_j, \quad \forall\, j. \tag{5.18}$$

Each system of equations has dimensions $(N-1)\times(N-1)$ and requires $\mathcal{O}(N^3)$ floating pint operations to be solved. Thus, using N processors, we can find $\tilde{\mathbf{u}}^x$, via Gaussian elimination, in $\mathcal{O}(N^3)$ operations for $\mathcal{O}(N^2)$ unknowns. Using (5.14) we can find the solution $\mathbf{u}^x$. Again, we exploit local data representation for efficiency, the fast algorithm taking advantage of the local data structure is stated

Algorithm 6 Fast convection-diffusion solver: spectral method

```
1. $\widetilde{\mathbf{G}} = \mathbf{G}\mathbf{B}_1^{-T}\mathbf{S}_1^{-T}$
2. $\tilde{\mathbf{u}}_j = \left(\gamma_1\widehat{\mathbf{C}}_1 + \gamma_2\widehat{\mathbf{A}}_1 + \lambda_j\widehat{\mathbf{B}}_1\right)^{-1}\tilde{\mathbf{g}}_j$   for all $j = 1, ..., N-1$
3. $\mathbf{U} = \widetilde{\mathbf{U}}\mathbf{S}_2^T$
```

in **Algorithm 6**. The total computational cost is $\mathcal{O}(N^3)$ floating point operations for $\mathcal{O}(N^2)$ unknowns, if we have N different processors to compute step two. The method is fully implicit.

In steps one and two of **Algorithm 6** we compute two complex matrices $\widetilde{\mathbf{G}}$ and $\widetilde{\mathbf{U}}$ while the matrix-matrix multiplication between the two complex matrices will give a real matrix in the last step. Because of computational roundoff errors, $\widetilde{\mathbf{U}}$ will have a small imaginary part. We overlook this, and merely consider the real part of $\widetilde{\mathbf{U}}$.

In section 5.6 we solve an unsteady convection-diffusion problem using **Algorithm 6** and verify the convergence rate of the method. First however, we study the generalized eigenvalue problem for the convection operator $\widehat{\mathbf{C}}_2$ by considering the convection problem.

5.5 The generalized eigenvalue problem for the convection operator

We now consider the linear convection problem

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0 \quad \text{in } \omega_x = (-1,1) \tag{5.19a}$$

$$u(-1,t) = 0 \quad \forall t \in \omega_t = [0,T]. \tag{5.19b}$$

To obtain the weak formulation we define the spaces

$$X = \left\{ v \in H^1(\omega_x) \,|\, v(-1) = 0 \right\},$$
$$Y(X) = \left\{ v \;\middle|\; \forall\, t^* \in \omega_t,\ v(x;t^*) \in X,\ \int_0^T \|v(x;t^*)\|^2_{H^1(\omega_x)}\,\mathrm{d}t < \infty \right\},$$

where t^* denotes a fixed value of t [10]. The weak formulation is stated as: Find $u \in Y(X)$ such that

$$\int_{-1}^{1} \frac{\partial u}{\partial t} v\,\mathrm{d}x + \int_{-1}^{1} \frac{\partial u}{\partial x} v\,\mathrm{d}x = 0, \quad \forall v \in X.$$

We now consider spectral discretization only in the space direction while the solution is continuous in time; so called semi-discretization. The discrete space is

$$X_N = X \cap \mathbb{P}_N(\omega_x).$$

We can now approximate the function u as

$$\hat{u}_N(\xi, t) = \sum_{i=1}^{N} u_i(t)\,\ell_i(\xi),$$

where $u_i(t) = \hat{u}(\xi_i, t)$ for $i = 1, ..., N$ and $\hat{u}_N(\xi, t) = u_N(x(\xi), t)$, where $x = \xi$. If we choose $\hat{v} = \ell_i(\xi)$ for $i = 1, ..., N$, the semi-discrete problem reads: Find $\hat{u}_N \in Y(X_N)$ such that

$$\int_{-1}^{1} \frac{\partial \hat{u}_N}{\partial t} \hat{v}\,\mathrm{d}\xi + \int_{-1}^{1} \frac{\partial \hat{u}_N}{\partial \xi} \hat{v}\,\mathrm{d}\xi = 0, \quad \forall \hat{v} \in X_N.$$

As before, we apply GLL quadrature and the system of equations can be expressed as

$$\widehat{\mathbf{B}}_2 \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \widehat{\mathbf{C}}_2\,\mathbf{u} = \mathbf{0}$$

or

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = -\widehat{\mathbf{B}}_2^{-1}\,\widehat{\mathbf{C}}_2\,\mathbf{u},$$

where

$$\left(\widehat{B}_2\right)_{ij} = \sum_{\alpha=0}^{N} \rho_\alpha \ell_i(\xi_\alpha)\ell_j(\xi_\alpha) = \rho_i \delta_{ij},$$
$$\left(\widehat{C}_2\right)_{ij} = \sum_{\alpha=0}^{N} \rho_\alpha \ell_i(\xi_\alpha)\ell_j'(\xi_\alpha) = \rho_i D_{ij},$$

and

$$\mathbf{u} = \begin{bmatrix} u_0 & u_1 & \cdots & u_N \end{bmatrix}^T.$$

Next, we want to study the stability of the above equation. Diagonalizing $-\widehat{\mathbf{B}}_2^{-1}\widehat{\mathbf{C}}_2$ will give us the model problem to study for stability

$$\frac{\mathrm{d}s}{\mathrm{d}t} = \lambda s,$$

where λ represents an eigenvalue of $-\widehat{\mathbf{B}}_2^{-1}\widehat{\mathbf{C}}_2$. The eigenvalue problem is exactly the same as we discussed for the fast solver in section 5.3. It can be shown that [1]

$$|\lambda| \leq \mathcal{O}(N^2). \tag{5.20}$$

Subject to the boundary condition (5.19b) the system has dimension $N \times N$. The real parts of the eigenvalues become strictly negative as N increases and the imaginary parts dominate the maximum absolute value of λ, see Figure 5.2. As expected, we can observe that the absolute value of λ satisfies (5.20) and that the maximum absolute value scales like $\mathcal{O}(N^2)$.

Roundoff errors can cause difficulties in solving the eigenvalue problem as N increases. One problem is that $\widehat{\mathbf{C}}_2$ will not be normal. We recall that a matrix $\mathbf{A}$ is normal if

$$\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T.$$

Examples of normal matrices are symmetric matrices $\mathbf{A} = \mathbf{A}^T$ and skew-symmetric matrices $\mathbf{A}^T = -\mathbf{A}$. Thus, $\widehat{\mathbf{C}}_1$ is normal. A normal matrix has a set of orthogonal eigenvectors and it is the absence of orthogonality that results in numerical difficulties. But the problems do not occur unless $N > 32$. In Figure 5.2 it is probably the roundoff errors that we observe when N goes from $N = 32$ to $N = 64$. However, in practice we will not consider a polynomial of that high a degree ($N > 32$).

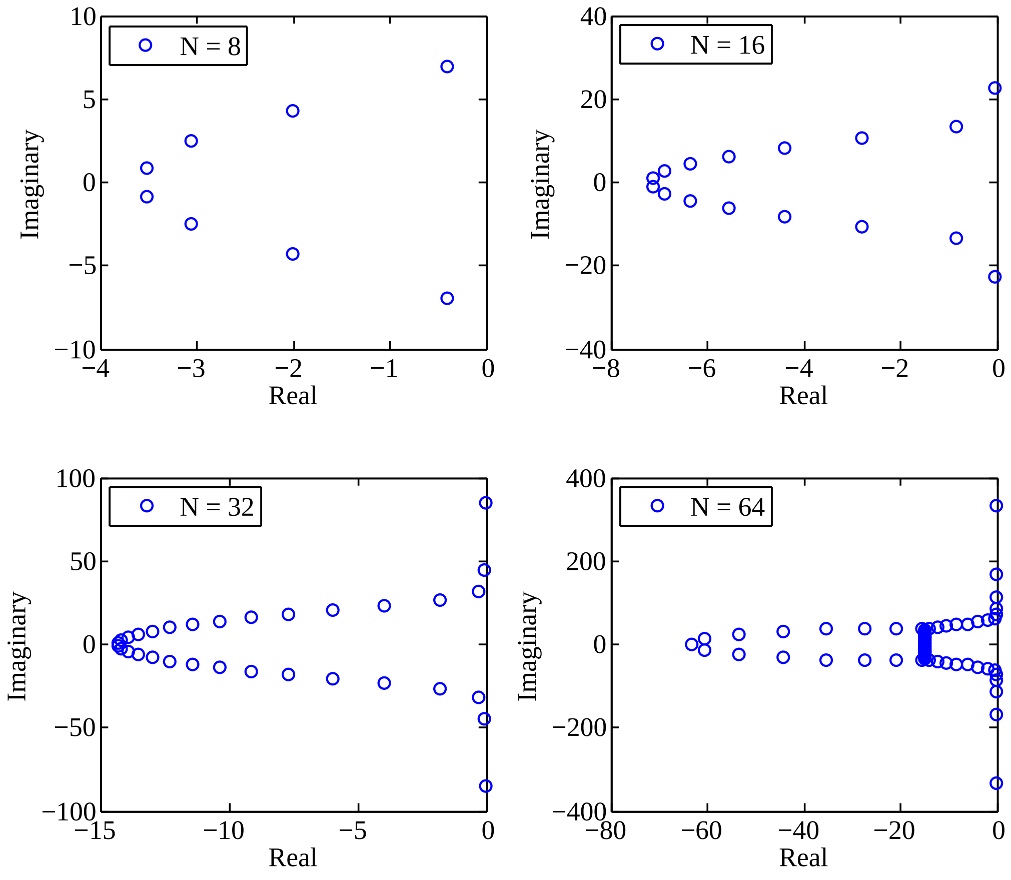


Figure 5.2: The eigenvalues of the generalized eigenvalue problem $\widehat{\mathbf{C}}_2\mathbf{S} = \widehat{\mathbf{B}}_2\mathbf{S}\Lambda$ for $N = 8, 16, 32, 64$. Roundoff errors occur when $N = 64$. Note the different scaling of the real and imaginary axes.

5.6 Numerical results

Let us now consider the unsteady one-dimensional convection-diffusion problem (5.1) with exact solution

$$u(x,t) = \sin(\pi x)\sin(\pi t) \quad \text{in } \Omega = \omega_x \times \omega_t = (0,1)\times(0,40), \tag{5.21}$$

subject to our model problem $L = 1$ and $T = 40$. To solve the problem we divide the domain into $K = 20$ spectral elements in time, such that $T = K\Delta T$, $\Delta T = 2$, and use a pure spectral method in space. The numerical solution can then be computed using the fast tensor-product solver given by **Algorithm 6** in each element. The initial condition in each element, except the first, is taken from the numerical solution in the previous element.

Figure 5.3 shows the numerical solution from the last spectral element, where $u_N(x,t) \in \mathbb{P}_{10}(\Omega_K)$ and $\Omega_K = ((0,1)\times(38,40))$.

The exact solution is infinitely smooth. We therefore expect exponential convergence. Figure 5.4 illuminates this. The error is measured in the discrete L_2 norm and energy norm, over the last spectral element $\Omega_K = (\omega_x \times (T-\Delta T, T))$.

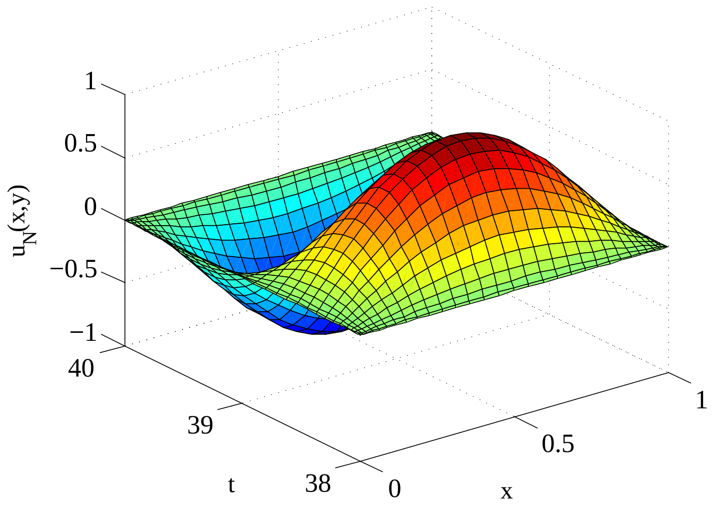


Figure 5.3: The discrete solution $u_N(x,t) \in \mathbb{P}_{10}(\Omega_K)$ computed at the last time step $t \in [38,40]$.

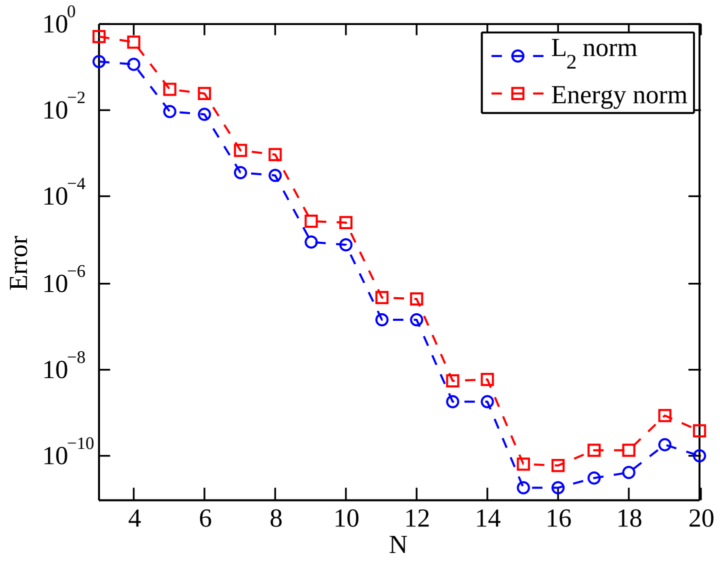


Figure 5.4: The discretization error $u - u_N$ of the one-dimensional unsteady convection-diffusion problem with exact solution $u(x,t) = \sin(\pi t)\sin(\pi x)$, where $u_N \in \mathbb{P}_N(\Omega_K)$. The error is measured in the discrete L_2 and energy norm in the domain of the last spectral element, $\Omega_K = (\omega_x \times (T - \Delta T, T))$, as a function of the polynomial degree N.

5.7 Nonlinear time-dependent PDE

We now consider a nonlinear time-dependent PDE, namely the unsteady convection-diffusion equation with a nonlinear term, subject to homogeneous Dirichlet boundary conditions:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \kappa \frac{\partial^2 u}{\partial x^2} + \alpha u^3 + f(x,t) \quad \text{in} \quad \omega_x = (0, L), \tag{5.22a}$$

$$u(x = 0, t) = 0, \quad \forall t \in \omega_t = [0, T], \tag{5.22b}$$

$$u(x = L, t) = 0, \quad \forall t \in \omega_t, \tag{5.22c}$$

with initial condition

$$u(x, t = 0) = s(x). \tag{5.22d}$$

$u(x,t)$ is the temperature (the solution), κ is the thermal diffusivity, α is a constant and $f(x,t)$ is a thermal heat source. We still assume κ is a constant.

Using a similar approach as we did for the unsteady convection-diffusion equation we end up with the weak formulation of the problem: Find $\hat{u} \in \widehat{X}$ such that

$$\int_{\widehat{\Omega}} \frac{\partial \hat{u}}{\partial \eta} \hat{v} \, \mathrm{d}\xi \, \mathrm{d}\eta + \gamma_1 \int_{\widehat{\Omega}} \frac{\partial \hat{u}}{\partial \xi} \hat{v} \, \mathrm{d}\xi \, \mathrm{d}\eta + \gamma_2 \int_{\widehat{\Omega}} \frac{\partial \hat{u}}{\partial \xi} \frac{\partial \hat{v}}{\partial \xi} \, \mathrm{d}\xi \, \mathrm{d}\eta = \beta \int_{\widehat{\Omega}} u^3 v \, \mathrm{d}\xi \, \mathrm{d}\eta + \gamma_3 \int_{\widehat{\Omega}} \hat{f} \hat{v} \, \mathrm{d}\xi \, \mathrm{d}\eta,$$

for all $\hat{v} \in \widehat{X}^0$, where

$$\widehat{X} = \left\{ \hat{w}, \quad \hat{w} \circ \Phi^{-1} \in X \right\}, \qquad \widehat{X}^0 = \left\{ \hat{w}, \quad \hat{w} \circ \Phi^{-1} \in X^0 \right\},$$

and

$$\gamma_1 = \frac{T}{L}, \qquad \gamma_2 = 2\kappa \frac{T}{L^2}, \qquad \gamma_3 = \frac{T}{2}, \qquad \beta = \frac{\alpha T}{2}.$$

The only term that is different from the unsteady convection-diffusion problem is the nonlinear term. We therefore focus on this term. For the unsteady convection-diffusion problem we approximated $\hat{u}(\xi, \eta)$ as

$$\hat{u}_N(\xi, \eta) = \sum_{i=0}^{N} \sum_{j=0}^{N} u_{ij}\, \ell_i(\xi)\, \ell_j(\eta).$$

If we discretize the nonlinear term in the same way, we get

$$\beta \int_{\widehat{\Omega}} \hat{v}\, \hat{u}_N^3 \,\mathrm{d}\xi\, \mathrm{d}\eta \;=\; \beta \int_{\widehat{\Omega}} \ell_i(\xi)\, \ell_j(\eta) \left(\sum_{m=0}^{N} \sum_{n=0}^{N} u_{mn}\, \ell_m(\xi)\, \ell_n(\eta) \right)^3 \mathrm{d}\xi\, \mathrm{d}\eta.$$

The right hand side of this equation will cause difficulties computationally. Let us instead define the nonlinear function

$$\hat{w}(\xi, \eta) \equiv \hat{u}^3(\xi, \eta).$$

We approximate this function as a polynomial of degree N, such that

$$\hat{w}_N(\xi, \eta) = \sum_{m=1}^{N-1} \sum_{n=1}^{N} w_{mn}\, \ell_m(\xi)\, \ell_n(\eta),$$

where $w_{mn} = u_{mn}^3$. The nonlinear term in the weak formulation then becomes

$$\begin{aligned} \beta \int_{\widehat{\Omega}} \hat{v}\, \hat{w}_N \,\mathrm{d}\xi\, \mathrm{d}\eta &= \beta \int_{\widehat{\Omega}} \ell_i(\xi)\, \ell_j(\eta) \sum_{m=1}^{N-1} \sum_{n=1}^{N} w_{mn}\, \ell_m(\xi)\, \ell_n(\eta) \\ &\approx \sum_{m=1}^{N-1} \sum_{n=1}^{N} \widehat{B}_{im}\, \widehat{B}_{jn}\, w_{mn}, \end{aligned}$$

where $\widehat{B}_{ij} = \rho_i \delta_{ij}$. The integrals are evaluated with GLL-quadrature. If the exact solution $\hat{u}(\xi, \eta)$ is a polynomial of degree N, then $\hat{w}(\xi, \eta)$ will be a polynomial of degree $3N$. The approximation of $\hat{w}$ will therefore not be exact unless we approximate the solution with a polynomial of degree $3N$. However, when we apply GLL-quadrature using N points we can never evaluate the integral exactly. GLL-quadrature only evaluates the integral exactly if the integrand is a polynomial of degree $K \leq 2N - 1$. If $N > 1$, this will never be the case for the nonlinear term where the integrand is a polynomial of degree $K = 4N$. On the other hand, the error will depend on the regularity of u. If u is very regular, say perhaps analytic, we expect the quadrature error of the nonlinear term to be small.

The nonlinear term can in global form be written as $\left(\widehat{\mathbf{B}}_2 \otimes \widehat{\mathbf{B}}_1\right)\mathbf{w}^x$. Adding this term to the global form of the discrete unsteady convection-diffusion problem (5.11) we end up with the following tensor-product form of the discrete problem

$$\left(\widehat{\mathbf{C}}_2 \otimes \widehat{\mathbf{B}}_1 + \gamma_1 \widehat{\mathbf{B}}_2 \otimes \widehat{\mathbf{C}}_1 + \gamma_2 \widehat{\mathbf{B}}_2 \otimes \widehat{\mathbf{A}}_1\right)\mathbf{u}^x - \left(\beta \widehat{\mathbf{B}}_2 \otimes \widehat{\mathbf{B}}_1\right)\mathbf{w}^x = \mathbf{g}^x. \tag{5.23}$$

As we did for the unsteady convection-diffusion equation, we diagonalize $\widehat{\mathbf{C}}_2$ and substitute (5.12) into the above equation,

$$\begin{aligned}\left(\widehat{\mathbf{B}}_2\mathbf{S}_2 \otimes \mathbf{I}_2\right)\left(\mathbf{\Lambda}_2 \otimes \widehat{\mathbf{B}}_1 + \mathbf{I}_2\left(\gamma_1\widehat{\mathbf{C}}_1 + \gamma_2\widehat{\mathbf{A}}_1\right)\right)\left(\mathbf{S}_2^{-1} \otimes \mathbf{I}_1\right)\mathbf{u}^x \\ - \left(\widehat{\mathbf{B}}_2\mathbf{S}_2 \otimes \mathbf{I}_1\right)\left(\beta \mathbf{I}_2 \otimes \widehat{\mathbf{B}}_1\right)\left(\mathbf{S}_2^{-1} \otimes \mathbf{I}_1\right)\mathbf{w}^x = \mathbf{g}^x.\end{aligned}$$

If we now define $\tilde{w}$ in the same way as $\tilde{u}$ in (5.14), we end up with the following expression

$$\left(\mathbf{\Lambda}_2 \otimes \widehat{\mathbf{B}}_1 + \mathbf{I}_2\left(\gamma_1\widehat{\mathbf{C}}_1 + \gamma_2\widehat{\mathbf{A}}_1\right)\right)\tilde{\mathbf{u}}^x - \left(\beta \mathbf{I}_2 \otimes \widehat{\mathbf{B}}_1\right)\tilde{\mathbf{w}}^x = \tilde{\mathbf{g}}^x,$$

where $\tilde{\mathbf{g}}^x$ is defined in (5.15). Let us now assume that we know $\tilde{\mathbf{w}}^x$. With the same notation as before we can write the system as

$$\left(\gamma_1\widehat{\mathbf{C}}_1 + \gamma_2\widehat{\mathbf{A}}_1 + \lambda_j\widehat{\mathbf{B}}_1\right)\tilde{\mathbf{u}}_j = \tilde{\mathbf{g}}_j + \beta\widehat{\mathbf{B}}_1\tilde{\mathbf{w}}_j \quad \text{for } j = 1, ..., N.$$

An iterative method is now introduced to solve the problem. In the first step we find $\widetilde{\mathbf{U}}$ from the above system of equations. Next, we compute $\mathbf{U}$ from (5.14). Then $\mathbf{W}$ can be computed as $w_{ij} = u_{ij}^3$ for $i = 1, ..., N-1$ and $j = 1, ..., N$. Finally we compute a new $\widetilde{\mathbf{W}}$ and start at step one again. The iteration continues until the relative error measured in the discrete L^2 norm is less than a given tolerance,

$$RE^k \equiv \frac{\| u_N^k - u_N^{k-1} \|_{L_N^2(\Omega)}}{\| u_N^k \|_{L_N^2(\Omega)}} < \text{TOL},$$

where TOL is the tolerance and u_N^k is the numerical solution at the k'th iteration. The algorithm is stated in **Algorithm 7**.

In the numerical results we choose $\widetilde{\mathbf{W}}^0$ to be the zero matrix. We will study the convergence rate of the algorithm and explore how it is influenced by the constant α.

Algorithm 7 Fast iterative solver for the nonlinear PDE

```
$k = 1$
**while** $RE^k > \text{TOL}$ **do**
    $\tilde{\mathbf{u}}_j = \left(\gamma_1\widehat{\mathbf{C}}_1 + \gamma_2\widehat{\mathbf{A}}_1 + \lambda_j\widehat{\mathbf{B}}_1\right)^{-1}\left(\tilde{\mathbf{g}}_j + \beta\widehat{\mathbf{B}}_1\tilde{\mathbf{w}}_j^{k-1}\right)$ for $j = 1, ..., N-1$
    $\mathbf{U}^k = \widetilde{\mathbf{U}}\mathbf{S}_2^T$
    $w_{ij} = (u_{ij}^k)^3$ for $i = 1, ..., N-1$, $j = 1, ..., N$
    $\widetilde{\mathbf{W}}^k = \mathbf{W}\mathbf{S}_2^{-T}$
    $k = k + 1$
**end while**
**return** $\mathbf{U}^k$
```

5.8 Numerical results

We consider the nonlinear time-dependent PDE (5.22) with the analytical solution

$$u(x,t) = \sin(\pi x)\sin(\pi t) \quad \text{in } \Omega = \omega_x \times \omega_t = (0,1)\times(0,2).$$

The iterative method in **Algorithm 7** solves the problem. The exact solution is infinitely smooth. We therefore expect exponential convergence of the problem as N increases. Figure 5.5 illuminates this when $\alpha = 1$. From numerical experiments we know that the iterative method does not converge when $\alpha \geq 5$. The numerical solution blows up to infinity. This is reasonable, as the solver is a stationary iterative method, and we know that other such methods have a limited convergence.

Next, we explore the convergence rate of the iterative method. One way of finding the convergence rate is to measure the relative error for each iteration k

$$\frac{\|u_N^k - u_N^{k-1}\|_{L_N^2(\Omega)}}{\|u_N^k\|_{L_N^2(\Omega)}},$$

and let k go to infinity. In Figure 5.6 the relative error is plotted as a function of k. The relative error is plotted on a logarithmic scale for different $\alpha's$. We see that the method obtains exponential convergence

$$\frac{\|u_N^k - u_N^{k-1}\|_{L_N^2(\Omega)}}{\|u_N^k\|_{L_N^2(\Omega)}} \sim e^{-\mu k}.$$

The convergence rate μ decreases when α increases. In Table 5.8, μ is calculated for different α's.

α	μ
0.01	4.6
0.1	4.2
1	2.2
2	1.6
3	1.0
4	0.8

Table 5.1: The convergence rate μ of the iterative method solving the nonlinear time-dependent PDE (5.22) with exact solution $u(x,t) = \sin(\pi x)\sin(\pi t)$. The convergence rate depends on α and is calculated from $\|u_N^k - u_N^{k-1}\|/\|u_N^k\| \sim e^{-\mu k}$, where the norm used is the discrete L_2 norm.

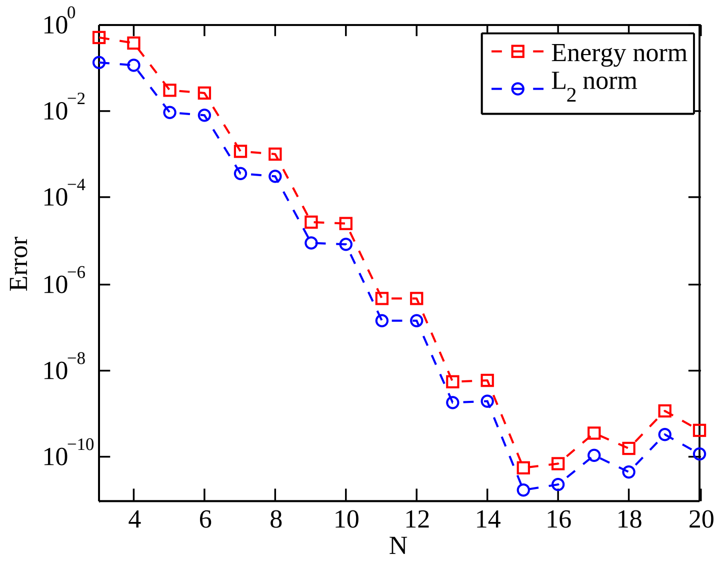


Figure 5.5: The discretization error $u(x) - u_N(x)$ measured in the discrete L^2 norm $\|\cdot\|_{L^2_N(\Omega)}$, and energy norm $\|\!|\cdot|\!\|_N$, as a function of the polynomial degree N. $u_N(x)$ is the numerical solution to the nonlinear time-dependent PDE (5.22) in the domain $\Omega = (0,1)\times(0,2)$, where the exact solution is $u(x,t) = \sin(\pi x)\sin(\pi t)$. The error is plotted on a logarithmic scale.

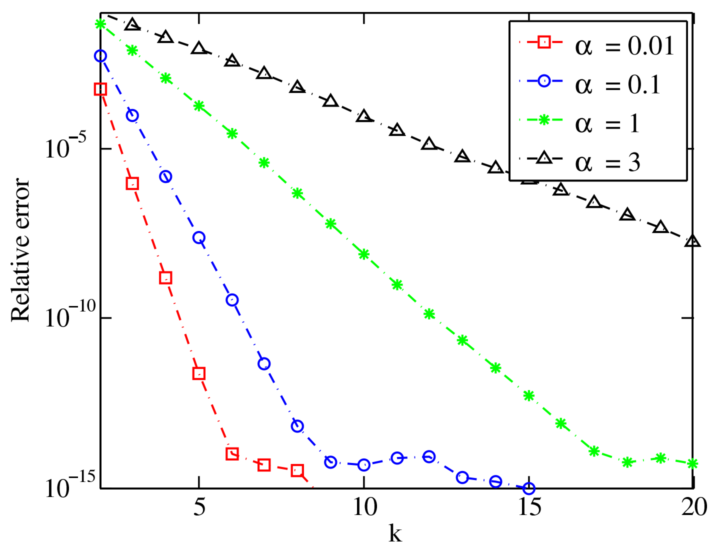


Figure 5.6: The relative error $\|u_N^k - u_N^{k-1}\|/\|u_N^k\|$ for different α's is measured in the discrete L_2 norm as a function of the iteration k. The relative error is plotted on a logarithmic scale and the tolerance is TOL= 10^{-15}.

Chapter 6

Conclusion

Spectral methods based on high order polynomials are powerful tools for finding numerical solutions to PDE's. What makes spectral methods unique compared to other methods is that the convergence rate depends on the regularity of the exact solution and the given data.

The spectral discretization leads to an algebraic system of equations. The computational complexity of solving this system depends on the solution method, and the difference in the computational cost can be significant. Two key ingredients for finding a fast solver is to exploit the tensor-product properties and take advantage of the local data structure.

The Poisson problem in a rectangular domain is studied using spectral discretization based on the Galerkin method. The fast tensor-product solver for this problem has a computational complexity of $\mathcal{O}(N^{d+1})$ floating point operations and $\mathcal{O}(N^d)$ floating point numbers for $\mathcal{O}(N^d)$ unknowns. If the exact solution is infinitely smooth, the spectral method has an exponential convergence rate. Tensor product solvers can also be derived for other discretization methods. Applying central differences for solving the Poisson problem results in a fast tensor-product method with equivalent computational complexity, but the convergence rate for smooth functions is only algebraic.

The Poisson problem is also studied in deformed domains using a spectral Galerkin method. We try to derive a fast tensor-product solver, but so far we have not succeeded. Iterative methods, such as the conjugate gradient method, can instead be applied to solve the algebraic system of equations. In this case it is crucial that we exploit the local data structure in order to derive a fast solver. In the conjugate gradient method one iteration has a complexity of $\mathcal{O}(N^{d+1})$ for $\mathcal{O}(N^d)$ unknowns. In deformed domains the approximated solution does not only depend on the regularity of the function, but also on the regularity of its geometry, due to the isoparametric mapping.

Fast tensor-product solvers can also be derived for linear time-dependent PDE's. The unsteady convection-diffusion equation in $\mathbb{R}$ is approximated using spectral discretization in both space and time. The solution algorithm is fully implicit and we end up at solving $\mathcal{O}(N)$ independent systems of equations. Each system is solved directly via Gaussian elimination at a cost of $\mathcal{O}(N^3)$. Spectral elements in time are used to illustrate the exponential convergence rate for smooth functions.

At last, fast tensor-product solvers are studied for nonlinear time-dependent PDE's. Discretizing the nonlinear term resulted in some difficulties. An iterative tensor-product solver is introduced to solve the derived algebraic system of equations. The method obtains exponential convergence rate for smooth functions when the nonlinear term is not too dominant.

Tensor product solvers are fast and accurate methods that can be applied on a number of PDE's. In this report we have focused on model problems with homogeneous Dirichlet boundary conditions, but it can also be applied to nonhomogeneous Dirichlet, homogeneous Neumann, nonhomogeneous Neumann and mixed boundary conditions. The methods are mainly discussed for the rectangular domain in $\mathbb{R}^2$, but they are easily extended to $\mathbb{R}^d$, $d > 2$.

Bibliography

[1] Canuto C, Hussaini MY, Quarteroni A, Zang TA. Spectral Methods Fundamentals in Singel Domains. Springer, Berlin; 2006.

[2] Morchoisne Y. Pseudo-spectral space-time calculations of incompressible viscous flows. In: W Unkel, J Schwoerer, J D Teare, & J F Louis, editor. AIAA, Aerospace Sciences Meeting. American Institute of Aeronautics and Astronautics; 1981. p. 8.

[3] Morchoisne Y. Inhomogeneous flow calculations by spectral methods - Monodomain and multi-domain techniques. NASA STI/Recon Technical Report A. 1982;83.

[4] Maday Y, Rønquist EM. Fast tensor product solvers: Part II: Spectral discretization in space and time; 2007, (unpublished).

[5] Bar-Yoseph P, Moses E, Zrahia U, Yari AL. Space-Time Spectral Element Methods for One-Dimensional Nonlinear Advection-Diffusion Problems. Journal of Computational Physics. 1994;119(1):62–64.

[6] Hesthaven JS, Gottlieb S, Gottlieb D. Spectral Methods for Time-Dependent Problems. Cambridge University Press; 2007.

[7] Lynch RE, Rice JR, Thomas RH. Direct Solution of Partial Differential Equations by Tensor Product Methods. Numerische Mathematik 6. 1964;p. 185–199.

[8] Jost J. Partial Differential Equations. vol. 214 of Graduate Text in Mathematics. 2nd ed. pages 187,341: Springer; 2007.

[9] Cormen TH, Leiserson CE, Stein C. Introduction to Algorithms. 2nd ed. The MIT Press; 2005.

[10] Rønquist EM. Lecture Notes, MA8502 - Numerical solution of partial differential equations; 2008.

[11] Brenner SC, Scott LR. The Mathematical Theory of Finite Element Methods. 3rd ed. Texts in Applied Mathematics. Springer; 2008.

[12] Deville MO, Fischer PF, Mund EH. High-Order Methods for Incompressible Fluid Flow. Cambridge; 2002.

[13] Nocedal J, Wright SJ. Numerical Optimization. 2nd ed. Springer Series in Operations Research. Springer; 2006.

[14] Gordon W, Hall C. International Journal for Numerical Methods in Engineering. vol. 7. pages 461-477; 1973.

[15] Kreyszig E. Advanced Engineering Mathematics. 8th ed. pages 446-458: Peter Janzon; 1999.